Analytics and ACLScript 15.0 Help

Table of contents

Using Analytics

Using Analytics

Analytics provides you with a wide-ranging set of tools for working with data. Beginning with the import of data, Analytics gives you numerous options for progressing through the data analysis cycle.

Analytics does not impose or require any particular data analysis workflow. The commands, functions, and other tools in Analytics can be assembled into whatever workflow you arrive at for analyzing a particular data set and achieving your analysis objectives.

That said, understanding the general data analysis cycle can help you structure your work in Analytics.

The data analysis cycle

The data analysis cycle contains five stages, which are summarized by the acronym **PIPAR**:

Plan, **I**mport, **P**repare, **A**nalyze, **R**eport

"Plan your work" on page 36	Planning your data analysis work is an important precursor to actually beginning the analysis in Analytics. Make sure to review "Plan your work" on page 36.
Import the data	You must import data into Analytics before you can analyze it.
Prepare the data	Often you must perform one or more data preparation tasks before data is ready to analyze.
Analyze the data	You perform analysis in Analytics by using commands and other tools to gain general insights about the data you are investigating, and to answer specific questions.
Report the results	Once your data analysis is complete, Analytics gives you several different ways to report or present your results.

> **Tip**
> To get an understanding of how the data analysis cycle can work in Analytics, do the introductory tutorial: "Getting started with Analytics (non-Unicode edition)" on page 35

Getting started

This section of the Analytics Help provides a variety of introductory and overview information, including:

"What is Analytics?" on the facing page	A high-level overview of Analytics features, and the end-to-end process of analyzing data using Analytics.
"What is ACL for Windows?" on page 32	Information about the ACL for Windows installation package and the ACL for Windows main screen.
"Getting started with Analytics (non-Unicode edition)" on page 35	A beginner-level, one-hour tutorial that introduces you to the end-to-end process of analyzing data using Analytics. Recommended for all new users of Analytics.
"Get help with Analytics" on page 103	Where to go for help as you are using Analytics.
"The Analytics user interface" on page 106	An overview of the Analytics interface, including customizable elements.
"Analytics projects" on page 163	Information about Analytics projects, which you use to contain and organize your work in Analytics.

What is Analytics?

product_description

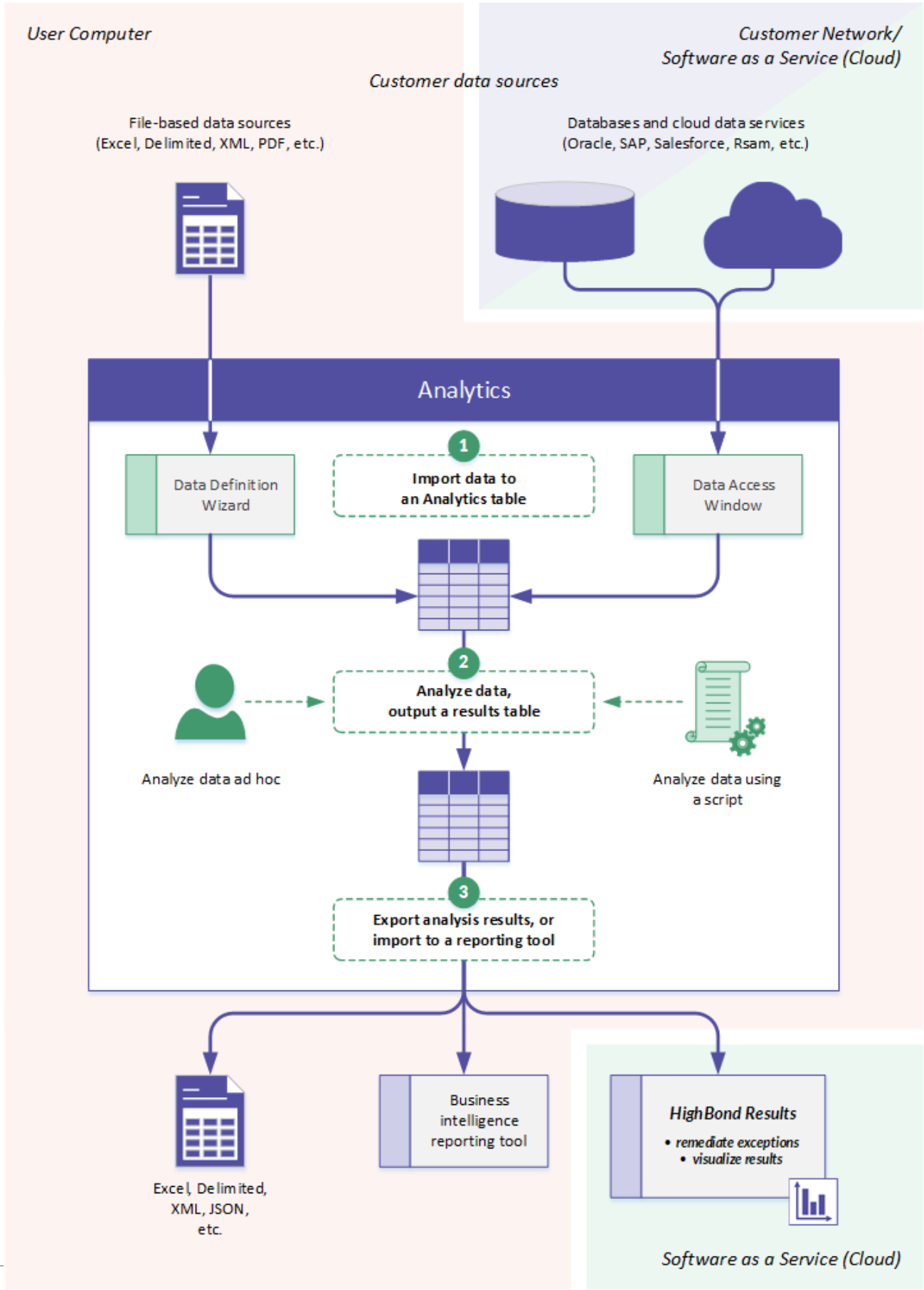
- **Data access** – Import a wide variety of different types of data from file-based data sources, databases, or cloud data services.
- **Data analysis** – Use Analytics commands, functions, and other tools to gain general insights about the data you are investigating, and to answer specific questions. You can perform data analysis ad hoc with the user interface, or automate your analysis using ACLScript, the powerful scripting language in Analytics.
- **Reporting** – Report your findings using native reporting features in Analytics, or import Analytics data into a third-party reporting tool such as Tableau.
- **Export capabilities** – Export findings, or any other data, to popular file types such as Excel or delimited text. You can also upload records to the Results app in the HighBond platform for processing and issue remediation using workflow automation tools, and for data visualization.

Basic workflow

The diagram below shows the basic workflow associated with Analytics:

1. Import data to an Analytics table
2. Analyze the data and output a results table
3. Export the analysis results, or import them to a reporting tool

   This step is optional. You can also use native reporting features in Analytics.



Working with data

Analytics provides immediate visibility into transactional data critical to your organization. The application reads and compares data, but does not allow modification of source data to ensure that it remains intact for complete data quality and integrity.

For more information, see "Data access by Analytics is read-only" on page 228.

Analytics allows you to work with data in the following ways:

- Analyze entire data populations, or samples of populations
- Identify trends and exceptions, and highlight potential areas of concern
- Identify control issues and ensure compliance with your organization's standards
- Age and analyze financial or time-sensitive transactions
- Automate analytic testing and receive immediate notification of results
- Log the analysis performed, allowing you to preserve analysis steps, and review and compare results

Unicode and non-Unicode editions

Analytics is available in Unicode and non-Unicode editions. Both editions are contained in the same installation package, and during the installation you specify which edition to install. In Analytics, in the dialog box containing the product and subscription information (**Help > About**), **Unicode** or **non-Unicode** appears after the version number.

For more information, see "Galvanize Unicode products" on page 2576.

Product name change

Starting with version 11.4, Analytics is a component of ACL for Windows, which also includes the Analysis App window and Offline Projects.

Prior to version 10.0, Analytics was called ACL Desktop. The short form of the name was *ACL*.

What is ACL for Windows?

ACL for Windows is a single, downloadable product that provides access to:

- Analytics
- the Analysis App window
- Robots
- Results
- Offline Projects

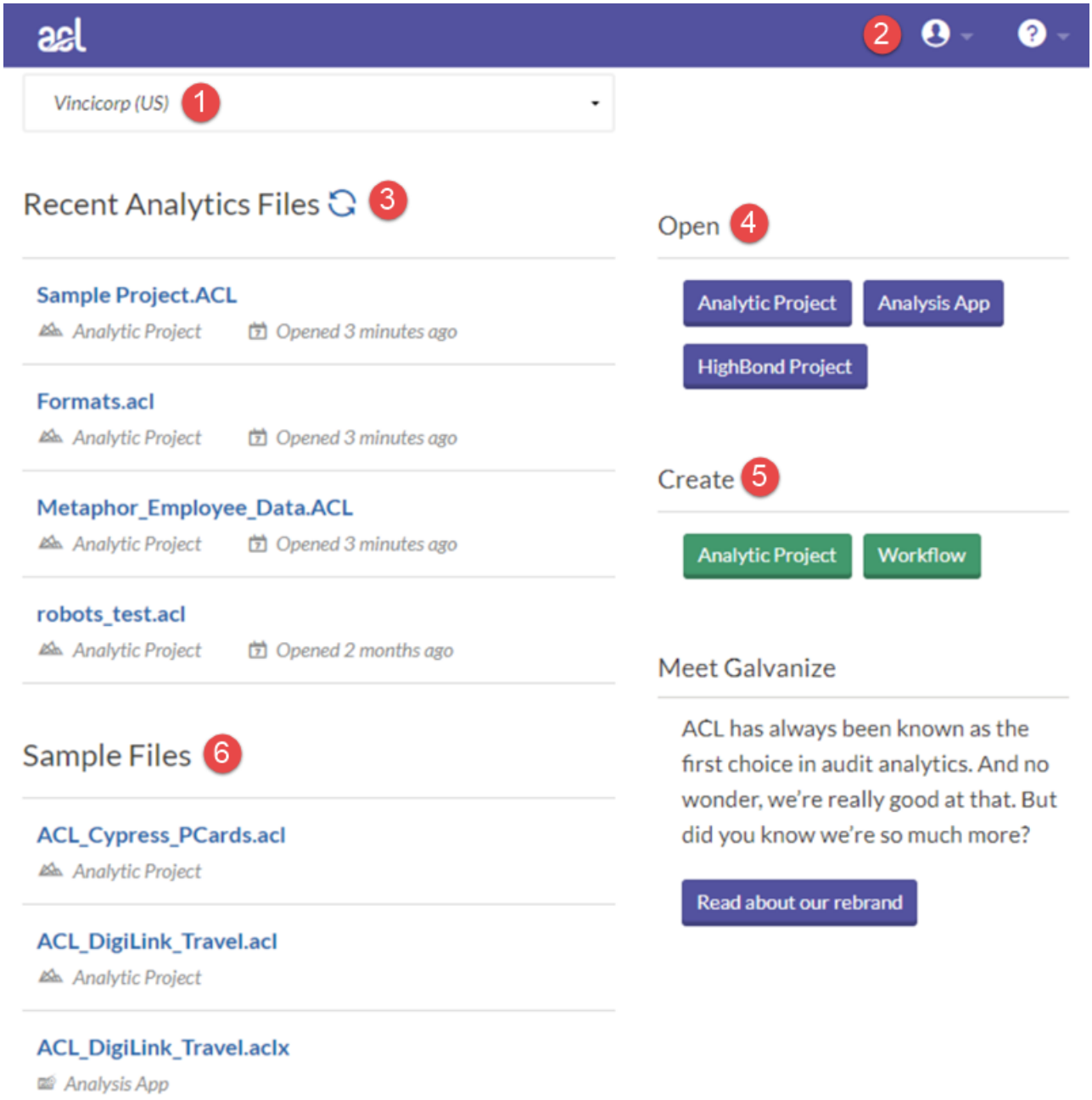
> **Note**
> Access to each component is determined by your Galvanize subscription type.

What can I access using the ACL for Windows main screen?

Using ACL for Windows, you can:

- activate Analytics to begin using the product
- create or open an Analytics project in Analytics, or open an analysis app in the Analysis App window
- connect to HighBond Results, where you can build workflows for organizing, tracking, and remediating exceptions
- access Launchpad, which provides links to all HighBond apps for HighBond users, and links to ScriptHub, Inspirations, and your user profile
- use Offline Projects to check out or check in a section from a HighBond project, and perform work offline
- access resources such as Support, quick start guides, product forums, online Help, and training courses

Overview of ACL for Windows main screen



Components

Number	Component	Description
1	**Organization selector**	Switch between accounts (organizations) you have access to using this drop-down list.
2	**Toolbar**	**Profile** - Update your profile or sign out of ACL for Windows. **Information** - Access help and product documentation or contact Support.
3	**Recent Analytics Files**	View recently accessed files in Analytics. Click **Refresh List** to update the list of files, or press **F5**.
4	**Open**	Open an existing Analytics project, analysis app, or HighBond project
5	**Create**	◦ **Analytic Project** - Create a new Analytics project and open it in Analytics. ◦ **Workflow** - Open collections in Results for viewing, or build a new workflow for organizing, tracking, and remediating exceptions
6	**Sample Files**	Open pre-built Analytics projects that include a variety of sample data.

Switching your HighBond instance

You might belong to multiple instances of HighBond if your organization has more than one subscription, you consult for multiple organizations, or you are part of a training instance. If you belong to more than one HighBond instance, you can use ACL for Windows to switch between them.

Switching between HighBond instances allows you to activate ACL for Windows using different subscriptions, and to access data belonging to different organizations or business units.

To switch your HighBond instance:

1. In ACL for Windows, select **Sign Out and close** from the profile drop-down list.

   You are signed out of your current instance.
2. Double-click the ACL for Windows shortcut on the desktop.

   The Launchpad sign-in screen opens.
3. Sign in using your HighBond account, by entering your user name (email) and password and clicking **Sign In**.
4. Select the appropriate instance from the drop-down list and click **Activate Analytics**.

   ACL for Windows opens. Any activities you perform involving HighBond now use the instance you just selected.

Getting started with Analytics (non-Unicode edition)

This getting started tutorial introduces you to the end-to-end process of analyzing data using Analytics.

Estimated time	60 minutes
Requirements	No previous Analytics experience is required. Some basic data analysis experience is assumed, but not absolutely critical.
Analytics version	13.0 or later (non-Unicode edition)
Do the right version of the tutorial	Do this version of the tutorial if you're using the non-Unicode edition of Analytics. If you're using the Unicode edition, do "Getting started with Analytics (Unicode edition)" on page 69.

> **Tip**
> To find out which edition of Analytics you're using, on the Analytics main menu, click **Help > About** to open the **Analytics** dialog box. The edition designation appears after the version number.

> **Note**
> The Chinese and Japanese user interfaces are Unicode-only.

Scenario

Review corporate credit card transactions

You're asked to review corporate credit card transactions from a two-month period. Your goal is to get a general picture of how employees used cards during the period, and also to identify any possible misuse of cards.

The transaction data is contained in three separate Excel worksheets. Before you can analyze the data, you need to import it into Analytics, and combine the separate data sets into a single Analytics table.

After you've analyzed the data, you want to present the results of your analysis visually, to better engage your audience.

Optional section

You're told that from now on, reviewing corporate credit card transactions will be a recurring responsibility.

To allow yourself, or someone else, to perform future reviews quickly and accurately, you decide to create a script to automate some of the work.

PIPAR – the data analysis cycle in Analytics

The data analysis cycle in Analytics contains five stages, which are summarized by the acronym **PIPAR**:

Plan, **I**mport, **P**repare, **A**nalyze, **R**eport

Plan your work

Planning your data analysis work is important, and often critical. If you skip the planning stage, and jump straight into running analytical commands against data, you may run into problems, create extra work for yourself, or even miss important analytical insights.

Even a basic plan is better than no plan. With experience, and increasing knowledge of Analytics, your planning will become more fully developed and more precise. Good planning is the key to data analysis projects that progress smoothly and efficiently.

Planning guidelines

Develop clear, specific objectives

What is the intended end product of your analysis?

You need clearly defined objectives in order to be able to plan how to achieve them. For example, in this tutorial, your specific objectives are:

- identify the count, and total amount, of corporate credit card transactions in each merchant category
- identify any transactions in prohibited categories

Map a step-by-step approach

How will you achieve your objectives?

Accomplishing an objective often requires more than one step, so map a detailed, step-by-step approach to guide you along the way.

For example, two of the steps in the planning for this tutorial could be:

- combine all the individual transaction files into a single file
- group the combined transaction data into merchant categories

Once you've broken down the larger objectives into individual steps, you can consider which Analytics features and functions to use to perform each step.

Identify what data you'll need

What data do you need to achieve your objectives?

Itemize the required source data to the level of specific data elements or fields. You won't be able to achieve your desired output without the appropriate input.

In this tutorial you have the main transaction files, but to achieve your second objective you'll also need a list of prohibited merchant category codes.

Consider technical requirements

Are there any technical considerations you must take into account?

Regardless of which tool you're using for data analysis, you must work within its constraints. Is the source data stored in a location or a system that the tool can access, and in a format that it can read? Is the analysis you're proposing supported by the tool?

For example, in order to combine multiple tables in Analytics, the data types of the corresponding fields in each table must be the same. Analytics supports changing the data type of a field, but that's a step you need to account for in your planning.

Be prepared to iterate

You may need to adjust your plan as you go along.

In the course of your analysis, you discover something unexpected that warrants further investigation. You realize you need additional data and additional analytical steps.

Your plan can evolve as your understanding of the data evolves. And it can serve as the basis for a more mature plan for future analysis of a similar nature.

Import data

You must import data into Analytics before you can analyze it.

We'll familiarize with the import process by using the **Data Definition Wizard** to import three Excel worksheets. Importing from Excel is one of the most common methods for acquiring data for analysis in Analytics. However, Analytics supports importing data from a wide variety of data sources.

Open Analytics and "Sample Project.ACL"

> **Note**
> The steps below assume you have already activated Analytics.

Steps

1. Double-click the **ACL for Windows** shortcut on your desktop.
2. In the **ACL for Windows** screen, under **Open**, click **Analytic Project**.
3. Navigate to `C:\Users\user_account_name\Documents\ACL Data\Sample Data Files` and double-click `Sample Project.ACL`.

   `Sample Project.ACL` opens in Analytics.

   If you did not install the `Sample Data Files` folder in the default location when you installed Analytics, navigate to the location where you installed it.

Import the first two Excel worksheets

You will get started by importing two Excel worksheets at the same time. Importing multiple Excel worksheets simultaneously is a great way to reduce labor.

Steps

1. From the Analytics main menu, select **Import > File**.
2. In the **Select File to Define** dialog box, locate and select `Trans_May.xls` and click **Open**.

   The Excel file is in the same folder as `Sample Project.ACL`.

3. In the **File Format** page, make sure the **Excel file** option is selected and click **Next**.
4. In the **Data Source** page, select both worksheets in the file:
   - `Trans1_May$`
   - `Trans2_May$`
5. Make sure **Use first row as Field Names** is selected, click **Next**, and then click **Finish**.

   The two Excel worksheets are imported into two separate Analytics tables.

Import the third Excel worksheet

Now import the third Excel worksheet by itself. When you import a single worksheet, you have the option of manually adjusting some of the metadata settings during the import process, rather than doing it later in Analytics.

Steps

1. Repeat the steps in the previous procedure to locate and select `Trans_April.xls`.
2. In the **File Format** page, make sure the **Excel file** option is selected and click **Next**.

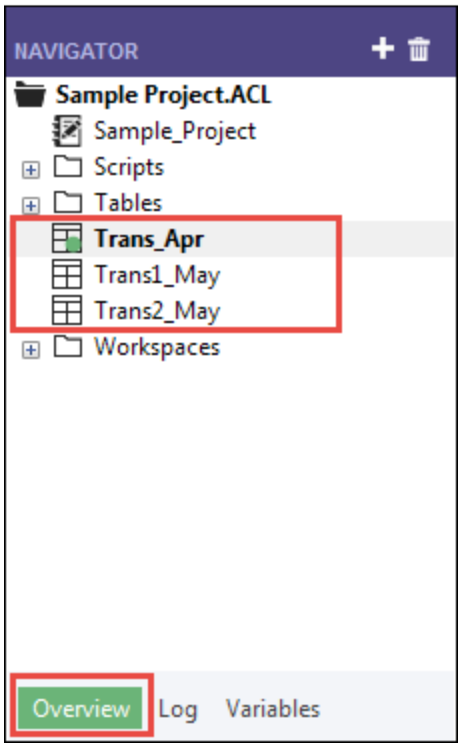
3. In the **Data Source** page, select `Trans_Apr$`.
4. Make sure **Use first row as Field Names** is selected, and click **Next**.
5. In the **Excel Import** page, click the header to select the **TRANS_DATE** column and make the following changes:
   - In the **Name** field, change `TRANS_DATE` to `DATE`.
   - In the **Length** field, change `19` to `10`.

   > **Note**
   > You're making adjustments to a data field in the Data Definition Wizard, during the import process. You can also make adjustments later, after you have completed importing the data. You'll see the reason for the adjustments in the next section of the tutorial.

6. Click **Next**, in the **File name** field type `Trans_Apr`, and click **Save**.
7. Click **Finish**, and then click **OK**.

   The third Excel worksheet is imported into an Analytics table.

You should now have three new Analytics tables in the **Overview** tab of the **Navigator**. These tables contain read-only copies of the Excel data. They do not contain the Excel source data itself.



Prepare data

Often you must perform one or more data preparation tasks before data is ready to analyze.

For this tutorial, you'll perform two preparation tasks:

- make additional adjustments to harmonize data fields
- combine the three new Analytics tables into a single table for analysis

As well, as a best practice, you should always verify the validity of imported data before performing analytical work. Even a small amount of invalid data in a table can invalidate all your subsequent data analysis.

Why do I need to prepare data?

You're eager to get on with the analysis of the data, but without proper data preparation you may not be able to perform the analysis. Or the analysis you perform may be flawed.

A wide variety of issues can affect source data making it unsuitable for analysis without some initial preparation.

For example:

- The source data is spread between several different files and needs to be consolidated so that it can be analyzed as a single set of data.
- Corresponding fields in different files need to be "harmonized", which means making them identical in structure and format as a prerequisite to processing them.
- "Dirty data" needs to be cleansed and standardized, which you can do with Analytics functions.

Key point

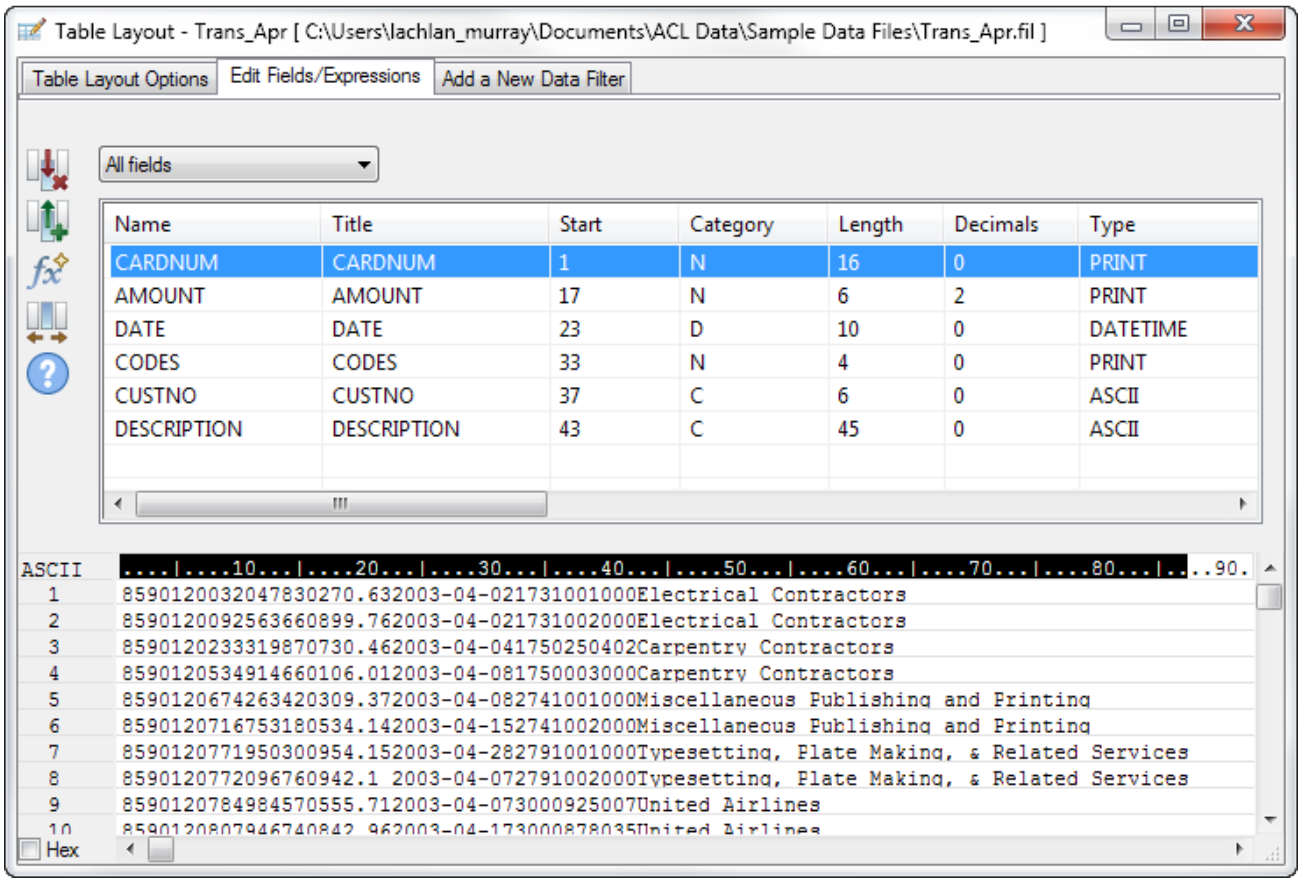
The time you spend importing and preparing data may exceed the time you spend on the actual analysis. However, they are critical initial stages, and provide the foundation that your analysis is built on.

Adjust the table layouts

Every table in an Analytics project has a **table layout**. The table layout contains metadata such as field names, the start position of fields, the length of fields, the data type of fields, and so on.

Before we can combine the three new Analytics tables into a single table, we need to harmonize some of the metadata in the table layouts.

Here's what the **Trans_Apr** table layout looks like. You'll quickly learn your way around table layouts as you become more familiar with Analytics. You can do a lot of useful things in the table layout.



Adjust the Trans_Apr table layout

First, you need to change the data type of two fields in the **Trans_Apr** table.

Steps

1. Open the **Trans_Apr** table, if it is not already open.

   To open a table, double-click it in the **Navigator**.

2. Above the table view, click **Edit Table Layout**.
3. Double-click the **CARDNUM** field to open the field definition for editing.
4. Under **Valid Data Types**, double-click **ASCII** to update the data type of the field.

   In the other two tables, the **CARDNUM** field has an ASCII data type. For the most part, combining data requires that corresponding fields in the tables being combined have the same data type.

5. Click **Accept Entry**.

   If a prompt appears, click **Yes** to save your changes.

6. Double-click the **CODES** field and change the data type to **ASCII**.
7. Click **Accept Entry**, and then click Close to exit the **Table Layout** dialog box.

Adjust the Trans_May table layouts

To finish the adjustments, you need to change the data type of two fields in both the **Trans1_May** and the **Trans2_May** layouts. You may also need to make an adjustment to the **DATE** field.

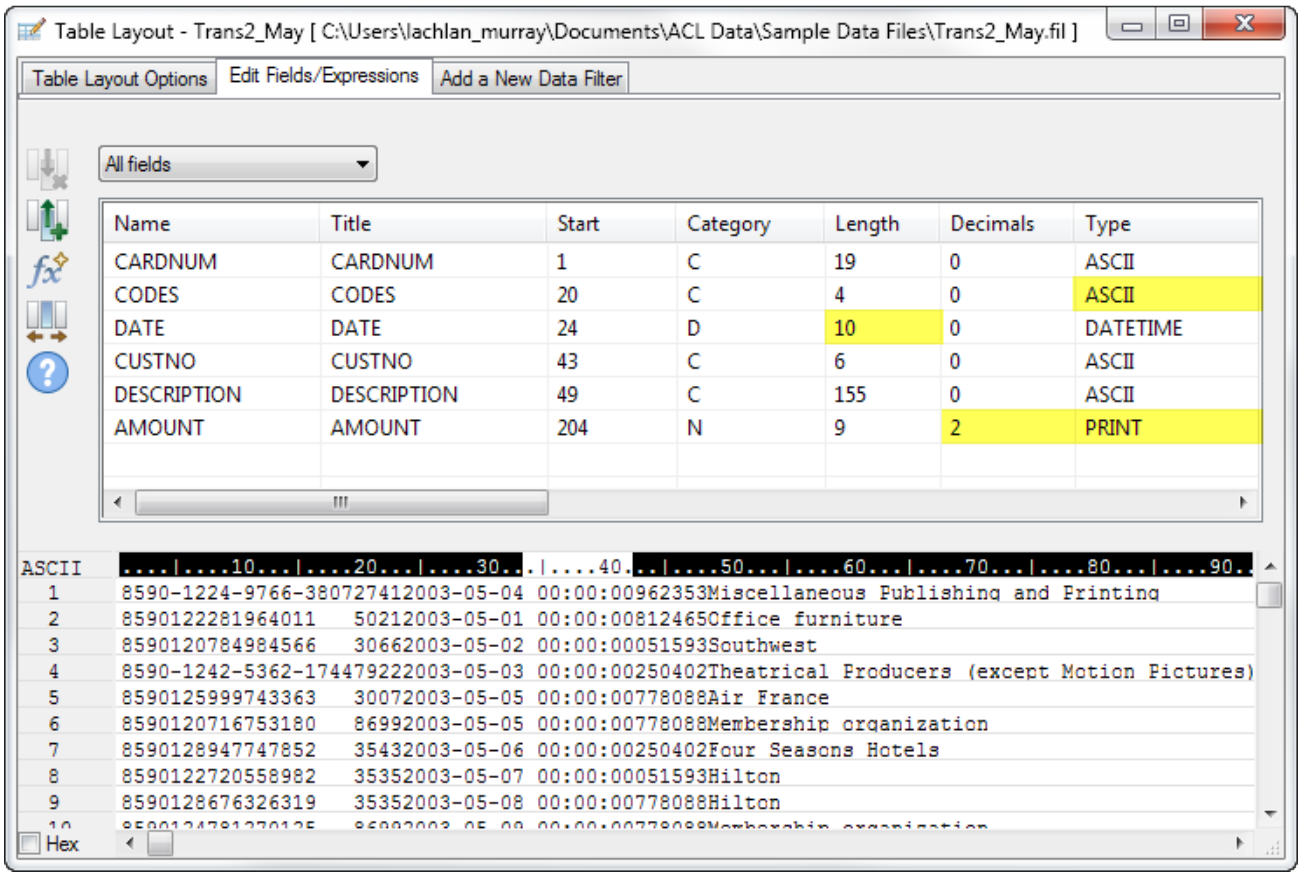
Steps

Follow the process above to make the following changes in both the **Trans1_May** and the **Trans2_May** layouts:

Field	Change data type to:	Additional change
CODES	ASCII	
AMOUNT	PRINT	Enter `2` in the **Dec.** field to specify that numeric values display two decimal places.
DATE	no change	**Note** If the DATE field already has a length of 10, no adjustment is required. ◦ In the **Len.** field, change `19` to `10`. This change omits the empty time data. ◦ In the **Format** dropdown list, select **YYYY-MM-DD**.

When you're finished, the May table layouts should look like the layout below.

> **Note**
> The date format (YYYY-MM-DD) isn't shown in the layout summary. The DESCRIPTION field length is different in the two May layouts.



Verify the imported data

Now let's verify the data in the three imported tables to make sure it's safe to proceed with additional data preparation, and data analysis.

> **Note**
>
> We're verifying the data **after** updating the data types. When you verify data in Analytics, you're checking that all the values in a field conform to the requirements of the field's data type. So it makes sense to verify data only once the data types are finalized.

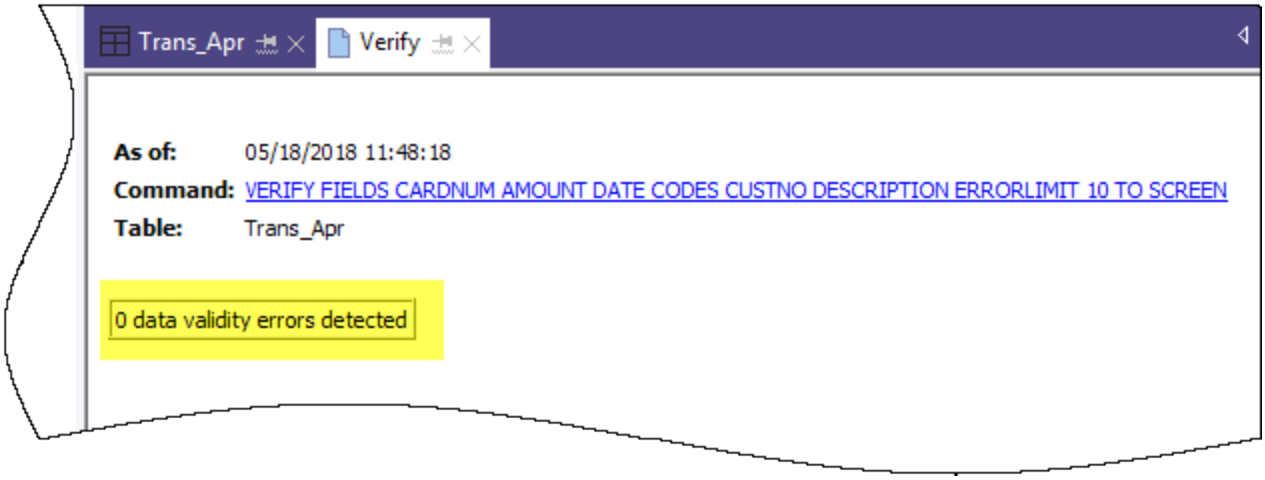
Steps

1. Open the **Trans_Apr** table.
2. From the Analytics main menu, select **Data > Verify**.
3. In the **Verify** dialog box, select all the fields in the field list.

> **Tip**
>
> Use **Shift+click** to select multiple adjacent fields.

4. Click **OK**.

   The result should be: **0 data validity errors detected**.



Learn more

Did you notice that Analytics automatically translated the action you performed in the user interface into the ACLScript `VERIFY` command? Every command-level action you perform in the user interface is automatically translated into its corresponding ACLScript command, and captured and stored in the command log that accompanies each Analytics project.

This automatic generation of valid, runnable script syntax is one of the most powerful features in Analytics. We'll be looking at scripting in an optional section at the end of the tutorial.

5. In the **Navigator**, double-click the **Trans1_May** table to open it, and repeat the steps to verify the data.
6. Do the same for the **Trans2_May** table.

   Both tables should not contain any data validity errors.

> **Note**
>
> If you get an error message stating **Maximum error limit reached**, check that you correctly changed the format of the **Date** field in the table layout to **YYYY-MM-DD**.

Learn more

If you want to see what happens when Analytics does identify data validity errors, open **Tables\Badfile** and run the verification process.

Combine the three Analytics tables

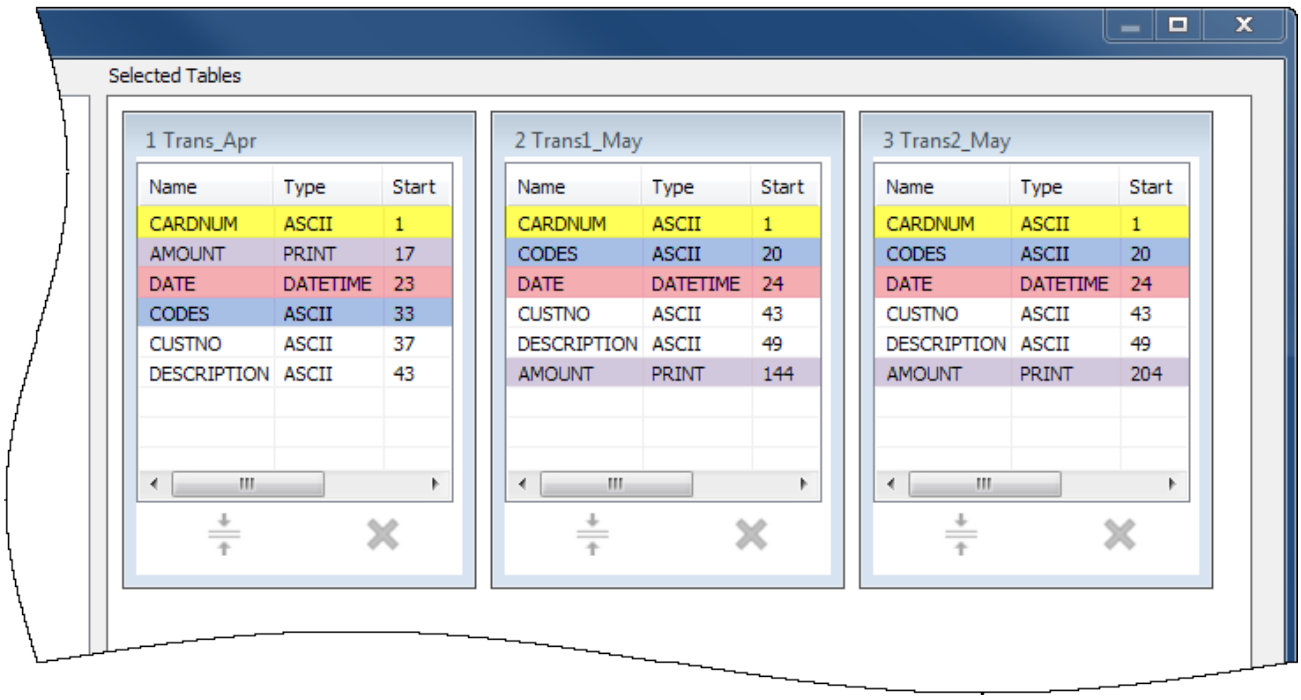
For the final data preparation task, you combine the three new Analytics tables into a single table.

For simplicity, the tutorial combines only three tables. However, you could use the same process to combine 12 monthly tables into a single annual table and perform analysis on data for an entire fiscal year.

Steps

1. From the Analytics main menu, select **Data > Append**.
2. Under **Available Tables**, double-click each of the new tables to add it to the **Selected Tables** area.
3. Take a look at the fields in the three tables and notice how the names and data types are identical based on the changes you made in the Data Definition Wizard and the **Table Layout** dialog box.

   To append corresponding fields, their names must be identical, and in most situations their data types must be identical.



4. Select **Use Output Table** so that the output table with the combined data opens automatically after you run the command.
5. In the **To** field, type `Trans_All` and click **OK**.

6. Click **Yes** in the notification that pops up.

> **Note**
> Don't worry about the notification. The append command performs some automatic harmonization of numeric fields, which saves you time and effort.

The new **Trans_All** table is created, and contains all the records from the three input tables. The record count in the status bar at the bottom of the Analytics interface should say **Records: 481**.

You're now ready to move on to some actual data analysis.

Analyze data

You perform analysis in Analytics by using commands and other tools to gain general insights about the data you are investigating, and to answer specific questions.

> **Note**
> The analysis stage is where the strength of your earlier planning becomes apparent. If you've formulated clear objectives regarding your investigation, you'll have a clearer idea of the types of analysis to perform.

The data analysis

For this tutorial, you'll perform the following analysis of the data in the **Trans_All** table:

- group the credit card transaction records by merchant category code in order to discover:
  - how employees are using corporate credit cards
  - how much money is being spent in each category
- create a filter to isolate any prohibited transactions

Group credit card transactions by merchant category code

Grouping or summarizing a set of data is an excellent way of quickly getting an overview of the data.

Steps

1. Open the **Trans_All** table, if it is not already open.
2. From the Analytics main menu, select **Analyze > Summarize**.

3. In the **Summarize** dialog box, select the following fields and options:

Tab	Field or option	Select or type
Main	Summarize On	select CODES
	Other Fields	select DESCRIPTION
	Subtotal Fields	select AMOUNT
	Avg, min, max	select the checkbox
Output	To	select File
	Name	type `Trans_All_Grouped`

4. Click **OK**.

   The new **Trans_All_Grouped** table is created. The table contains 110 records, one for each unique merchant category code in the **Trans_All** table. The **COUNT** field tells you how many source records are in each group.

> **Tip**
>
> Right-click the table view and select **Resize All Columns** to make the view more compact.

Simple tools for investigation

Now that you have a summarized version of the data, you can use some basic Analytics tools to gain general insight into corporate credit card use.

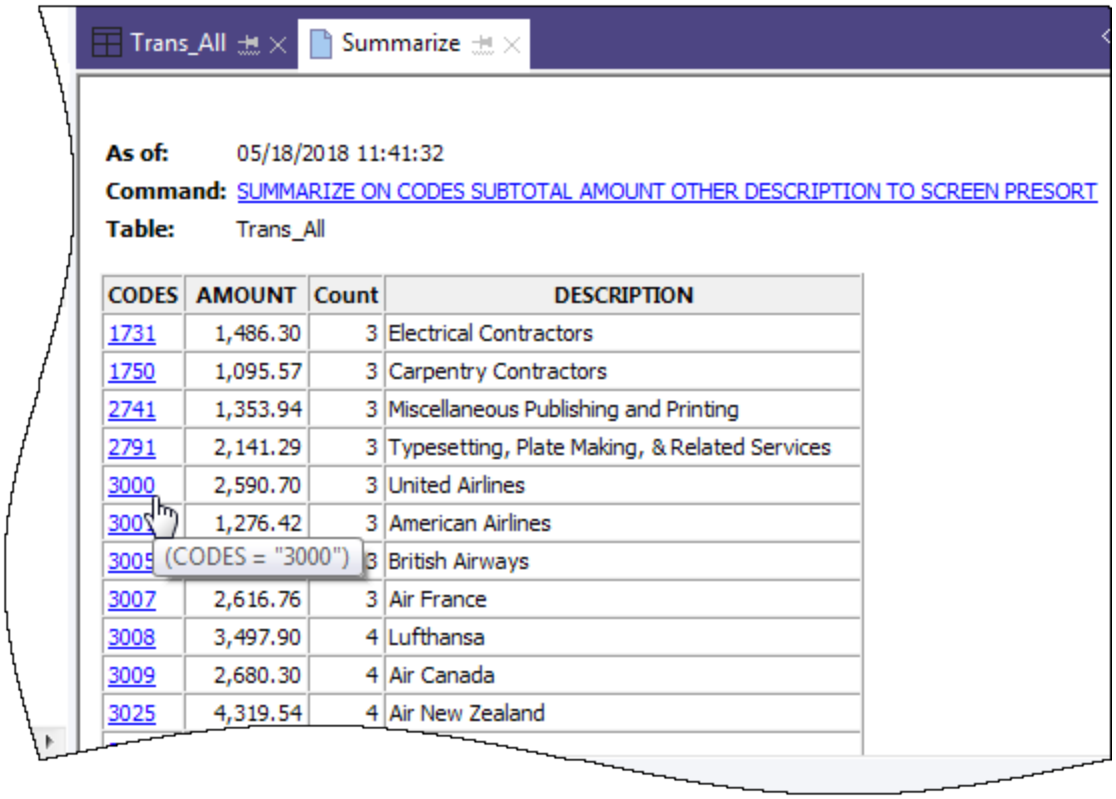
You can learn a lot about patterns of use, and possible misuse, in just a few clicks.

To gain this insight:	Do this in the Trans_All_Grouped table:
What was the total amount charged by employees during April and May?	◦ Select the **Total AMOUNT** header. ◦ Select **Analyze > Total**. Total expenditure was $187,177.13.
Where did employees spend the most money?	◦ Right-click the **Total AMOUNT** header and select **Quick Sort Descending** The **Description** field shows you that the most money was spent on: ◦ Caterers ◦ Eating places and Restaurants ◦ Hilton International
What were the largest single expenditures?	◦ Right-click the **Maximum AMOUNT** header and select **Quick Sort Descending** The **Description** and **Maximum AMOUNT** fields show you that the largest single expenditure was a Club Med amount of $1999.06. Is Club Med an authorized merchant code for the corporate credit card? If the credit card limit is $2000, was an employee charging an amount just under the limit?

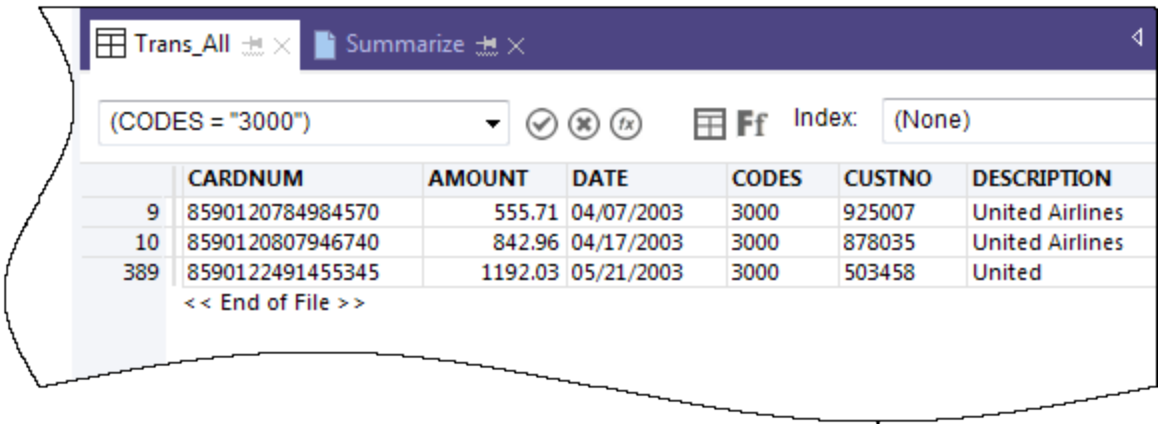
To gain this insight:	Do this in the Trans_All_Grouped table:
What does an examination of infrequently used codes reveal?	◦ Right-click the **COUNT** header and select **Quick Sort Ascending** Five categories had only a single charge each. Are some of them prohibited categories? Perhaps one or more employees thought that misusing a company card only very occasionally would allow them to escape detection. ◦ Cigar Stores & Stands ◦ Dating & Escort Svcs. ◦ Babysitting services ◦ Amusement Parks ◦ Civic, Fraternal, and Social Associations
Are any of the categories prohibited?	◦ Right-click the **DESCRIPTION** header and select **Quick Sort Ascending** to alphabetize the field values for easier scanning ◦ Scan down the field looking for suspicious categories Perhaps one or more of these categories are prohibited? ◦ Babysitting services ◦ Betting (including Lottery Tickets, Casino) ◦ Civic, Fraternal, and Social Associations ◦ Dating & Escort Svcs. ◦ Massage Parlors ◦ Precious Stones and Metals, Watches and Jewel ◦ Video Game Arcades/Establishments **Note** Manual scanning is impractical for all but small data sets. We'll look at a more practical, more reliable method next.

Learn more

Perhaps you just want to perform some quick analysis and you don't want to output the results to a new table. When you summarized the **Trans_All** table, instead of selecting **File** in the **Summarize** dialog box, you could select **Screen**, and output the results to the Analytics display area.



Outputting to screen is only practical for smaller data sets. However, it has the advantage of providing an easy way to drill-down on individual groups and see only the source records in each group.



Create a filter to isolate prohibited transactions

Filters allow you to isolate just the records you are interested in at a particular moment. Filters are a powerful tool for answering specific questions about data.

A general review of the corporate credit card transactions alerted you to some possible prohibited transactions. You decide to confirm whether any transactions are prohibited by matching a list of prohibited merchant category codes against the data.

Steps

Create the filter expression

1. Open the **Trans_All** table.
2. Click **Edit View Filter** at the top of the table view to open the **Expression Builder**.

   The **Expression Builder** is an Analytics component that lets you use the mouse to create expressions, rather than typing expression syntax manually. Expressions are combinations of values and operators that perform a calculation and return a result.
3. In the **Functions** drop-down list, select **Logical**, and then double-click the MATCH function to add it to the **Expression** text box.

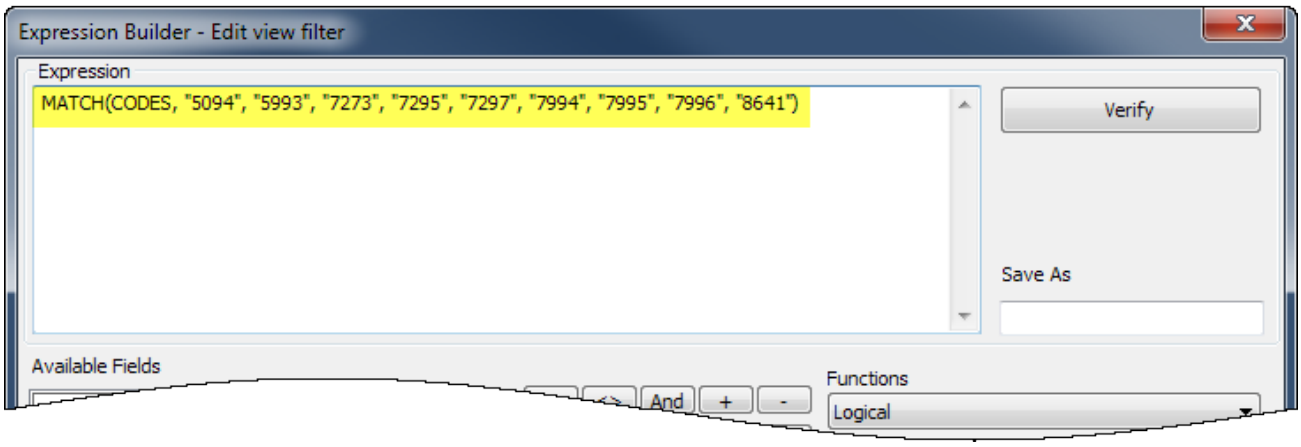
   You're going to use MATCH to isolate several prohibited merchant category codes in the **CODES** field.
4. In the **Expression** text box, highlight the `comparison_value` placeholder, and then in the **Available Fields** list, double-click **CODES**.

   The `CODES` field replaces `comparison_value`.
5. Copy the string of prohibited codes below and use them to replace the `test1 , test2 <,test3...>` placeholder:

   `"5094", "5993", "7273", "7295", "7297", "7994", "7995", "7996", "8641"`

   > **Note**
   > Make sure you copy the entire string, including all quotation marks.

   Your expression should look like this:



Verify the expression and save and apply the filter

1. Click **Verify** to test that the syntax of your expression is valid.

   Verifying expressions as soon as you create them is a best practice because it can help avoid more time-consuming troubleshooting later.

   If you get an error message, double-check that the syntax of the expression exactly matches the syntax shown above.

2. In the **Save As** field, type or copy the filter name `f_Prohibited_codes`.

   Galvanize recommends that you preface the names of saved filters with `f_`

3. Click **OK**.

   The **f_Prohibited_codes** filter is applied to the **Trans_All** table. Transactions that use a prohibited merchant category code are now isolated and plain to see. Consider a table with tens of thousands of records, or more, and the value of filters quickly becomes apparent.

Remove or reapply the filter

Try removing and reapplying the filter:

1. To remove the filter, click **Remove Filter**  .
2. To reapply the filter, do either of the following:
   - Select the filter name from the Filter history drop-down list at the top of the view.
   - Click **Edit View Filter**  to open the **Expression Builder**, double-click the filter name in the **Filters** list, and click **OK**.

> **Tip**
> The Filter history list holds a maximum of 10 filters, so at times you may need to use the **Expression Builder** method for reapplying a saved filter.

Learn more

Beyond filters

Filters work well if the number of criteria or conditions contained by the filter are manageable. The filter you created in this tutorial contains only 9 codes. But what if your list of prohibited merchant category codes was several dozen, or more?

A more efficient approach would be to join an Analytics table containing the prohibited codes with the transactions table. Every match in the joined output table would be a prohibited transaction.

Joins are beyond the scope of this tutorial, but they are a frequently used feature in Analytics.

Report results

Once your data analysis is complete, Analytics gives you several different ways to report or present your results.

Traditional reports with columns of data are available, but we'll look at conveying results using the more engaging data visualization described below.

Treemap visualization

This treemap visualization shows the grouped credit card transactions you output in the **Trans_All_Grouped** table. The relation between groups is conveyed in two different ways:

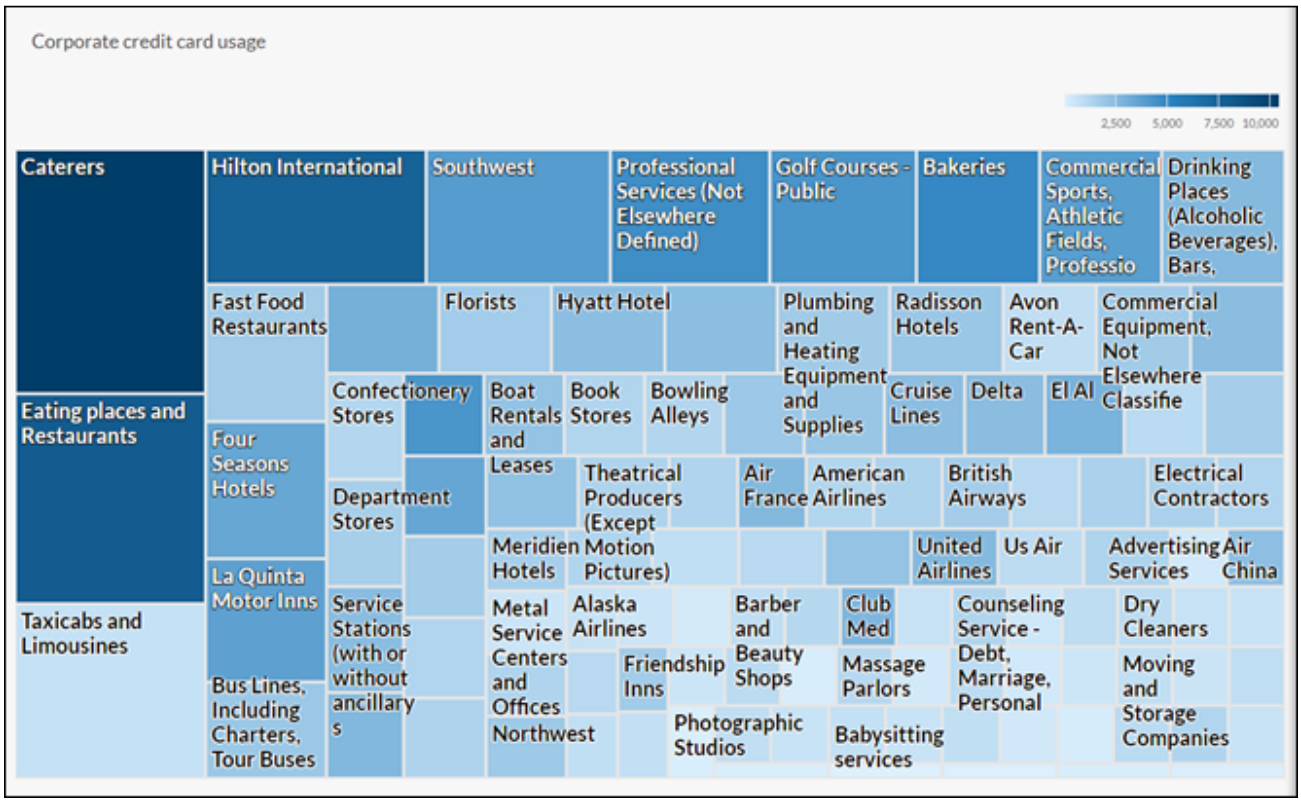
- **size of the box** – indicates the count of individual transactions in each group

  The larger the box, the greater the number of transactions. The boxes are arranged in size from top left to bottom right.

- **color intensity of the box** – indicates the total amount of each group

  The darker the box, the greater the total amount.

So, for example, the size of the **Club Med** box, in the bottom right quadrant, indicates only a small number of transactions, but the color indicates that the total transaction amount is significant.



First, a little pre-work

You're going to create the treemap visualization in Results, the issue remediation app in the cloud-based HighBond platform. Access to a lite version of Results is included in your ACL Robotics subscription.

In order to create the visualization, you must first create a simple, two-level data container to hold it. The first level is called a Collection, and the second level is called an Analysis. They're quick and easy to create.

Sign in to Launchpad and access Results

> **Note**
> If for some reason you cannot sign in to Launchpad or access Results, you can use one of the alternative report creation methods listed in "Other reporting methods in Analytics" on page 57.

Steps

1. Go to Launchpad (www.highbond.com).
2. Enter your HighBond account credentials (e-mail and password) and click **Sign In**.

   Launchpad opens.

3. Click **Results**.

   The Results homepage opens.

> **Note**
>
> If you cannot access Results, you may not be assigned an appropriate subscription type or Results role. Use one of the alternative report creation methods listed in "Other reporting methods in Analytics" on page 57.
>
> If you would like to access Results, contact your company's Analytics account administrator.

Create a Collection

Steps

1. From the Results homepage, click **New Collection**.
2. On the **New Collection** page, in the **Name** field, enter or copy `ACL Tutorial`.
3. At the bottom of the page, click **Create Collection**.

   The Collection settings page opens.

Create an Analysis

Steps

1. At the bottom of the Collection settings page, under **What's Next?**, click **create your first Data Analysis**.

   The **Analysis Details** page opens.

2. On the **Analysis Details** page, in the **Name** field, enter or copy `Sample Report`.
3. Click **Create Analysis**.

   The new **ACL Tutorial** Collection opens with the empty **Sample Report** Analysis that you just created.

> **Note**
>
> Leave Results open. You will be coming back to create the data visualization.

Export data from Analytics to Results

The next stage is to export the **Trans_All_Grouped** table from Analytics to Results.

Steps

1. In Analytics, open the **Trans_All_Grouped** table.
2. From the Analytics main menu, select **Data > Export**.

3. In the **Export** dialog box, select the following options:

Tab	Option	Select
Main	View	select View
	Export As	select HighBond

4. Click **To**, and in the **Select Destination Test** dialog box navigate to the **Sample Report** Analysis container you just created and double-click to open it.
5. In the **New data analytic** field enter or copy `Trans_All_Grouped` and click **Create**.

   You are returned to the **Export** dialog box and an ID number and data center code are prefilled in the **To** text box.
6. Click **OK**.

   The data in the **Trans_All_Grouped** table is exported to Results.

Create the visualization

Now you're ready to create the visualization in Results.

Steps

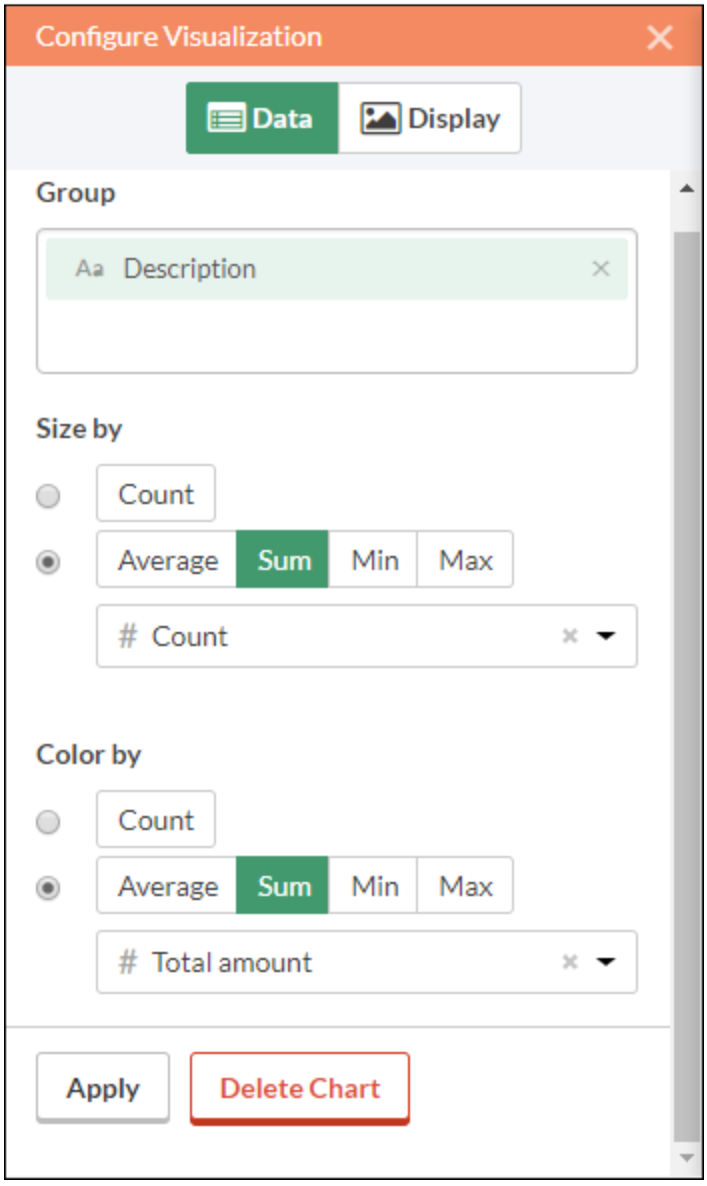
1. Return to the **ACL Tutorial** collection in Results and press **F5** to refresh the browser window.

   The **Trans_All_Grouped** table appears.
2. Under **Remediate**, click **View Records**.

   The **Table View** opens and displays the records.
3. Click **Add Visualization** and click the **Treemap** visualization.

4. In the **Configure Visualization** panel, select the fields and options shown below.

> **Note**
>
> If you can't see the **Configure Visualization** panel, click **Configure**  .



5. Click **Apply**.

   The Treemap visualization is generated.

   You can hover your mouse over the individual boxes in the treemap to see the embedded data.

   If you change the size of the browser window, the treemap dynamically updates by repositioning boxes, and by displaying and suppressing a different selection of associated descriptions.

Save the visualization

If you want to keep any visualizations you create you need to save them. You need to save each visualization individually, and also the container that holds them, called **an interpretation**.

Steps

1. Click **Untitled** at the top left corner of the Treemap visualization and type a title for the visualization such as `Transaction Treemap` and press Enter.
2. Click **Save > Save As** .
3. In the **Title** field, type a name for the interpretation such as `Tutorial visualizations` and click **Save**.

   The interpretation and the visualization are both saved and can be reopened later.
4. Click the name of the collection, `ACL Tutorial`, in title bar to return to the **Sample Report** Analysis container.
5. Click the number in the **Interpretations column**. The **Interpretations** dialog box appears and notice that it lists the newly created interpretation, `Tutorial visualizations`.

   You can create multiple visualizations and interpretations in each Analysis container. Each visualization is based on the data in the **Table View**.

Publish to Storyboards

Create a storyboard to display the visualization you just created. A storyboard is a communication platform that displays multiple visualizations and rich text content in a single presentation.

Steps

1. Open the **Storyboards** app.
2. Click **Add Storyboard**.
3. Enter a descriptive title for your storyboard. Storyboard titles can be a maximum of 80 characters.
4. Click **Add** and select **Add Chart**
5. Select one of the following options:
   - To display table view from the interpretation, select the parent table entry, `Tutorial visualizations`.
   - To display visualization from the interpretation, select the child chart entry, `Transaction Treemap`.

   You can enter a keyword or phrase into the search field to filter the list of available visualizations.
6. In the top right-hand corner, click **Save** > **Save**.

Other reporting methods in Analytics

In addition to the data visualizations available in Results, Analytics has several other methods you can use for reporting the results of your data analysis:

Reporting method	Description
Data visualizations in the Analysis App window	The data visualization capability in Results is also available locally in the Analysis App window, a freestanding component of Analytics.

Reporting method	Description
	Note Some of the charts and visualizations available in Results may not be available in the Analysis App window until a new version of Analytics is released. For more information, see "Interpretations and visualizations" on page 2652.
Legacy Analytics charts	Analytics contains a legacy charting and graphing capability that allows you to create basic visual reports. For more information, see "Working with Analytics graphs" on page 1336.
Traditional columnar reports	In some cases, a traditional text- and number-based report with rows and columns of data is all you need. For more information, see "Formatting and generating Analytics reports" on page 1330.
Third-party reporting tool	You can use a third-party reporting tool such as Tableau or Microsoft BI and import data directly from Analytics. For more information, see "Connecting to Analytics from a third-party reporting application" on page 1348.
Exporting data to Excel or CSV	You can export data to Excel, or to a comma-separated file, and use the reporting capabilities of Excel, or of any tool that can work with a CSV file. For more information, see "Exporting data" on page 203.

You're finished

Congratulations! You've completed your end-to-end introduction to analyzing data using Analytics.

Where to next?

You have several options for continuing to learn about Analytics:

Academy	Academy offers a range of courses for various experience levels. **ACL Analytics Foundations Program** is a series of six mini-courses that teaches Analytics basics for new users. Academy is the Galvanize online training resource center. Go to the course catalog to see the available courses. Academy courses are included at no extra cost for any user with a subscription.
Analytics and	You're currently in the Analytics and ACLScript Help. The Help provides reference-style

ACLScript Help	conceptual material, step-by-step instructions, and ACLScript syntax for all aspects of Analytics. For example, here are the Help topics for the append operation, which formed part of the tutorial you just completed: ◦ "Appending tables" on page 858 (conceptual) ◦ "Append tables" on page 868 (step-by-step instructions) ◦ "APPEND command" on page 1565 (ACLScript syntax)
Community	Community is a web-based platform with a variety of customer resources, including a customer forum where experienced Analytics users share their expertise and answer questions. The customer forum is the best place to learn about the real-world usage and application of Analytics.

Script your work (optional section)

Estimated time	20 minutes
Requirements	No previous scripting experience is required.
Analytics version	13.0 or later (non-Unicode edition)

You can gain a lot of value using Analytics in an ad hoc or manual fashion without ever writing a script. For the most part, anything that can be done in a script can be done in the user interface, and vice versa. However, to gain the most value, power, and efficiency from Analytics, you need to script.

The good news is that Analytics provides tools to make scripting relatively easy, even for a novice.

The case for scripting

Imagine that in addition to all your current responsibilities you're now responsible for reviewing corporate credit card usage on a regular basis.

Save time

The basic review process is standardized. With each review cycle, you can spend time repeating the basic process manually, or you can save time by automating the process.

Delegate with confidence

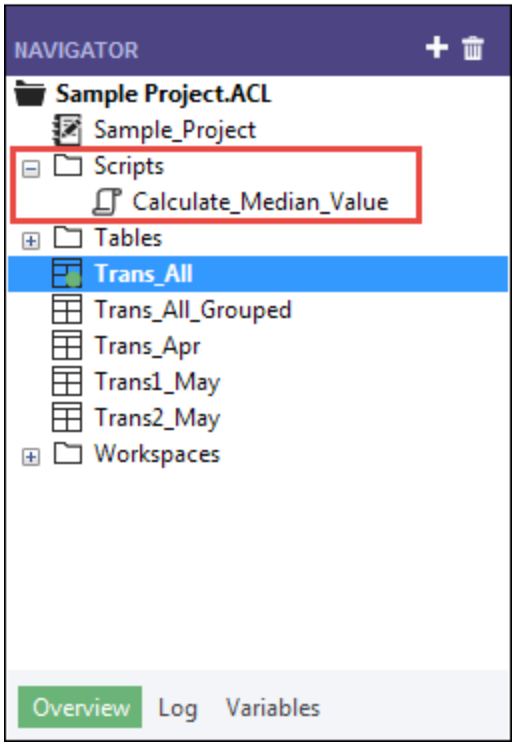
If the process is automated, maybe you can delegate the task to a more junior staff member. A tested script gives you the confidence that less experienced employees can perform the task

consistently and accurately, without a significant increase to their workload.

What is a script?

An Analytics script is a series of ACLScript commands that perform a particular task, or several related tasks. For example, everything that you just did manually in the first part of this tutorial could also be performed using a script.

ACLScript is the command language that forms the basis of Analytics. Scripts are stored in Analytics projects. Individual scripts appear in the **Navigator**, and are prefaced by the script icon .



How the Analytics command log works

You may have noticed that the **Navigator** contains the **Log** tab. As a script writer, you'll discover that the Analytics command log is your best friend.

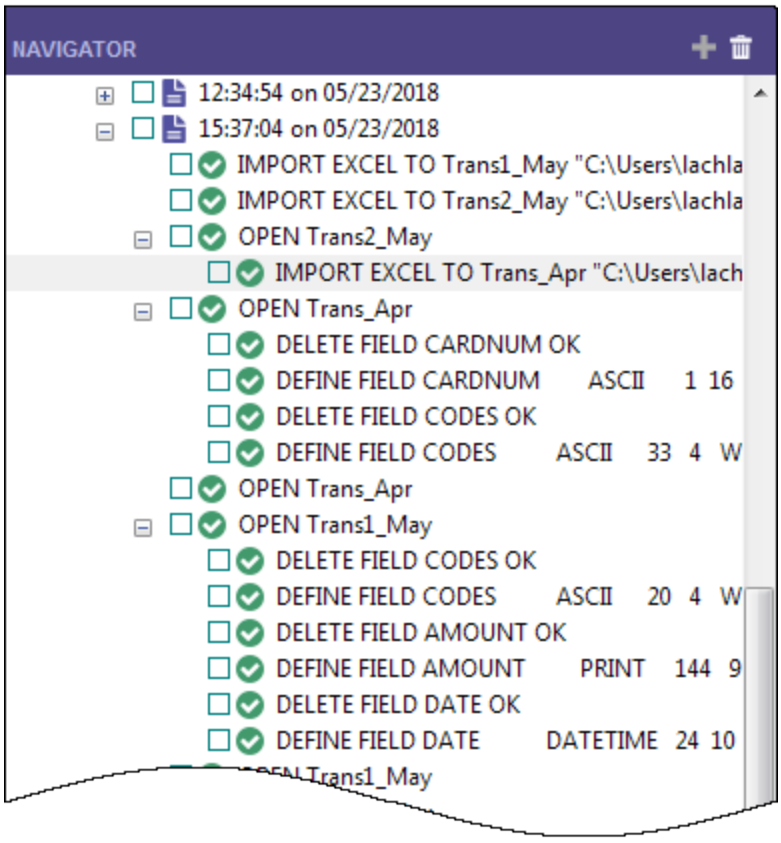
Steps

1. Click the **Log** tab to open it.

   You're looking at the Analytics command log. You can drag the **Navigator** panel wider to see more of the content.

   The log records the exact sequence of commands executed during each Analytics session, and saves them as part of the Analytics project.

If you've just finished the first part of this tutorial, the log contains a list of all the actions you've just performed in the user interface.



2. In the log, locate and click the SUMMARIZE command that outputs results to a new table.

The command prefills the **Command Line** near the top of the Analytics interface, just below the toolbar.

> **Note**
>
> If the **Command Line** isn't visible, select **Window > Command Line** from the Analytics main menu.

3. Open the **Trans_All** table, if it is not already open.
4. If the `f_Prohibited_codes` filter is applied, remove it.
5. Click in the **Command Line**, change `"Trans_All_Grouped.FIL"` to `"Trans_All_Grouped_2.FIL"`, and press Enter.

   The Summarize command is re-run on the **Trans_All** table and outputs the **Trans_All_Grouped_2** table, which replicates the first output table you created manually.

With a minimal amount of effort you re-performed all your earlier manual work required to summarize the **Trans_All** table. Running a command from the command line is like running a simple one-line script.

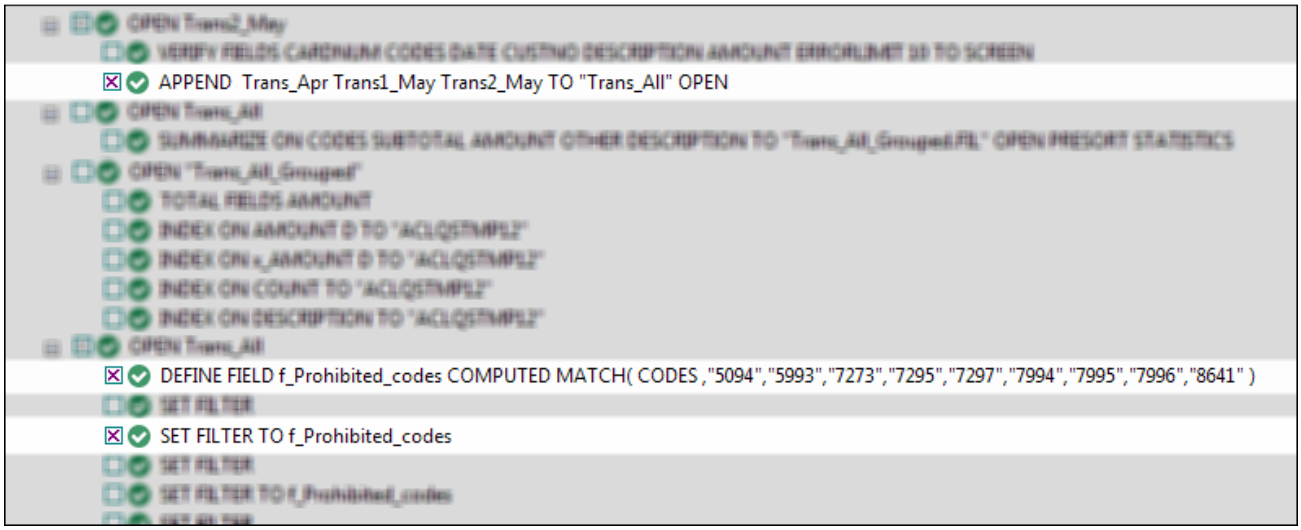
Building a script by copying commands from the log

You'll again reuse ACLScript syntax from the log, but this time you'll copy the syntax to an Analytics script. To keep things quick and easy, you'll script only a portion of the work you performed manually in the tutorial, but you could script all of it.

> **Note**
> We're going to skip over some scripting best practices in order to keep this introduction to scripting brief. The goal is to demonstrate how easy it is for even new users to create scripts in Analytics.

Steps

1. In the log, locate and select the following commands:



2. Right-click the log and select **Save Selected Items > Script**.
3. In the **Save Script As** dialog box, enter the script name `Append_and_filter` and click **OK**.
4. In the **Overview** tab of the **Navigator**, double-click the newly created **Append_and_filter** script to open it in the **Script Editor**.

   The script opens and contains the complete syntax of the three commands you selected in the log.

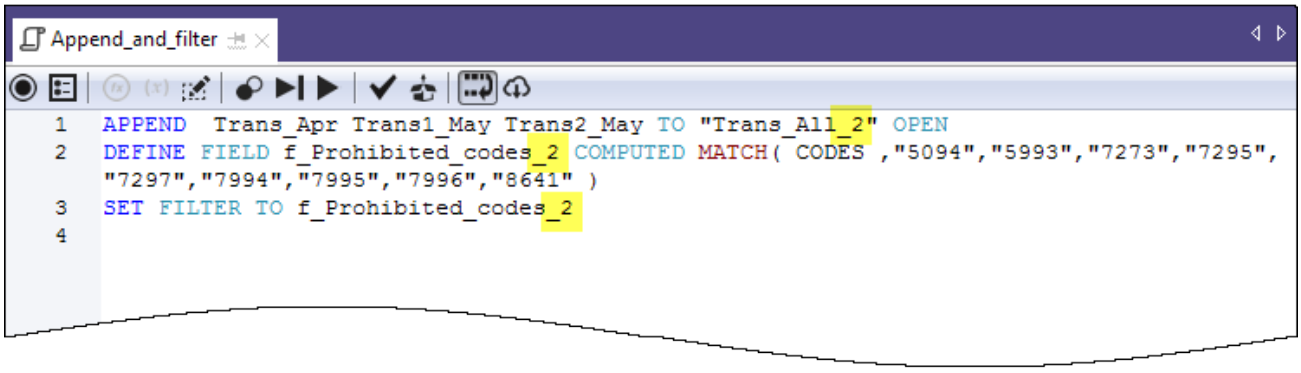
5. Take a moment to read the syntax for each command.

   Do you see how the actions you previously performed in the user interface correspond to individual pieces of ACLScript syntax? For example, after the `APPEND` command, there are the names of the three tables you appended:

`Trans_Apr Trans1_May Trans2_May`

For the most part, the correspondence between ACLScript syntax and actions in the user interface is relatively straightforward, which means the syntax is not that difficult to understand.

6. Modify the script by adding `_2` in the following locations:



You're adding `_2` to avoid name conflicts with the table and filter you already created manually.

7. On the **Script Editor** toolbar click **Run** to run the script.

   Click **Yes** to any prompts that appear.

   The script runs and performs the following tasks:

   - appends the three tables you imported from Excel into a single table, and opens the new table
   - creates the prohibited codes filter
   - applies the filter to the new table

   As you can see, running a script is much faster than performing the same actions manually. Imagine the time savings, and improved consistency, in a real-world situation with much more complex analysis performed on a weekly or monthly basis.

> **Note**
> You can also run a script by right-clicking it in the **Navigator** and selecting **Run**. A script does not have to be open to be run.

The entire tutorial in a script

The entire tutorial you just performed manually appears below in a script (in the "Steps" section). To finish this brief introduction to scripting, you're going to copy the script to Analytics and then redo the tutorial work, but this time with just a couple of clicks of the mouse.

> **Note**
>
> The script assumes that the `Sample Data Files` folder is installed in the default location. If the folder is installed in a different location, you need to modify the navigation paths in the script to point to the correct location.
>
> The tables created by the script are appended with `_s` so that they don't overwrite the tables you created manually.

Steps

Create a new, empty script

1. In the **Overview** tab in the **Navigator**, right-click the **Scripts** folder and select **New > Script**.
2. Right-click the **New_Script**, select **Rename**, type or copy `Getting_Started_tutorial`, and press **Enter**.

Copy and paste the tutorial script

1. Click **Show me the script** below.
2. Click and drag to select the entire script and then press **Ctrl+C** to copy the script.

   > **Note**
   >
   > It's important that you select the entire script and don't miss any lines. Alternately, you can download a text file with the script here: Getting started tutorial (non-Unicode edition)

3. Click in the Script Editor window and press **Ctrl+V** to paste the script syntax into the empty **Getting_Started_tutorial** script.

Update and save the script

1. Update the navigation paths in the script:
   a. Click the first line of the script.
   b. Right-click and select **Find**.
   c. Type the following entries in the **Replace** dialog box:
      - **Find what:** `user_account_name`
      - **Replace with:** *the actual account name on your computer*
   d. Perform the find-and-replace of all instances of `user_account_name`
2. Click **Save the Open Project**  , and click **Yes** in the prompt that appears.

   If you do not find the save icon, select **Window > Toolbar** in the Analytics main menu to enable the toolbar.

Run the script

On the **Script Editor** toolbar click **Run**  to run the script.

The script runs and replicates all the tutorial work. Interactive notifications provide key information as the script runs.

Show me the script

> **Note**
>
> If you haven't worked with scripts before, the script syntax may look overwhelming at first. Keep in mind that almost all the syntax was simply copied from the Analytics log.
>
> The syntax for the interactive notifications in the script (DIALOG commands) was auto-generated by another relatively simple Analytics tool.
>
> The green COMMENT commands walk you through the script at a high level. You'll recognize the tasks that you just completed in the preceding tutorial.

```
COMMENT
*** Non-Unicode Edition ***
This script performs all the actions that you performed manually in the "Get-
ting Started with ACL Analytics" tutorial.
END

COMMENT Allows overwriting of tables without a user confirmation.
SET SAFETY OFF

COMMENT Imports the three Excel worksheets.

IMPORT EXCEL TO Trans1_May_s "C:\Users\user_account_name\Documents\ACL
Data\Sample Data Files\Trans1_May_s.fil" FROM "Trans_May.xls" TABLE "Trans1_
May$" KEEPTITLE  FIELD "CARDNUM" C WID 19 AS "" FIELD "CODES" N WID 4 DEC 0 AS
"" FIELD "DATE" D WID 19 PIC "YYYY-MM-DD hh:mm:ss" AS "" FIELD "CUSTNO" C WID
6 AS "" FIELD "DESCRIPTION" C WID 95 AS "" FIELD "AMOUNT" C WID 9 AS ""

IMPORT EXCEL TO Trans2_May_s "C:\Users\user_account_name\Documents\ACL
Data\Sample Data Files\Trans2_May_s.fil" FROM "Trans_May.xls" TABLE "Trans2_
May$" KEEPTITLE  FIELD "CARDNUM" C WID 19 AS "" FIELD "CODES" N WID 4 DEC 0 AS
"" FIELD "DATE" D WID 19 PIC "YYYY-MM-DD hh:mm:ss" AS "" FIELD "CUSTNO" C WID
6 AS "" FIELD "DESCRIPTION" C WID 155 AS "" FIELD "AMOUNT" C WID 9 AS ""

IMPORT EXCEL TO Trans_Apr_s "C:\Users\user_account_name\Documents\ACL
Data\Sample Data Files\Trans_Apr_s.fil" FROM "Trans_April.xls" TABLE "Trans_
Apr$" KEEPTITLE FIELD "CARDNUM" N WID 16 DEC 0 AS "" FIELD "AMOUNT" N WID 6
DEC 2 AS "" FIELD "DATE" D WID 10 PIC "YYYY-MM-DD" AS "" FIELD "CODES" N WID 4
DEC 0 AS "" FIELD "CUSTNO" C WID 6 AS "" FIELD "DESCRIPTION" C WID 45 AS ""
```

```
COMMENT Adjusts the table layouts of the three new Analytics tables.

OPEN Trans_Apr_s
DELETE FIELD CARDNUM OK
DEFINE FIELD CARDNUM ASCII 1 16 WIDTH 19
DELETE FIELD CODES OK
DEFINE FIELD CODES ASCII 33 4 WIDTH 7

OPEN Trans1_May_s
DELETE FIELD CODES OK
DEFINE FIELD CODES ASCII 20 4 WIDTH 7
DELETE FIELD AMOUNT OK
DEFINE FIELD AMOUNT PRINT 144 9 2 WIDTH 9
DELETE FIELD DATE OK
DEFINE FIELD DATE DATETIME 24 10 PICTURE "YYYY-MM-DD" WIDTH 27

OPEN Trans2_May_s
DELETE FIELD CODES OK
DEFINE FIELD CODES ASCII 20 4 WIDTH 7
DELETE FIELD AMOUNT OK
DEFINE FIELD AMOUNT PRINT 204 9 2 WIDTH 9
DELETE FIELD DATE OK
DEFINE FIELD DATE DATETIME 24 10 PICTURE "YYYY-MM-DD" WIDTH 27

COMMENT Verifies the imported data and provides user notifications.

OPEN Trans_Apr_s
VERIFY FIELDS CARDNUM AMOUNT DATE CODES CUSTNO DESCRIPTION ERRORLIMIT 10
IF WRITE1=0 DIALOG (DIALOG TITLE "User Dialog" WIDTH 490 HEIGHT 100 )
(BUTTONSET TITLE "&OK;&Cancel" AT 360 12 DEFAULT 1 ) (TEXT TITLE "Trans_Apr_s
table:  0 data validity errors detected" AT 12 28 )
IF WRITE1>0 DIALOG (DIALOG TITLE "User Dialog" WIDTH 490 HEIGHT 100 )
(BUTTONSET TITLE "&OK;&Cancel" AT 360 12 DEFAULT 1 ) (TEXT TITLE "Trans_Apr_s
table:  %WRITE1% data validity errors detected" AT 12 28 )

OPEN Trans1_May_s
VERIFY FIELDS CARDNUM CODES DATE CUSTNO DESCRIPTION AMOUNT ERRORLIMIT 10
IF WRITE1=0 DIALOG (DIALOG TITLE "User Dialog" WIDTH 490 HEIGHT 100 )
(BUTTONSET TITLE "&OK;&Cancel" AT 360 12 DEFAULT 1 ) (TEXT TITLE "Trans1_May_s
table:  0 data validity errors detected" AT 12 28 )
IF WRITE1>0 DIALOG (DIALOG TITLE "User Dialog" WIDTH 490 HEIGHT 100 )
(BUTTONSET TITLE "&OK;&Cancel" AT 360 12 DEFAULT 1 ) (TEXT TITLE "Trans1_May_s
table:  %WRITE1% data validity errors detected" AT 12 28 )

OPEN Trans2_May_s
VERIFY FIELDS CARDNUM CODES DATE CUSTNO DESCRIPTION AMOUNT ERRORLIMIT 10
IF WRITE1=0 DIALOG (DIALOG TITLE "User Dialog" WIDTH 490 HEIGHT 100 )
```

```
(BUTTONSET TITLE "&OK;&Cancel" AT 360 12 DEFAULT 1 ) (TEXT TITLE "Trans2_May_s
table:  0 data validity errors detected" AT 12 28 )
IF WRITE1>0 DIALOG (DIALOG TITLE "User Dialog" WIDTH 490 HEIGHT 100 )
(BUTTONSET TITLE "&OK;&Cancel" AT 360 12 DEFAULT 1 ) (TEXT TITLE "Trans2_May_s
table:  %WRITE1% data validity errors detected" AT 12 28 )

COMMENT Verifies the Badfile table and provides a user notification.
OPEN Badfile
VERIFY FIELDS InvoiceNo Prodno Price OrderQty ShipQty Total ERRORLIMIT 10
IF WRITE1=0 DIALOG (DIALOG TITLE "User Dialog" WIDTH 490 HEIGHT 100 )
(BUTTONSET TITLE "&OK;&Cancel" AT 360 12 DEFAULT 1 ) (TEXT TITLE "Badfile
table:  0 data validity errors detected" AT 12 28 )
IF WRITE1>0 DIALOG (DIALOG TITLE "User Dialog" WIDTH 490 HEIGHT 100 )
(BUTTONSET TITLE "&OK;&Cancel" AT 360 12 DEFAULT 1 ) (TEXT TITLE "Badfile
table:  %WRITE1% data validity errors detected" AT 12 28 )
CLOSE

COMMENT Appends the three new Analytics tables into a single combined table.
APPEND Trans_Apr_s Trans1_May_s Trans2_May_s TO "Trans_All_s" OPEN
DIALOG (DIALOG TITLE "User Dialog" WIDTH 630 HEIGHT 100 ) (BUTTONSET TITLE
"&OK;&Cancel" AT 500 12 DEFAULT 1 ) (TEXT TITLE "The combined transactions
table (Trans_All_s) contains %WRITE1% records" AT 12 28 )

COMMENT Groups the combined table by merchant category code.
SUMMARIZE ON CODES SUBTOTAL AMOUNT OTHER DESCRIPTION TO "Trans_All_Grouped_
s.FIL" OPEN PRESORT STATISTICS
DIALOG (DIALOG TITLE "User Dialog" WIDTH 700 HEIGHT 100 ) (BUTTONSET TITLE
"&OK;&Cancel" AT 570 12 DEFAULT 1 ) (TEXT TITLE "The grouped transactions
table (Trans_All_Grouped_s) contains %WRITE1% merchant category codes" AT 12
28 WIDTH 550 )

COMMENT Filters the combined table to show only prohibited transactions.
OPEN Trans_All_s
DEFINE FIELD f_Prohibited_codes COMPUTED MATCH(CODES, "5094", "5993", "7273",
"7295", "7297", "7994", "7995", "7996", "8641")
SET FILTER TO f_Prohibited_codes

COMMENT Successful completion message.
DIALOG (DIALOG TITLE "User Dialog" WIDTH 490 HEIGHT 100 ) (BUTTONSET TITLE
"&OK;&Cancel" AT 360 12 DEFAULT 1 ) (TEXT TITLE "The script successfully com-
pleted" AT 12 28 )

COMMENT A user confirmation is required before overwriting a table.
SET SAFETY ON
```

You're finished

That's the end of this brief introduction to scripting. We hope you've seen enough to be convinced of the value of scripting and that you want to learn more.

Where to next?

You have several options for learning more about scripting in Analytics:

Option	Useful information
Tutorials	The Analytics Help contains the following beginner-level tutorials: ◦ "Scripting for complete beginners" on page 1379 ◦ "Analytics scripting basics" on page 1397 ◦ "How to use functions" on page 1424 The Help also contains a complete ACLScript language reference with detailed information about every Analytics command and function.
Academy	Academy offers both an introductory and an advanced scripting course: ◦ Introduction to scripting in ACL Analytics (ACL 106) ◦ ACL Analytics Scripting (ACL 303) Academy is the Galvanize online training resource center. Go to the course catalog to see the available courses. Academy courses are included at no extra cost for any user with an ACL subscription.
Community	Community is a web-based platform with a variety of customer resources, including a customer forum where Analytics scripting is frequently discussed in depth.

Getting started with Analytics (Unicode edition)

This getting started tutorial introduces you to the end-to-end process of analyzing data using Analytics.

Estimated time	60 minutes
Requirements	No previous Analytics experience is required. Some basic data analysis experience is assumed, but not absolutely critical.
Analytics version	13.0 or later (Unicode edition)
Do the right version of the tutorial	Do this version of the tutorial if you're using the Unicode edition of Analytics. If you're using the non-Unicode edition, do "Getting started with Analytics (non-Unicode edition)" on page 35.

> **Tip**
> To find out which edition of Analytics you're using, on the Analytics main menu, click **Help > About** to open the **Analytics** dialog box. The edition designation appears after the version number.

> **Note**
> The Chinese and Japanese user interfaces are Unicode-only.

Scenario

Review corporate credit card transactions

You're asked to review corporate credit card transactions from a two-month period. Your goal is to get a general picture of how employees used cards during the period, and also to identify any possible misuse of cards.

The transaction data is contained in three separate Excel worksheets. Before you can analyze the data, you need to import it into Analytics, and combine the separate data sets into a single Analytics table.

After you've analyzed the data, you want to present the results of your analysis visually, to better engage your audience.

Optional section

You're told that from now on, reviewing corporate credit card transactions will be a recurring responsibility.

To allow yourself, or someone else, to perform future reviews quickly and accurately, you decide to create a script to automate some of the work.

PIPAR – the data analysis cycle in Analytics

The data analysis cycle in Analytics contains five stages, which are summarized by the acronym **PIPAR**:

Plan, **I**mport, **P**repare, **A**nalyze, **R**eport

Plan your work

Planning your data analysis work is important, and often critical. If you skip the planning stage, and jump straight into running analytical commands against data, you may run into problems, create extra work for yourself, or even miss important analytical insights.

Even a basic plan is better than no plan. With experience, and increasing knowledge of Analytics, your planning will become more fully developed and more precise. Good planning is the key to data analysis projects that progress smoothly and efficiently.

Planning guidelines

Develop clear, specific objectives

What is the intended end product of your analysis?

You need clearly defined objectives in order to be able to plan how to achieve them. For example, in this tutorial, your specific objectives are:

- identify the count, and total amount, of corporate credit card transactions in each merchant category
- identify any transactions in prohibited categories

Map a step-by-step approach

How will you achieve your objectives?

Accomplishing an objective often requires more than one step, so map a detailed, step-by-step approach to guide you along the way.

For example, two of the steps in the planning for this tutorial could be:

- combine all the individual transaction files into a single file
- group the combined transaction data into merchant categories

Once you've broken down the larger objectives into individual steps, you can consider which Analytics features and functions to use to perform each step.

Identify what data you'll need

What data do you need to achieve your objectives?

Itemize the required source data to the level of specific data elements or fields. You won't be able to achieve your desired output without the appropriate input.

In this tutorial you have the main transaction files, but to achieve your second objective you'll also need a list of prohibited merchant category codes.

Consider technical requirements

Are there any technical considerations you must take into account?

Regardless of which tool you're using for data analysis, you must work within its constraints. Is the source data stored in a location or a system that the tool can access, and in a format that it can read? Is the analysis you're proposing supported by the tool?

For example, in order to combine multiple tables in Analytics, the data types of the corresponding fields in each table must be the same. Analytics supports changing the data type of a field, but that's a step you need to account for in your planning.

Be prepared to iterate

You may need to adjust your plan as you go along.

In the course of your analysis, you discover something unexpected that warrants further investigation. You realize you need additional data and additional analytical steps.

Your plan can evolve as your understanding of the data evolves. And it can serve as the basis for a more mature plan for future analysis of a similar nature.

Import data

You must import data into Analytics before you can analyze it.

We'll familiarize with the import process by using the **Data Definition Wizard** to import three Excel worksheets. Importing from Excel is one of the most common methods for acquiring data for analysis in Analytics. However, Analytics supports importing data from a wide variety of data sources.

Open Analytics and "Sample Project.ACL"

> **Note**
> The steps below assume you have already activated Analytics.

Steps

1. Double-click the **ACL for Windows** shortcut on your desktop.
2. In the **ACL for Windows** screen, under **Open**, click **Analytic Project**.
3. Navigate to `C:\Users\user_account_name\Documents\ACL Data\Sample Data Files` and double-click `Sample Project.ACL`.

   `Sample Project.ACL` opens in Analytics.

   If you did not install the `Sample Data Files` folder in the default location when you installed Analytics, navigate to the location where you installed it.

Import the first two Excel worksheets

You will get started by importing two Excel worksheets at the same time. Importing multiple Excel worksheets simultaneously is a great way to reduce labor.

Steps

1. From the Analytics main menu, select **Import > File**.
2. In the **Select File to Define** dialog box, locate and select `Trans_May.xls` and click **Open**.

   The Excel file is in the same folder as `Sample Project.ACL`.

3. In the **File Format** page, make sure the **Excel file** option is selected and click **Next**.
4. In the **Data Source** page, select both worksheets in the file:
   - `Trans1_May$`
   - `Trans2_May$`
5. Make sure **Use first row as Field Names** is selected, click **Next**, and then click **Finish**.

   The two Excel worksheets are imported into two separate Analytics tables.

Import the third Excel worksheet

Now import the third Excel worksheet by itself. When you import a single worksheet, you have the option of manually adjusting some of the metadata settings during the import process, rather than doing it later in Analytics.

Steps

1. Repeat the steps in the previous procedure to locate and select `Trans_April.xls`.
2. In the **File Format** page, make sure the **Excel file** option is selected and click **Next**.

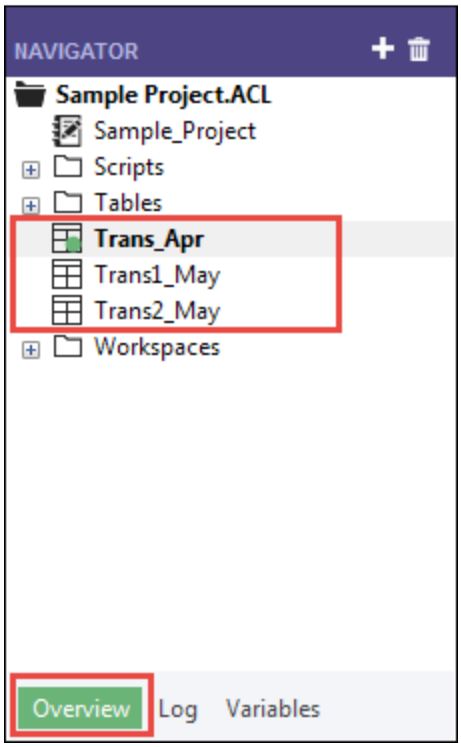
3. In the **Data Source** page, select `Trans_Apr$`.
4. Make sure **Use first row as Field Names** is selected, and click **Next**.
5. In the **Excel Import** page, click the header to select the **TRANS_DATE** column and make the following changes:
   - In the **Name** field, change `TRANS_DATE` to `DATE`.
   - In the **Length** field, change `19` to `10`.

   > **Note**
   > You're making adjustments to a data field in the Data Definition Wizard, during the import process. You can also make adjustments later, after you have completed importing the data. You'll see the reason for the adjustments in the next section of the tutorial.

6. Click **Next**, in the **File name** field type `Trans_Apr`, and click **Save**.
7. Click **Finish**, and then click **OK**.

   The third Excel worksheet is imported into an Analytics table.

You should now have three new Analytics tables in the **Overview** tab of the **Navigator**. These tables contain read-only copies of the Excel data. They do not contain the Excel source data itself.



Prepare data

Often you must perform one or more data preparation tasks before data is ready to analyze.

For this tutorial, you'll perform two preparation tasks:

- make additional adjustments to harmonize data fields
- combine the three new Analytics tables into a single table for analysis

As well, as a best practice, you should always verify the validity of imported data before performing analytical work. Even a small amount of invalid data in a table can invalidate all your subsequent data analysis.

Why do I need to prepare data?

You're eager to get on with the analysis of the data, but without proper data preparation you may not be able to perform the analysis. Or the analysis you perform may be flawed.

A wide variety of issues can affect source data making it unsuitable for analysis without some initial preparation.

For example:

- The source data is spread between several different files and needs to be consolidated so that it can be analyzed as a single set of data.
- Corresponding fields in different files need to be "harmonized", which means making them identical in structure and format as a prerequisite to processing them.
- "Dirty data" needs to be cleansed and standardized, which you can do with Analytics functions.

Key point

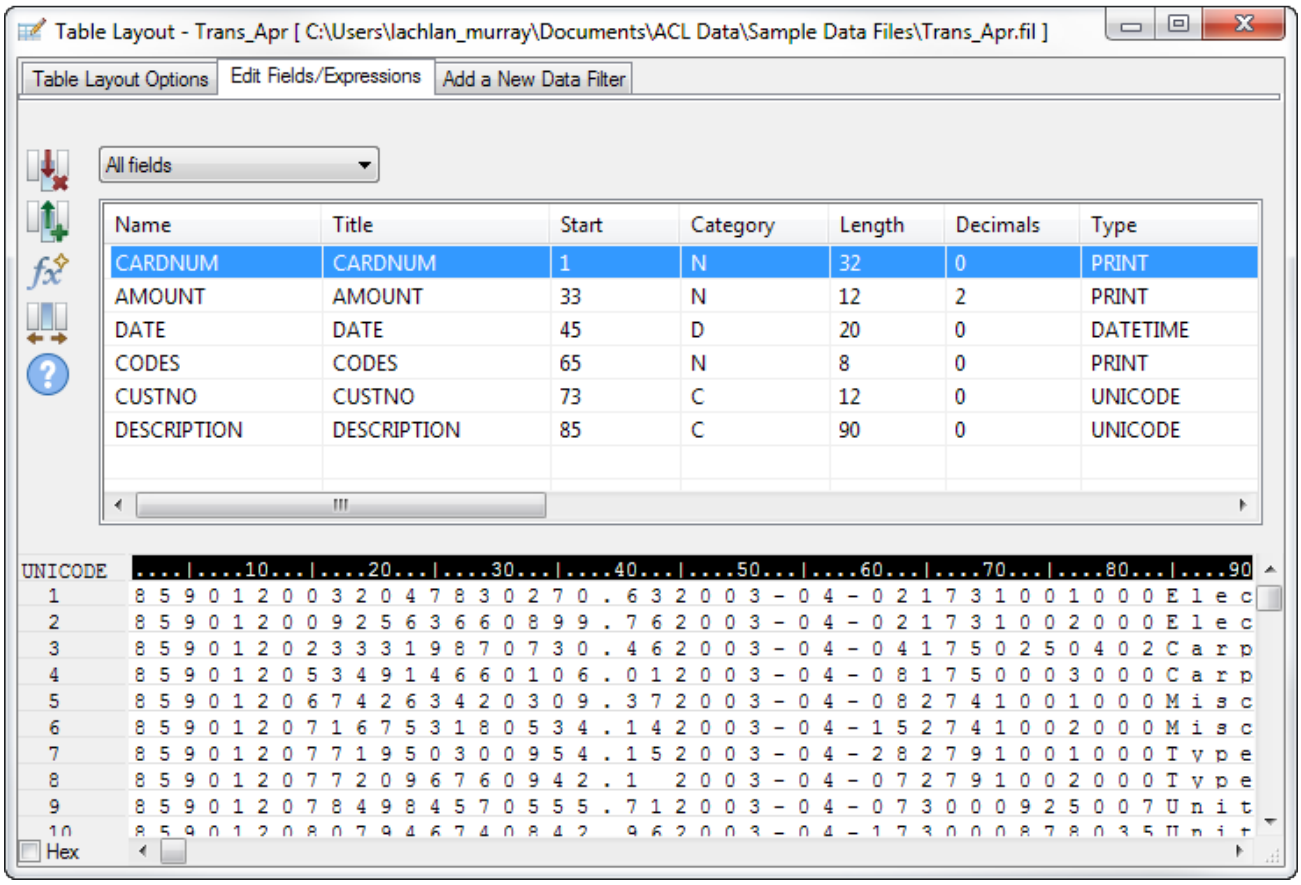
The time you spend importing and preparing data may exceed the time you spend on the actual analysis. However, they are critical initial stages, and provide the foundation that your analysis is built on.

Adjust the table layouts

Every table in an Analytics project has a **table layout**. The table layout contains metadata such as field names, the start position of fields, the length of fields, the data type of fields, and so on.

Before we can combine the three new Analytics tables into a single table, we need to harmonize some of the metadata in the table layouts.

Here's what the **Trans_Apr** table layout looks like. You'll quickly learn your way around table layouts as you become more familiar with Analytics. You can do a lot of useful things in the table layout.



Adjust the Trans_Apr table layout

First, you need to change the data type of two fields in the **Trans_Apr** table.

Steps

1. Open the **Trans_Apr** table, if it is not already open.

   To open a table, double-click it in the **Navigator**.

2. Above the table view, click **Edit Table Layout**.
3. Double-click the **CARDNUM** field to open the field definition for editing.
4. Under **Valid Data Types**, double-click **UNICODE** to update the data type of the field.

   In the other two tables, the **CARDNUM** field has an UNICODE data type. For the most part, combining data requires that corresponding fields in the tables being combined have the same data type.

5. Click **Accept Entry**.

   If a prompt appears, click **Yes** to save your changes.

6. Double-click the **CODES** field and change the data type to **UNICODE**.
7. Click **Accept Entry**, and then click Close to exit the **Table Layout** dialog box.

Adjust the Trans_May table layouts

To finish the adjustments, you need to change the data type of two fields in both the **Trans1_May** and the **Trans2_May** layouts. You may also need to make an adjustment to the **DATE** field.

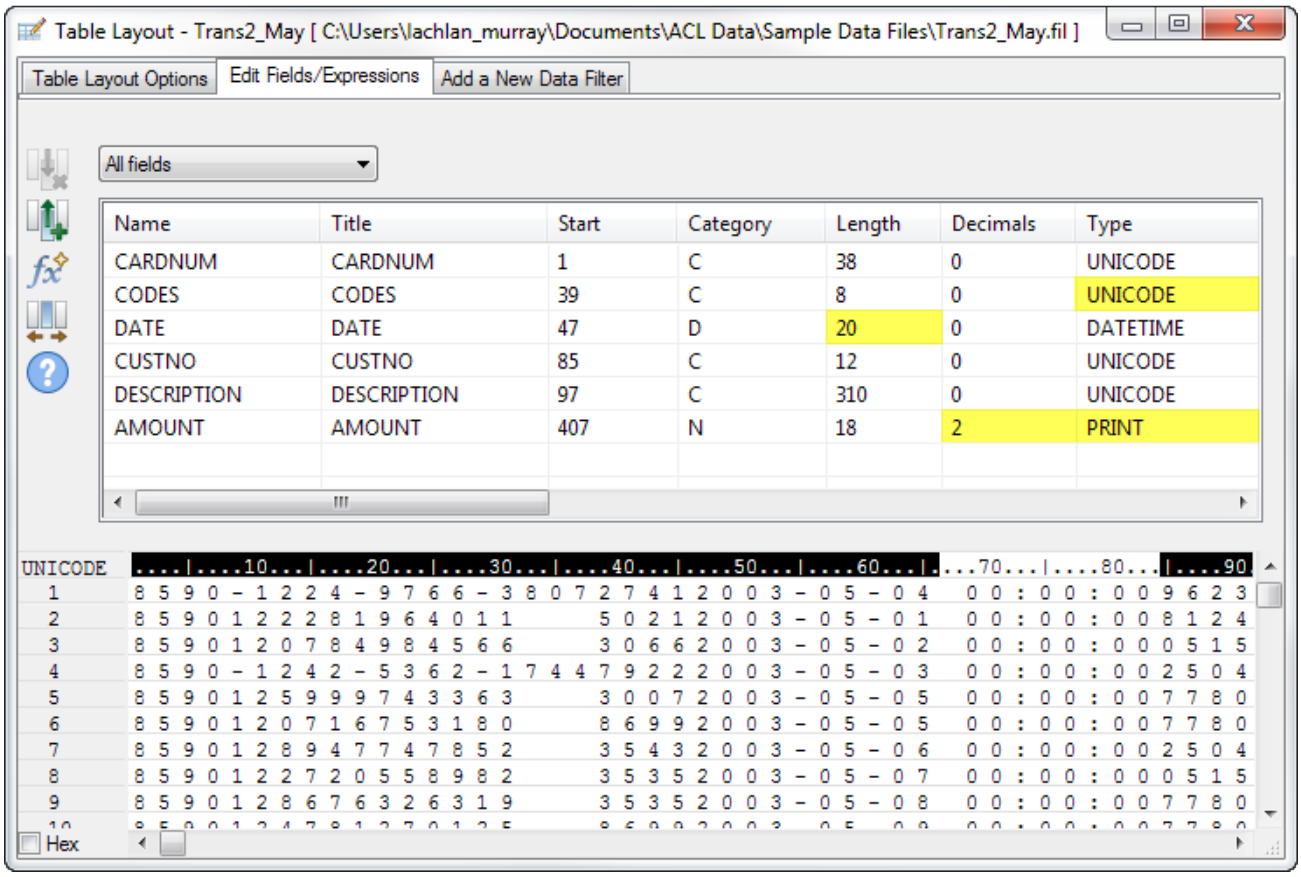
Steps

Follow the process above to make the following changes in both the **Trans1_May** and the **Trans2_May** layouts:

Field	Change data type to:	Additional change
CODES	UNICODE	
AMOUNT	PRINT	Enter `2` in the **Dec.** field to specify that numeric values display two decimal places.
DATE	no change	**Note** If the DATE field already has a length of 20, no adjustment is required. ◦ In the **Len.** field, change `38` to `20`. This change omits the empty time data. ◦ In the **Format** dropdown list, select **YYYY-MM-DD**.

When you're finished, the May table layouts should look like the layout below.

> **Note**
> The date format (YYYY-MM-DD) isn't shown in the layout summary. The DESCRIPTION field length is different in the two May layouts.



Verify the imported data

Now let's verify the data in the three imported tables to make sure it's safe to proceed with additional data preparation, and data analysis.

> **Note**
>
> We're verifying the data **after** updating the data types. When you verify data in Analytics, you're checking that all the values in a field conform to the requirements of the field's data type. So it makes sense to verify data only once the data types are finalized.

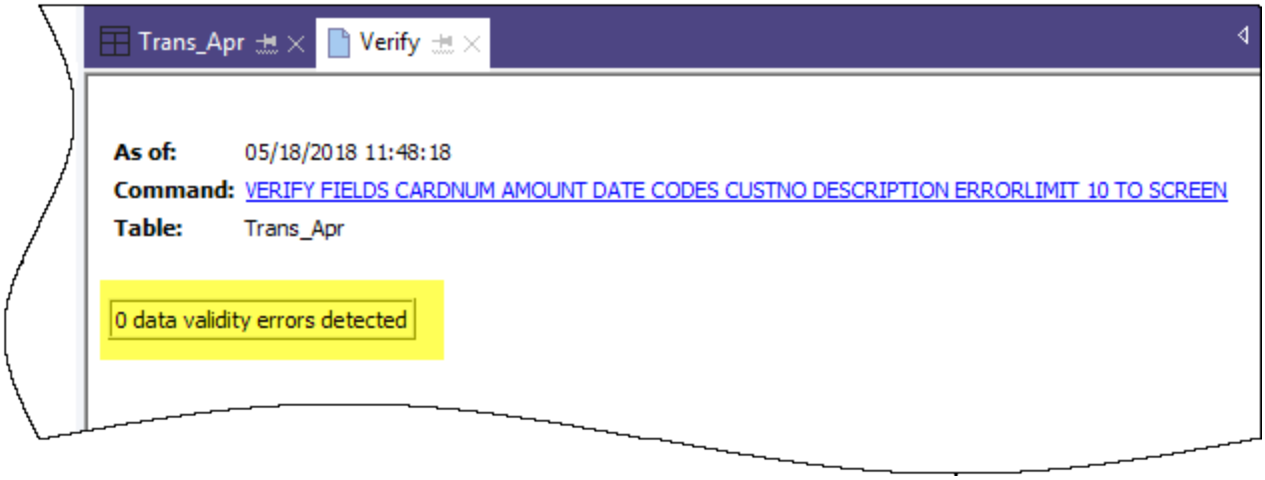
Steps

1. Open the **Trans_Apr** table.
2. From the Analytics main menu, select **Data > Verify**.
3. In the **Verify** dialog box, select all the fields in the field list.

> **Tip**
>
> Use **Shift+click** to select multiple adjacent fields.

4. Click **OK**.

   The result should be: **0 data validity errors detected**.



Learn more

Did you notice that Analytics automatically translated the action you performed in the user interface into the ACLScript `VERIFY` command? Every command-level action you perform in the user interface is automatically translated into its corresponding ACLScript command, and captured and stored in the command log that accompanies each Analytics project.

This automatic generation of valid, runnable script syntax is one of the most powerful features in Analytics. We'll be looking at scripting in an optional section at the end of the tutorial.

5. In the **Navigator**, double-click the **Trans1_May** table to open it, and repeat the steps to verify the data.
6. Do the same for the **Trans2_May** table.

   Both tables should not contain any data validity errors.

> **Note**
>
> If you get an error message stating **Maximum error limit reached**, check that you correctly changed the format of the **Date** field in the table layout to **YYYY-MM-DD**.

Learn more

If you want to see what happens when Analytics does identify data validity errors, open **Tables\Badfile** and run the verification process.

Combine the three Analytics tables

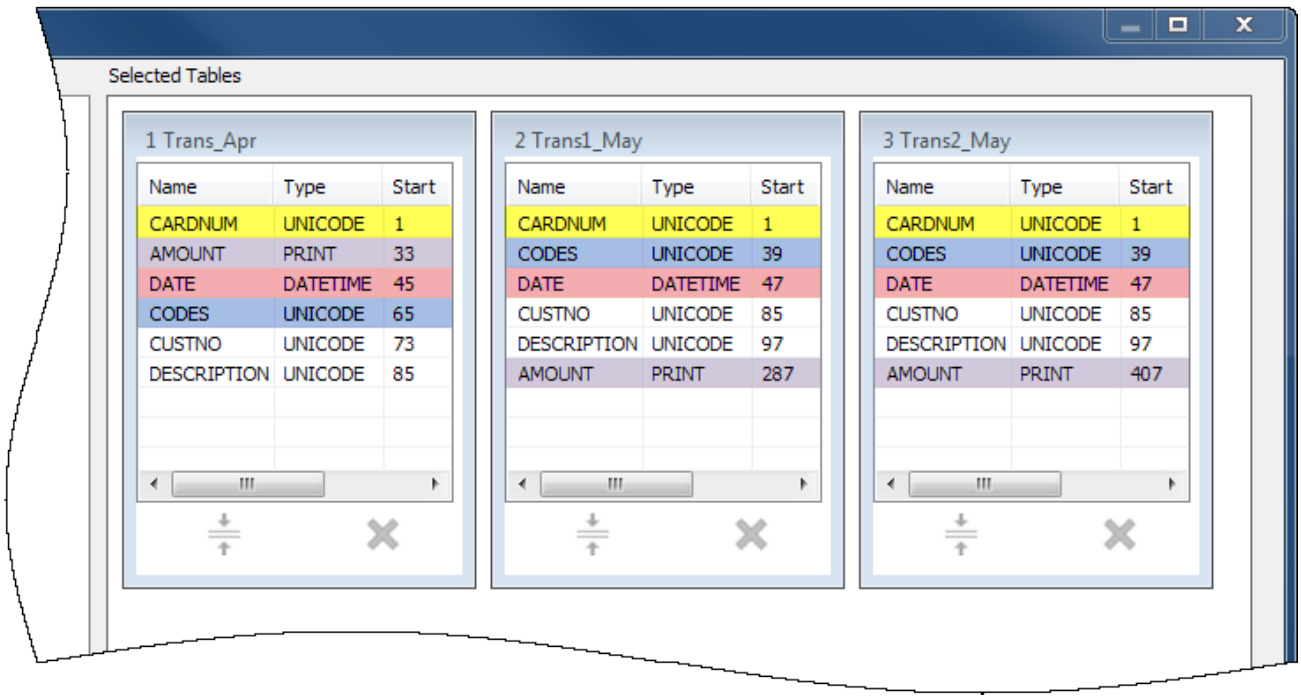
For the final data preparation task, you combine the three new Analytics tables into a single table.

For simplicity, the tutorial combines only three tables. However, you could use the same process to combine 12 monthly tables into a single annual table and perform analysis on data for an entire fiscal year.

Steps

1. From the Analytics main menu, select **Data > Append**.
2. Under **Available Tables**, double-click each of the new tables to add it to the **Selected Tables** area.
3. Take a look at the fields in the three tables and notice how the names and data types are identical based on the changes you made in the Data Definition Wizard and the **Table Layout** dialog box.

   To append corresponding fields, their names must be identical, and in most situations their data types must be identical.



4. Select **Use Output Table** so that the output table with the combined data opens automatically after you run the command.
5. In the **To** field, type `Trans_All` and click **OK**.

6. Click **Yes** in the notification that pops up.

> **Note**
> Don't worry about the notification. The append command performs some automatic harmonization of numeric fields, which saves you time and effort.

The new **Trans_All** table is created, and contains all the records from the three input tables. The record count in the status bar at the bottom of the Analytics interface should say **Records: 481**.

You're now ready to move on to some actual data analysis.

Analyze data

You perform analysis in Analytics by using commands and other tools to gain general insights about the data you are investigating, and to answer specific questions.

> **Note**
> The analysis stage is where the strength of your earlier planning becomes apparent. If you've formulated clear objectives regarding your investigation, you'll have a clearer idea of the types of analysis to perform.

The data analysis

For this tutorial, you'll perform the following analysis of the data in the **Trans_All** table:

- group the credit card transaction records by merchant category code in order to discover:
  - how employees are using corporate credit cards
  - how much money is being spent in each category
- create a filter to isolate any prohibited transactions

Group credit card transactions by merchant category code

Grouping or summarizing a set of data is an excellent way of quickly getting an overview of the data.

Steps

1. Open the **Trans_All** table, if it is not already open.
2. From the Analytics main menu, select **Analyze > Summarize**.

3. In the **Summarize** dialog box, select the following fields and options:

Tab	Field or option	Select or type
Main	Summarize On	select CODES
	Other Fields	select DESCRIPTION
	Subtotal Fields	select AMOUNT
	Avg, min, max	select the checkbox
Output	To	select File
	Name	type `Trans_All_Grouped`

4. Click **OK**.

   The new **Trans_All_Grouped** table is created. The table contains 110 records, one for each unique merchant category code in the **Trans_All** table. The **COUNT** field tells you how many source records are in each group.

> **Tip**
>
> Right-click the table view and select **Resize All Columns** to make the view more compact.

Simple tools for investigation

Now that you have a summarized version of the data, you can use some basic Analytics tools to gain general insight into corporate credit card use.

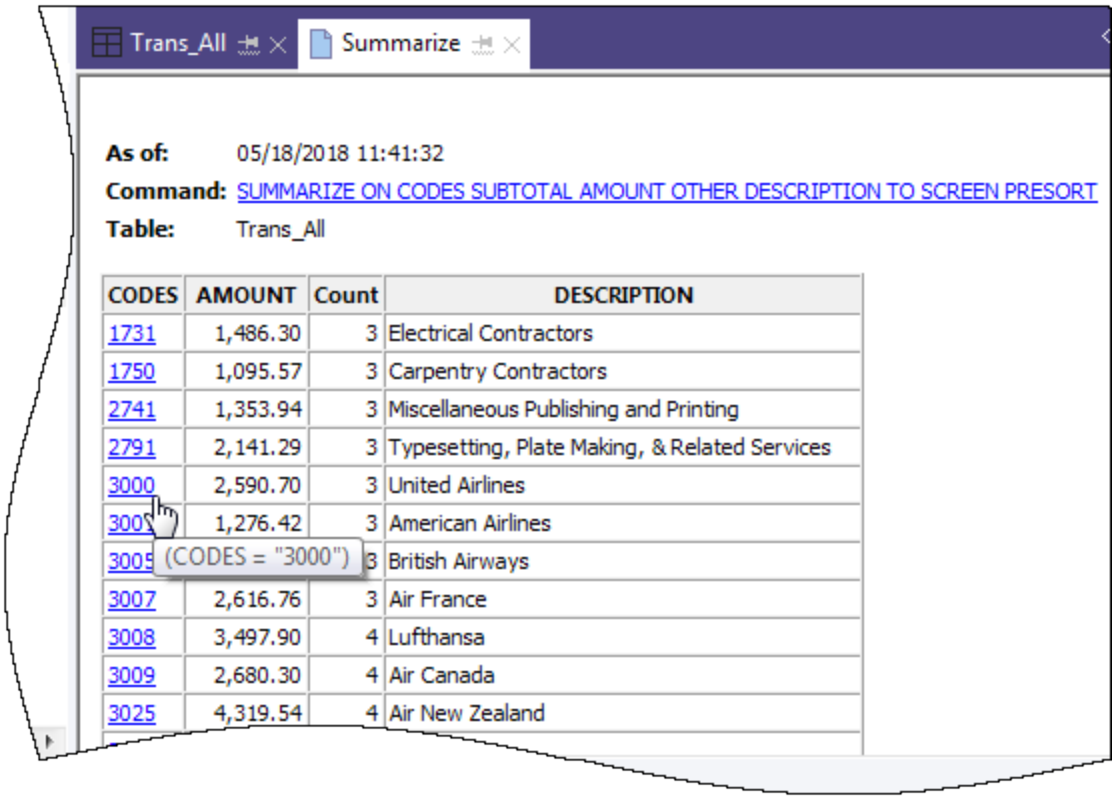
You can learn a lot about patterns of use, and possible misuse, in just a few clicks.

To gain this insight:	Do this in the Trans_All_Grouped table:
What was the total amount charged by employees during April and May?	◦ Select the **Total AMOUNT** header. ◦ Select **Analyze > Total**. Total expenditure was $187,177.13.
Where did employees spend the most money?	◦ Right-click the **Total AMOUNT** header and select **Quick Sort Descending** The **Description** field shows you that the most money was spent on: ◦ Caterers ◦ Eating places and Restaurants ◦ Hilton International
What were the largest single expenditures?	◦ Right-click the **Maximum AMOUNT** header and select **Quick Sort Descending** The **Description** and **Maximum AMOUNT** fields show you that the largest single expenditure was a Club Med amount of $1999.06. Is Club Med an authorized merchant code for the corporate credit card? If the credit card limit is $2000, was an employee charging an amount just under the limit?

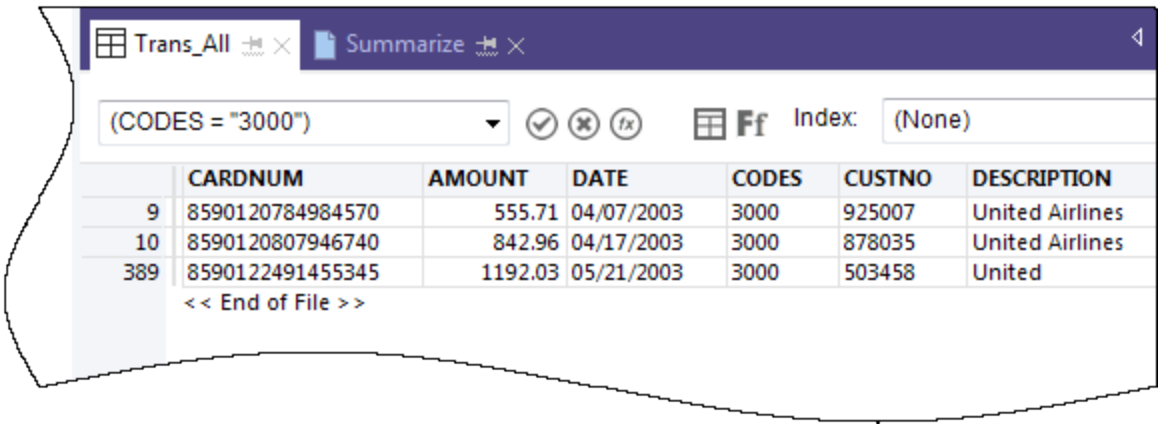
To gain this insight:	Do this in the Trans_All_Grouped table:
What does an examination of infrequently used codes reveal?	◦ Right-click the **COUNT** header and select **Quick Sort Ascending** Five categories had only a single charge each. Are some of them prohibited categories? Perhaps one or more employees thought that misusing a company card only very occasionally would allow them to escape detection. ◦ Cigar Stores & Stands ◦ Dating & Escort Svcs. ◦ Babysitting services ◦ Amusement Parks ◦ Civic, Fraternal, and Social Associations
Are any of the categories prohibited?	◦ Right-click the **DESCRIPTION** header and select **Quick Sort Ascending** to alphabetize the field values for easier scanning ◦ Scan down the field looking for suspicious categories Perhaps one or more of these categories are prohibited? ◦ Babysitting services ◦ Betting (including Lottery Tickets, Casino) ◦ Civic, Fraternal, and Social Associations ◦ Dating & Escort Svcs. ◦ Massage Parlors ◦ Precious Stones and Metals, Watches and Jewel ◦ Video Game Arcades/Establishments **Note** Manual scanning is impractical for all but small data sets. We'll look at a more practical, more reliable method next.

Learn more

Perhaps you just want to perform some quick analysis and you don't want to output the results to a new table. When you summarized the **Trans_All** table, instead of selecting **File** in the **Summarize** dialog box, you could select **Screen**, and output the results to the Analytics display area.



Outputting to screen is only practical for smaller data sets. However, it has the advantage of providing an easy way to drill-down on individual groups and see only the source records in each group.



Create a filter to isolate prohibited transactions

Filters allow you to isolate just the records you are interested in at a particular moment. Filters are a powerful tool for answering specific questions about data.

A general review of the corporate credit card transactions alerted you to some possible prohibited transactions. You decide to confirm whether any transactions are prohibited by matching a list of prohibited merchant category codes against the data.

Steps

Create the filter expression

1. Open the **Trans_All** table.
2. Click **Edit View Filter** at the top of the table view to open the **Expression Builder**.

   The **Expression Builder** is an Analytics component that lets you use the mouse to create expressions, rather than typing expression syntax manually. Expressions are combinations of values and operators that perform a calculation and return a result.
3. In the **Functions** drop-down list, select **Logical**, and then double-click the MATCH function to add it to the **Expression** text box.

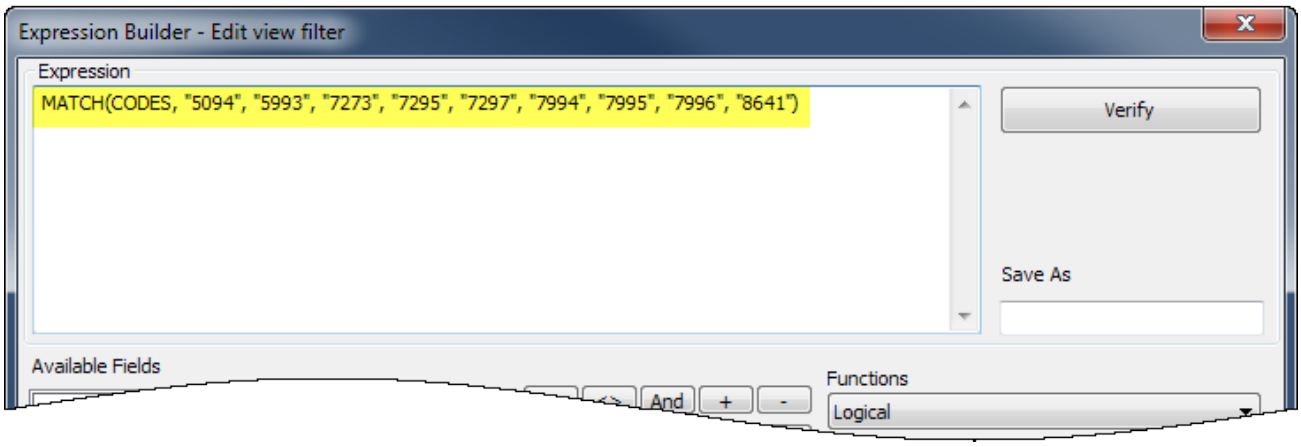
   You're going to use MATCH to isolate several prohibited merchant category codes in the **CODES** field.
4. In the **Expression** text box, highlight the `comparison_value` placeholder, and then in the **Available Fields** list, double-click **CODES**.

   The `CODES` field replaces `comparison_value`.
5. Copy the string of prohibited codes below and use them to replace the `test1 , test2 <,test3...>` placeholder:

   `"5094", "5993", "7273", "7295", "7297", "7994", "7995", "7996", "8641"`

   > **Note**
   > Make sure you copy the entire string, including all quotation marks.

   Your expression should look like this:



Verify the expression and save and apply the filter

1. Click **Verify** to test that the syntax of your expression is valid.

   Verifying expressions as soon as you create them is a best practice because it can help avoid more time-consuming troubleshooting later.

   If you get an error message, double-check that the syntax of the expression exactly matches the syntax shown above.

2. In the **Save As** field, type or copy the filter name `f_Prohibited_codes`.

   Galvanize recommends that you preface the names of saved filters with `f_`

3. Click **OK**.

   The **f_Prohibited_codes** filter is applied to the **Trans_All** table. Transactions that use a prohibited merchant category code are now isolated and plain to see. Consider a table with tens of thousands of records, or more, and the value of filters quickly becomes apparent.

Remove or reapply the filter

Try removing and reapplying the filter:

1. To remove the filter, click **Remove Filter**  .
2. To reapply the filter, do either of the following:
   - Select the filter name from the Filter history drop-down list at the top of the view.
   - Click **Edit View Filter**  to open the **Expression Builder**, double-click the filter name in the **Filters** list, and click **OK**.

> **Tip**
> The Filter history list holds a maximum of 10 filters, so at times you may need to use the **Expression Builder** method for reapplying a saved filter.

Learn more

Beyond filters

Filters work well if the number of criteria or conditions contained by the filter are manageable. The filter you created in this tutorial contains only 9 codes. But what if your list of prohibited merchant category codes was several dozen, or more?

A more efficient approach would be to join an Analytics table containing the prohibited codes with the transactions table. Every match in the joined output table would be a prohibited transaction.

Joins are beyond the scope of this tutorial, but they are a frequently used feature in Analytics.

Report results

Once your data analysis is complete, Analytics gives you several different ways to report or present your results.

Traditional reports with columns of data are available, but we'll look at conveying results using the more engaging data visualization described below.

Treemap visualization

This treemap visualization shows the grouped credit card transactions you output in the **Trans_All_Grouped** table. The relation between groups is conveyed in two different ways:

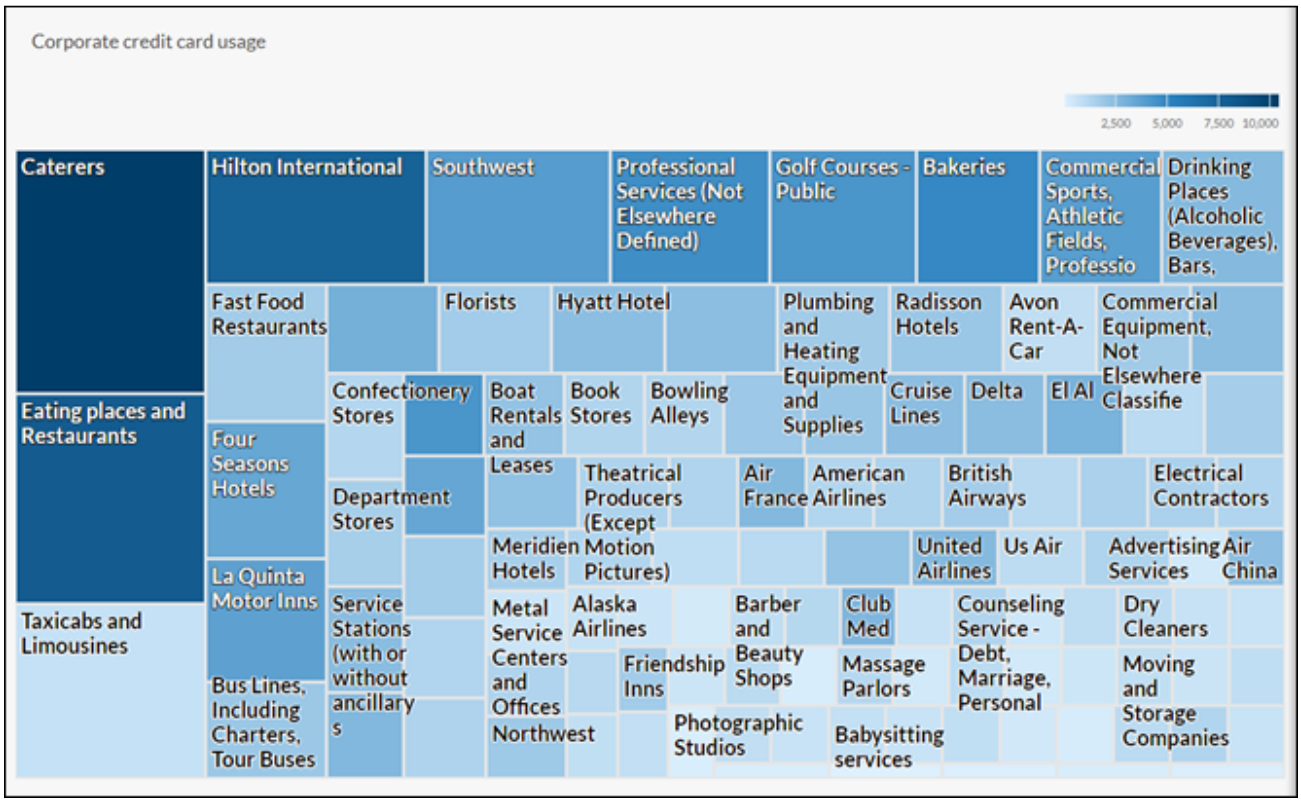
- **size of the box** – indicates the count of individual transactions in each group

  The larger the box, the greater the number of transactions. The boxes are arranged in size from top left to bottom right.

- **color intensity of the box** – indicates the total amount of each group

  The darker the box, the greater the total amount.

So, for example, the size of the **Club Med** box, in the bottom right quadrant, indicates only a small number of transactions, but the color indicates that the total transaction amount is significant.



First, a little pre-work

You're going to create the treemap visualization in Results, the issue remediation app in the cloud-based HighBond platform. Access to a lite version of Results is included in your ACL Robotics subscription.

In order to create the visualization, you must first create a simple, two-level data container to hold it. The first level is called a Collection, and the second level is called an Analysis. They're quick and easy to create.

Sign in to Launchpad and access Results

> **Note**
> If for some reason you cannot sign in to Launchpad or access Results, you can use one of the alternative report creation methods listed in "Other reporting methods in Analytics" on page 91.

Steps

1. Go to Launchpad (www.highbond.com).
2. Enter your HighBond account credentials (e-mail and password) and click **Sign In**.

   Launchpad opens.

3. Click **Results**.

   The Results homepage opens.

> **Note**
>
> If you cannot access Results, you may not be assigned an appropriate subscription type or Results role. Use one of the alternative report creation methods listed in "Other reporting methods in Analytics" on page 91.
>
> If you would like to access Results, contact your company's Analytics account administrator.

Create a Collection

Steps

1. From the Results homepage, click **New Collection**.
2. On the **New Collection** page, in the **Name** field, enter or copy `ACL Tutorial`.
3. At the bottom of the page, click **Create Collection**.

   The Collection settings page opens.

Create an Analysis

Steps

1. At the bottom of the Collection settings page, under **What's Next?**, click **create your first Data Analysis**.

   The **Analysis Details** page opens.

2. On the **Analysis Details** page, in the **Name** field, enter or copy `Sample Report`.
3. Click **Create Analysis**.

   The new **ACL Tutorial** Collection opens with the empty **Sample Report** Analysis that you just created.

> **Note**
>
> Leave Results open. You will be coming back to create the data visualization.

Export data from Analytics to Results

The next stage is to export the **Trans_All_Grouped** table from Analytics to Results.

Steps

1. In Analytics, open the **Trans_All_Grouped** table.
2. From the Analytics main menu, select **Data > Export**.

3. In the **Export** dialog box, select the following options:

Tab	Option	Select
Main	View	select View
	Export As	select HighBond

4. Click **To**, and in the **Select Destination Test** dialog box navigate to the **Sample Report** Analysis container you just created and double-click to open it.
5. In the **New data analytic** field enter or copy `Trans_All_Grouped` and click **Create**.

   You are returned to the **Export** dialog box and an ID number and data center code are prefilled in the **To** text box.
6. Click **OK**.

   The data in the **Trans_All_Grouped** table is exported to Results.

Create the visualization

Now you're ready to create the visualization in Results.

Steps

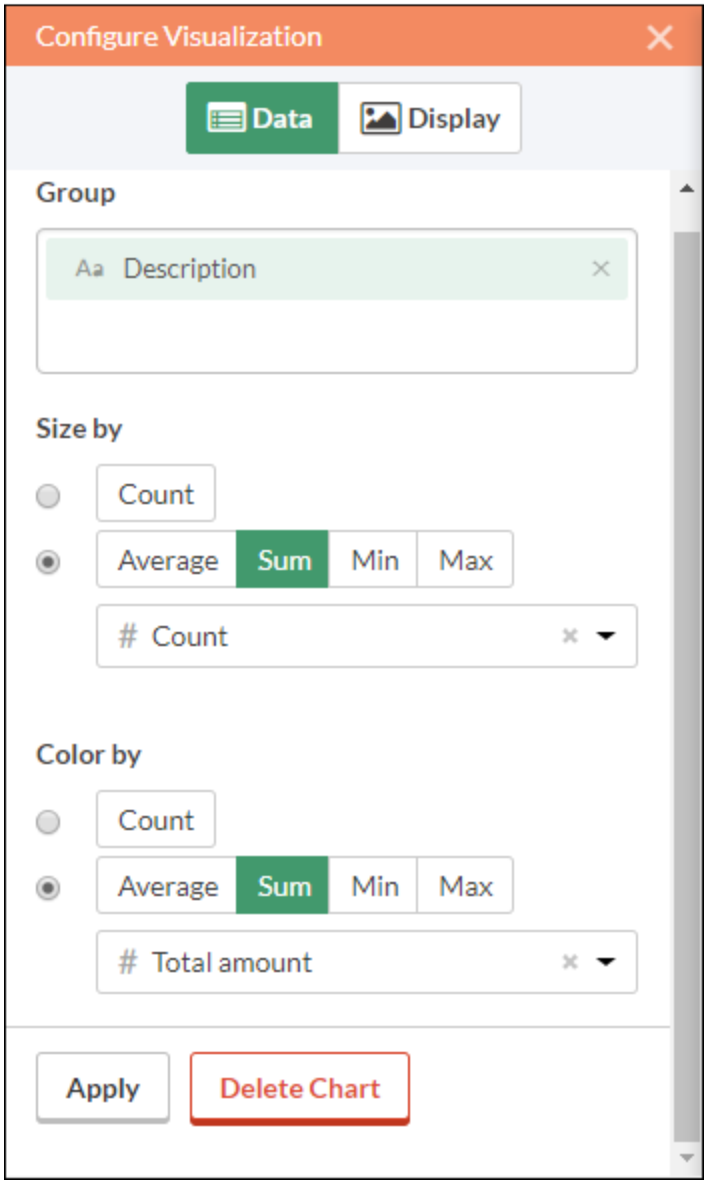
1. Return to the **ACL Tutorial** collection in Results and press **F5** to refresh the browser window.

   The **Trans_All_Grouped** table appears.
2. Under **Remediate**, click **View Records**.

   The **Table View** opens and displays the records.
3. Click **Add Visualization** and click the **Treemap** visualization.

4. In the **Configure Visualization** panel, select the fields and options shown below.

> **Note**
>
> If you can't see the **Configure Visualization** panel, click **Configure**  .



5. Click **Apply**.

   The Treemap visualization is generated.

   You can hover your mouse over the individual boxes in the treemap to see the embedded data.

   If you change the size of the browser window, the treemap dynamically updates by repositioning boxes, and by displaying and suppressing a different selection of associated descriptions.

Save the visualization

If you want to keep any visualizations you create you need to save them. You need to save each visualization individually, and also the container that holds them, called **an interpretation**.

Steps

1. Click **Untitled** at the top left corner of the Treemap visualization and type a title for the visualization such as `Transaction Treemap` and press Enter.
2. Click **Save > Save As** .
3. In the **Title** field, type a name for the interpretation such as `Tutorial visualizations` and click **Save**.

   The interpretation and the visualization are both saved and can be reopened later.
4. Click the name of the collection, `ACL Tutorial`, in title bar to return to the **Sample Report** Analysis container.
5. Click the number in the **Interpretations column**. The **Interpretations** dialog box appears and notice that it lists the newly created interpretation, `Tutorial visualizations`.

   You can create multiple visualizations and interpretations in each Analysis container. Each visualization is based on the data in the **Table View**.

Publish to Storyboards

Create a storyboard to display the visualization you just created. A storyboard is a communication platform that displays multiple visualizations and rich text content in a single presentation.

Steps

1. Open the **Storyboards** app.
2. Click **Add Storyboard**.
3. Enter a descriptive title for your storyboard. Storyboard titles can be a maximum of 80 characters.
4. Click **Add** and select **Add Chart**
5. Select one of the following options:
   - To display table view from the interpretation, select the parent table entry, `Tutorial visualizations`.
   - To display visualization from the interpretation, select the child chart entry, `Transaction Treemap`.

   You can enter a keyword or phrase into the search field to filter the list of available visualizations.
6. In the top right-hand corner, click **Save** > **Save**.

Other reporting methods in Analytics

In addition to the data visualizations available in Results, Analytics has several other methods you can use for reporting the results of your data analysis:

Reporting method	Description
Data visualizations in the Analysis App window	The data visualization capability in Results is also available locally in the Analysis App window, a freestanding component of Analytics.

Reporting method	Description
	Note Some of the charts and visualizations available in Results may not be available in the Analysis App window until a new version of Analytics is released. For more information, see "Interpretations and visualizations" on page 2652.
Legacy Analytics charts	Analytics contains a legacy charting and graphing capability that allows you to create basic visual reports. For more information, see "Working with Analytics graphs" on page 1336.
Traditional columnar reports	In some cases, a traditional text- and number-based report with rows and columns of data is all you need. For more information, see "Formatting and generating Analytics reports" on page 1330.
Third-party reporting tool	You can use a third-party reporting tool such as Tableau or Microsoft BI and import data directly from Analytics. For more information, see "Connecting to Analytics from a third-party reporting application" on page 1348.
Exporting data to Excel or CSV	You can export data to Excel, or to a comma-separated file, and use the reporting capabilities of Excel, or of any tool that can work with a CSV file. For more information, see "Exporting data" on page 203.

You're finished

Congratulations! You've completed your end-to-end introduction to analyzing data using Analytics.

Where to next?

You have several options for continuing to learn about Analytics:

Academy	Academy offers a range of courses for various experience levels. **ACL Analytics Foundations Program** is a series of six mini-courses that teaches Analytics basics for new users. Academy is the Galvanize online training resource center. Go to the course catalog to see the available courses. Academy courses are included at no extra cost for any user with a subscription.
Analytics and	You're currently in the Analytics and ACLScript Help. The Help provides reference-style

ACLScript Help	conceptual material, step-by-step instructions, and ACLScript syntax for all aspects of Analytics. For example, here are the Help topics for the append operation, which formed part of the tutorial you just completed: ◦ "Appending tables" on page 858 (conceptual) ◦ "Append tables" on page 868 (step-by-step instructions) ◦ "APPEND command" on page 1565 (ACLScript syntax)
Community	Community is a web-based platform with a variety of customer resources, including a customer forum where experienced Analytics users share their expertise and answer questions. The customer forum is the best place to learn about the real-world usage and application of Analytics.

Script your work (optional section)

Estimated time	20 minutes
Requirements	No previous scripting experience is required.
Analytics version	13.0 or later (Unicode edition)

You can gain a lot of value using Analytics in an ad hoc or manual fashion without ever writing a script. For the most part, anything that can be done in a script can be done in the user interface, and vice versa. However, to gain the most value, power, and efficiency from Analytics, you need to script.

The good news is that Analytics provides tools to make scripting relatively easy, even for a novice.

The case for scripting

Imagine that in addition to all your current responsibilities you're now responsible for reviewing corporate credit card usage on a regular basis.

Save time

The basic review process is standardized. With each review cycle, you can spend time repeating the basic process manually, or you can save time by automating the process.

Delegate with confidence

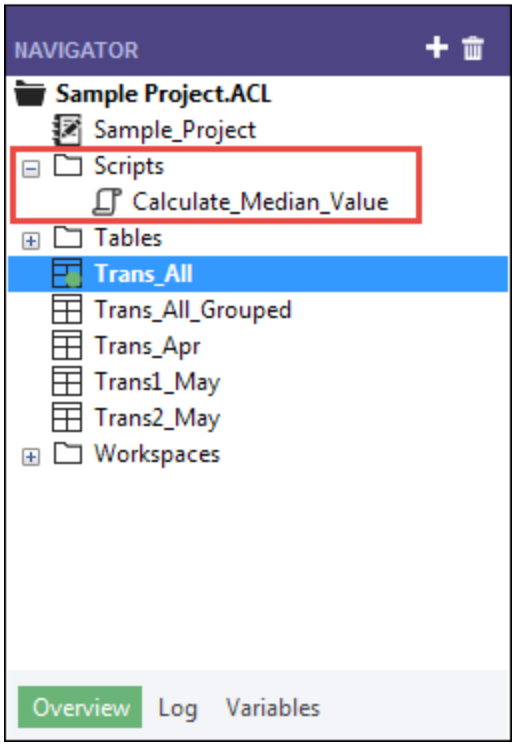
If the process is automated, maybe you can delegate the task to a more junior staff member. A tested script gives you the confidence that less experienced employees can perform the task

consistently and accurately, without a significant increase to their workload.

What is a script?

An Analytics script is a series of ACLScript commands that perform a particular task, or several related tasks. For example, everything that you just did manually in the first part of this tutorial could also be performed using a script.

ACLScript is the command language that forms the basis of Analytics. Scripts are stored in Analytics projects. Individual scripts appear in the **Navigator**, and are prefaced by the script icon .



How the Analytics command log works

You may have noticed that the **Navigator** contains the **Log** tab. As a script writer, you'll discover that the Analytics command log is your best friend.

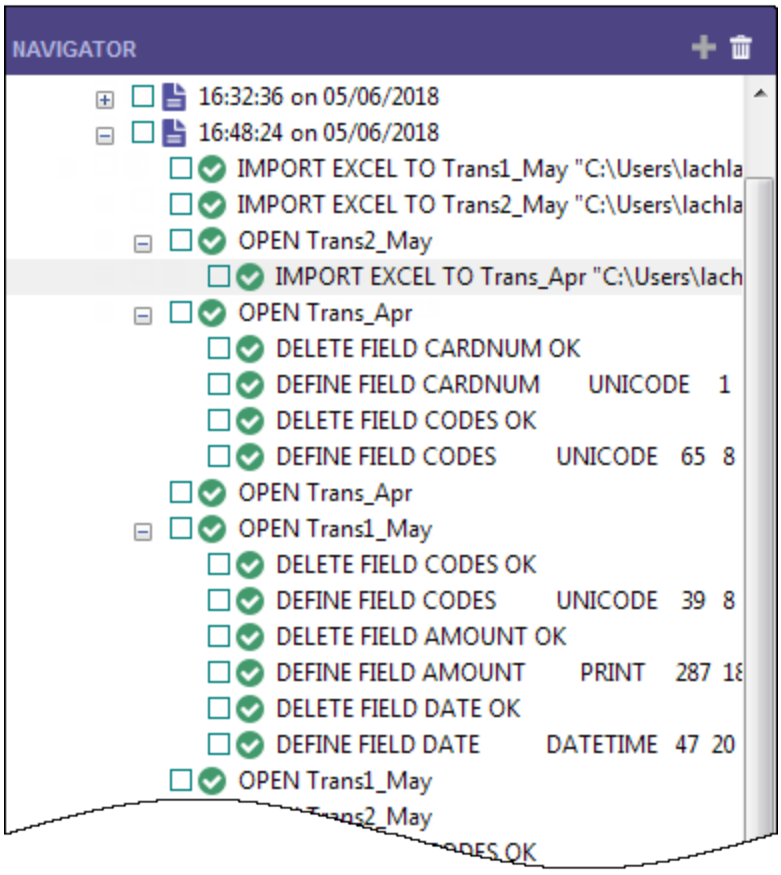
Steps

1. Click the **Log** tab to open it.

   You're looking at the Analytics command log. You can drag the **Navigator** panel wider to see more of the content.

   The log records the exact sequence of commands executed during each Analytics session, and saves them as part of the Analytics project.

If you've just finished the first part of this tutorial, the log contains a list of all the actions you've just performed in the user interface.



2. In the log, locate and click the SUMMARIZE command that outputs results to a new table.

The command prefills the **Command Line** near the top of the Analytics interface, just below the toolbar.

> **Note**
> If the **Command Line** isn't visible, select **Window > Command Line** from the Analytics main menu.

3. Open the **Trans_All** table, if it is not already open.
4. If the `f_Prohibited_codes` filter is applied, remove it.
5. Click in the **Command Line**, change `"Trans_All_Grouped.FIL"` to `"Trans_All_Grouped_2.FIL"`, and press Enter.

   The Summarize command is re-run on the **Trans_All** table and outputs the **Trans_All_Grouped_2** table, which replicates the first output table you created manually.

With a minimal amount of effort you re-performed all your earlier manual work required to summarize the **Trans_All** table. Running a command from the command line is like running a simple one-line script.

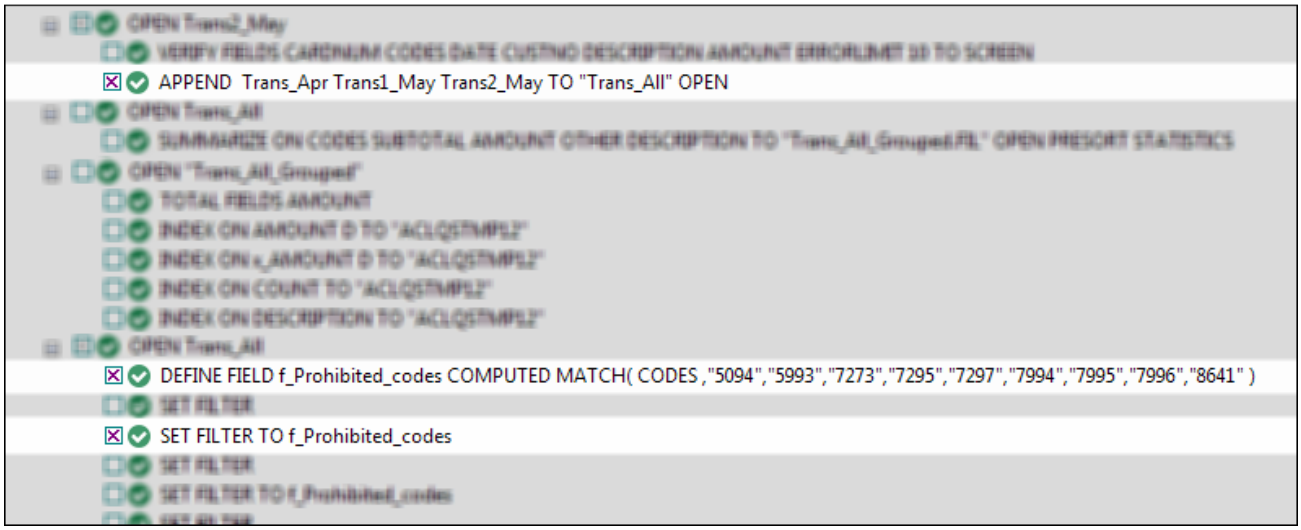
Building a script by copying commands from the log

You'll again reuse ACLScript syntax from the log, but this time you'll copy the syntax to an Analytics script. To keep things quick and easy, you'll script only a portion of the work you performed manually in the tutorial, but you could script all of it.

> **Note**
> We're going to skip over some scripting best practices in order to keep this introduction to scripting brief. The goal is to demonstrate how easy it is for even new users to create scripts in Analytics.

Steps

1. In the log, locate and select the following commands:



2. Right-click the log and select **Save Selected Items > Script**.
3. In the **Save Script As** dialog box, enter the script name `Append_and_filter` and click **OK**.
4. In the **Overview** tab of the **Navigator**, double-click the newly created **Append_and_filter** script to open it in the **Script Editor**.

   The script opens and contains the complete syntax of the three commands you selected in the log.

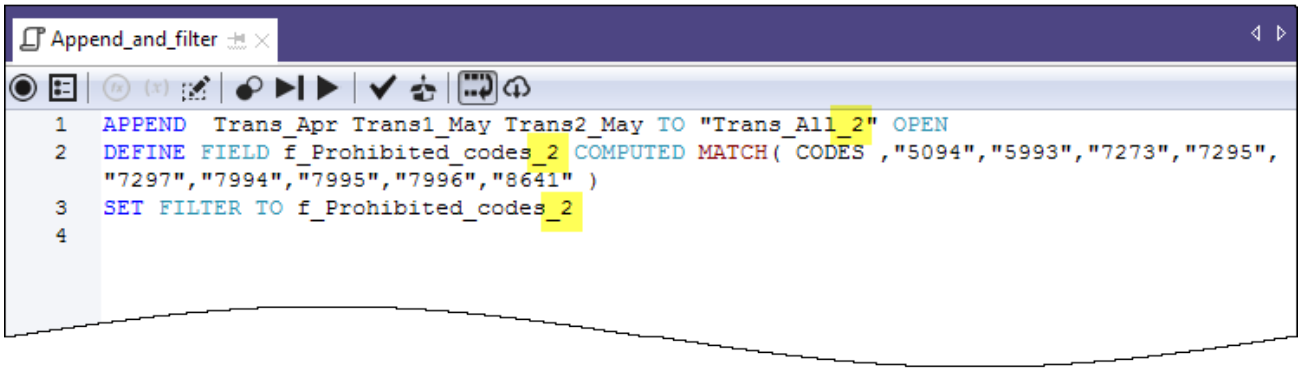
5. Take a moment to read the syntax for each command.

   Do you see how the actions you previously performed in the user interface correspond to individual pieces of ACLScript syntax? For example, after the `APPEND` command, there are the names of the three tables you appended:

`Trans_Apr Trans1_May Trans2_May`

For the most part, the correspondence between ACLScript syntax and actions in the user interface is relatively straightforward, which means the syntax is not that difficult to understand.

6. Modify the script by adding `_2` in the following locations:



You're adding `_2` to avoid name conflicts with the table and filter you already created manually.

7. On the **Script Editor** toolbar click **Run** to run the script.

Click **Yes** to any prompts that appear.

The script runs and performs the following tasks:

- appends the three tables you imported from Excel into a single table, and opens the new table
- creates the prohibited codes filter
- applies the filter to the new table

As you can see, running a script is much faster than performing the same actions manually. Imagine the time savings, and improved consistency, in a real-world situation with much more complex analysis performed on a weekly or monthly basis.

> **Note**
> You can also run a script by right-clicking it in the **Navigator** and selecting **Run**. A script does not have to be open to be run.

The entire tutorial in a script

The entire tutorial you just performed manually appears below in a script (in the "Steps" section). To finish this brief introduction to scripting, you're going to copy the script to Analytics and then redo the tutorial work, but this time with just a couple of clicks of the mouse.

> **Note**
>
> The script assumes that the `Sample Data Files` folder is installed in the default location. If the folder is installed in a different location, you need to modify the navigation paths in the script to point to the correct location.
>
> The tables created by the script are appended with `_s` so that they don't overwrite the tables you created manually.

Steps

Create a new, empty script

1. In the **Overview** tab in the **Navigator**, right-click the **Scripts** folder and select **New > Script**.
2. Right-click the **New_Script**, select **Rename**, type or copy `Getting_Started_tutorial`, and press **Enter**.

Copy and paste the tutorial script

1. Click **Show me the script** below.
2. Click and drag to select the entire script and then press **Ctrl+C** to copy the script.

> **Note**
>
> It's important that you select the entire script and don't miss any lines. Alternately, you can download a text file with the script here: Getting started tutorial (Unicode edition)

3. Click in the Script Editor window and press **Ctrl+V** to paste the script syntax into the empty **Getting_Started_tutorial** script.

Update and save the script

1. Update the navigation paths in the script:
   a. Click the first line of the script.
   b. Right-click and select **Find**.
   c. Type the following entries in the **Replace** dialog box:
      - **Find what:** `user_account_name`
      - **Replace with:** *the actual account name on your computer*
   d. Perform the find-and-replace of all instances of `user_account_name`
2. Click **Save the Open Project**  , and click **Yes** in the prompt that appears.

   If you do not find the save icon, select **Window > Toolbar** in the Analytics main menu to enable the toolbar.

Run the script

On the **Script Editor** toolbar click **Run**  to run the script.

The script runs and replicates all the tutorial work. Interactive notifications provide key information as the script runs.

Show me the script

> **Note**
>
> If you haven't worked with scripts before, the script syntax may look overwhelming at first. Keep in mind that almost all the syntax was simply copied from the Analytics log.
>
> The syntax for the interactive notifications in the script (DIALOG commands) was auto-generated by another relatively simple Analytics tool.
>
> The green COMMENT commands walk you through the script at a high level. You'll recognize the tasks that you just completed in the preceding tutorial.

```
COMMENT
*** Unicode Edition ***
This script performs all the actions that you performed manually in the "Get-
ting Started with ACL Analytics" tutorial.
END

COMMENT Allows overwriting of tables without a user confirmation.
SET SAFETY OFF

COMMENT Imports the three Excel worksheets.

IMPORT EXCEL TO Trans1_May_s "C:\Users\user_account_name\Documents\ACL
Data\Sample Data Files\Trans1_May_s.fil" FROM "Trans_May.xls" TABLE "Trans1_
May$" KEEPTITLE  FIELD "CARDNUM" C WID 19 AS "" FIELD "CODES" N WID 4 DEC 0 AS
"" FIELD "DATE" D WID 19 PIC "YYYY-MM-DD hh:mm:ss" AS "" FIELD "CUSTNO" C WID
6 AS "" FIELD "DESCRIPTION" C WID 95 AS "" FIELD "AMOUNT" C WID 9 AS ""

IMPORT EXCEL TO Trans2_May_s "C:\Users\user_account_name\Documents\ACL
Data\Sample Data Files\Trans2_May_s.fil" FROM "Trans_May.xls" TABLE "Trans2_
May$" KEEPTITLE  FIELD "CARDNUM" C WID 19 AS "" FIELD "CODES" N WID 4 DEC 0 AS
"" FIELD "DATE" D WID 19 PIC "YYYY-MM-DD hh:mm:ss" AS "" FIELD "CUSTNO" C WID
6 AS "" FIELD "DESCRIPTION" C WID 155 AS "" FIELD "AMOUNT" C WID 9 AS ""

IMPORT EXCEL TO Trans_Apr_s "C:\Users\user_account_name\Documents\ACL
Data\Sample Data Files\Trans_Apr_s.fil" FROM "Trans_April.XLS" TABLE "Trans_
Apr$" KEEPTITLE FIELD "CARDNUM" N WID 16 DEC 0 AS "" FIELD "AMOUNT" N WID 6
DEC 2 AS "" FIELD "DATE" D WID 10 PIC "YYYY-MM-DD" AS "" FIELD "CODES" N WID 4
DEC 0 AS "" FIELD "CUSTNO" C WID 6 AS "" FIELD "DESCRIPTION" C WID 45 AS ""
```

```
COMMENT Adjusts the table layouts of the three new Analytics tables.

OPEN Trans_Apr_s
DELETE FIELD CARDNUM OK
DEFINE FIELD CARDNUM UNICODE 1 32 WIDTH 35
DELETE FIELD CODES OK
DEFINE FIELD CODES UNICODE 65 8 WIDTH 11

OPEN Trans1_May_s
DELETE FIELD CODES OK
DEFINE FIELD CODES UNICODE 39 8 WIDTH 11
DELETE FIELD AMOUNT OK
DEFINE FIELD AMOUNT PRINT 287 18 2 WIDTH 9
DELETE FIELD DATE OK
DEFINE FIELD DATE DATETIME 47 20 PICTURE "YYYY-MM-DD" WIDTH 27

OPEN Trans2_May_s
DELETE FIELD CODES OK
DEFINE FIELD CODES UNICODE 39 8 WIDTH 11
DELETE FIELD AMOUNT OK
DEFINE FIELD AMOUNT PRINT 407 18 2 WIDTH 9
DELETE FIELD DATE OK
DEFINE FIELD DATE DATETIME 47 20 PICTURE "YYYY-MM-DD" WIDTH 27

COMMENT Verifies the imported data and provides user notifications.

OPEN Trans_Apr_s
VERIFY FIELDS CARDNUM AMOUNT DATE CODES CUSTNO DESCRIPTION ERRORLIMIT 10
IF WRITE1=0 DIALOG (DIALOG TITLE "User Dialog" WIDTH 490 HEIGHT 100 )
(BUTTONSET TITLE "&OK;&Cancel" AT 360 12 DEFAULT 1 ) (TEXT TITLE "Trans_Apr_s
table:  0 data validity errors detected" AT 12 28 )
IF WRITE1>0 DIALOG (DIALOG TITLE "User Dialog" WIDTH 490 HEIGHT 100 )
(BUTTONSET TITLE "&OK;&Cancel" AT 360 12 DEFAULT 1 ) (TEXT TITLE "Trans_Apr_s
table:  %WRITE1% data validity errors detected" AT 12 28 )

OPEN Trans1_May_s
VERIFY FIELDS CARDNUM CODES DATE CUSTNO DESCRIPTION AMOUNT ERRORLIMIT 10
IF WRITE1=0 DIALOG (DIALOG TITLE "User Dialog" WIDTH 490 HEIGHT 100 )
(BUTTONSET TITLE "&OK;&Cancel" AT 360 12 DEFAULT 1 ) (TEXT TITLE "Trans1_May_s
table:  0 data validity errors detected" AT 12 28 )
IF WRITE1>0 DIALOG (DIALOG TITLE "User Dialog" WIDTH 490 HEIGHT 100 )
(BUTTONSET TITLE "&OK;&Cancel" AT 360 12 DEFAULT 1 ) (TEXT TITLE "Trans1_May_s
table:  %WRITE1% data validity errors detected" AT 12 28 )

OPEN Trans2_May_s
VERIFY FIELDS CARDNUM CODES DATE CUSTNO DESCRIPTION AMOUNT ERRORLIMIT 10
IF WRITE1=0 DIALOG (DIALOG TITLE "User Dialog" WIDTH 490 HEIGHT 100 )
```

```
(BUTTONSET TITLE "&OK;&Cancel" AT 360 12 DEFAULT 1 ) (TEXT TITLE "Trans2_May_s
table:  0 data validity errors detected" AT 12 28 )
IF WRITE1>0 DIALOG (DIALOG TITLE "User Dialog" WIDTH 490 HEIGHT 100 )
(BUTTONSET TITLE "&OK;&Cancel" AT 360 12 DEFAULT 1 ) (TEXT TITLE "Trans2_May_s
table:  %WRITE1% data validity errors detected" AT 12 28 )

COMMENT Verifies the Badfile table and provides a user notification.
OPEN Badfile
VERIFY FIELDS InvoiceNo Prodno Price OrderQty ShipQty Total ERRORLIMIT 10
IF WRITE1=0 DIALOG (DIALOG TITLE "User Dialog" WIDTH 490 HEIGHT 100 )
(BUTTONSET TITLE "&OK;&Cancel" AT 360 12 DEFAULT 1 ) (TEXT TITLE "Badfile
table:  0 data validity errors detected" AT 12 28 )
IF WRITE1>0 DIALOG (DIALOG TITLE "User Dialog" WIDTH 490 HEIGHT 100 )
(BUTTONSET TITLE "&OK;&Cancel" AT 360 12 DEFAULT 1 ) (TEXT TITLE "Badfile
table:  %WRITE1% data validity errors detected" AT 12 28 )
CLOSE

COMMENT Appends the three new Analytics tables into a single combined table.
APPEND Trans_Apr_s Trans1_May_s Trans2_May_s TO "Trans_All_s" OPEN
DIALOG (DIALOG TITLE "User Dialog" WIDTH 630 HEIGHT 100 ) (BUTTONSET TITLE
"&OK;&Cancel" AT 500 12 DEFAULT 1 ) (TEXT TITLE "The combined transactions
table (Trans_All_s) contains %WRITE1% records" AT 12 28 )

COMMENT Groups the combined table by merchant category code.
SUMMARIZE ON CODES SUBTOTAL AMOUNT OTHER DESCRIPTION TO "Trans_All_Grouped_
s.FIL" OPEN PRESORT STATISTICS
DIALOG (DIALOG TITLE "User Dialog" WIDTH 700 HEIGHT 100 ) (BUTTONSET TITLE
"&OK;&Cancel" AT 570 12 DEFAULT 1 ) (TEXT TITLE "The grouped transactions
table (Trans_All_Grouped_s) contains %WRITE1% merchant category codes" AT 12
28 WIDTH 550 )

COMMENT Filters the combined table to show only prohibited transactions.
OPEN Trans_All_s
DEFINE FIELD f_Prohibited_codes COMPUTED MATCH(CODES, "5094", "5993", "7273",
"7295", "7297", "7994", "7995", "7996", "8641")
SET FILTER TO f_Prohibited_codes

COMMENT Successful completion message.
DIALOG (DIALOG TITLE "User Dialog" WIDTH 490 HEIGHT 100 ) (BUTTONSET TITLE
"&OK;&Cancel" AT 360 12 DEFAULT 1 ) (TEXT TITLE "The script successfully com-
pleted" AT 12 28 )

COMMENT A user confirmation is required before overwriting a table.
SET SAFETY ON
```

You're finished

That's the end of this brief introduction to scripting. We hope you've seen enough to be convinced of the value of scripting and that you want to learn more.

Where to next?

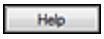
You have several options for learning more about scripting in Analytics:

Option	Useful information
Tutorials	The Analytics Help contains the following beginner-level tutorials: ◦ "Scripting for complete beginners" on page 1379 ◦ "Analytics scripting basics" on page 1397 ◦ "How to use functions" on page 1424 The Help also contains a complete ACLScript language reference with detailed information about every Analytics command and function.
Academy	Academy offers both an introductory and an advanced scripting course: ◦ Introduction to scripting in ACL Analytics (ACL 106) ◦ ACL Analytics Scripting (ACL 303) Academy is the Galvanize online training resource center. Go to the course catalog to see the available courses. Academy courses are included at no extra cost for any user with an ACL subscription.
Community	Community is a web-based platform with a variety of customer resources, including a customer forum where Analytics scripting is frequently discussed in depth.

Get help with Analytics

There are several ways to get help when you are working with Analytics.

Context-sensitive Help

Press **F1** from any location in Analytics, or click the **Help** button  from most locations, to open a Help topic that explains the currently active window, dialog box, tab, or wizard screen.

From this initial Help topic you can often click links to access additional, more detailed information in the online Help.

Online Help and documentation

From the Analytics main menu, select **Help > Analytics Help** to access the online Help directly, without any context sensitivity.

Generally, you should use the most recent version of the Help even if you are using an older version of Analytics.

Software version and subscription information

From the Analytics main menu, select **Help > About** to open a dialog box with the following information:

- software version number
- edition type (**Unicode** or **non-Unicode**)
- name and company of the subscription license holder
- subscription expiry date

Community

Go to Community, a web-based platform with a variety of customer resources, including a customer forum where you can post questions about Analytics features and functionality.

Support

(Account sign-in required)

From the Analytics main menu, select **Help > Contact Galvanize** to open a web browser and connect to Support.

The Analytics user interface

This section provides general information about the Analytics user interface:

- an overview of user interface menus, tabs, dialog boxes, and other user interface elements
- the structure of Analytics tables
- customizable Analytics features
- an overview of Analytics projects
- other general information

Open Analytics

To open Analytics, double-click the ACL for Windows desktop shortcut, then click an option in ACL for Windows:

- **New Analytic Project** – create a new, empty Analytics project
- **Open Analytic Project** – open an existing Analytics project
- **Open Analysis App** – open an existing analysis app
- Under **Recent Analytics Files**, or **Sample Files** – open a recently opened or a sample Analytics project (.acl) or analysis app (.aclx)

If you create or open an Analytics project it opens in Analytics. If you open an analysis app, it opens in the Analysis App window, a free-standing component of Analytics.

Close Analytics

To close Analytics, select **File > Exit**.

If any unsaved changes are detected in your project, you are prompted to save them before you exit. Click **Yes** in the confirmation dialog box to save your changes and exit.

Analytics user interface overview

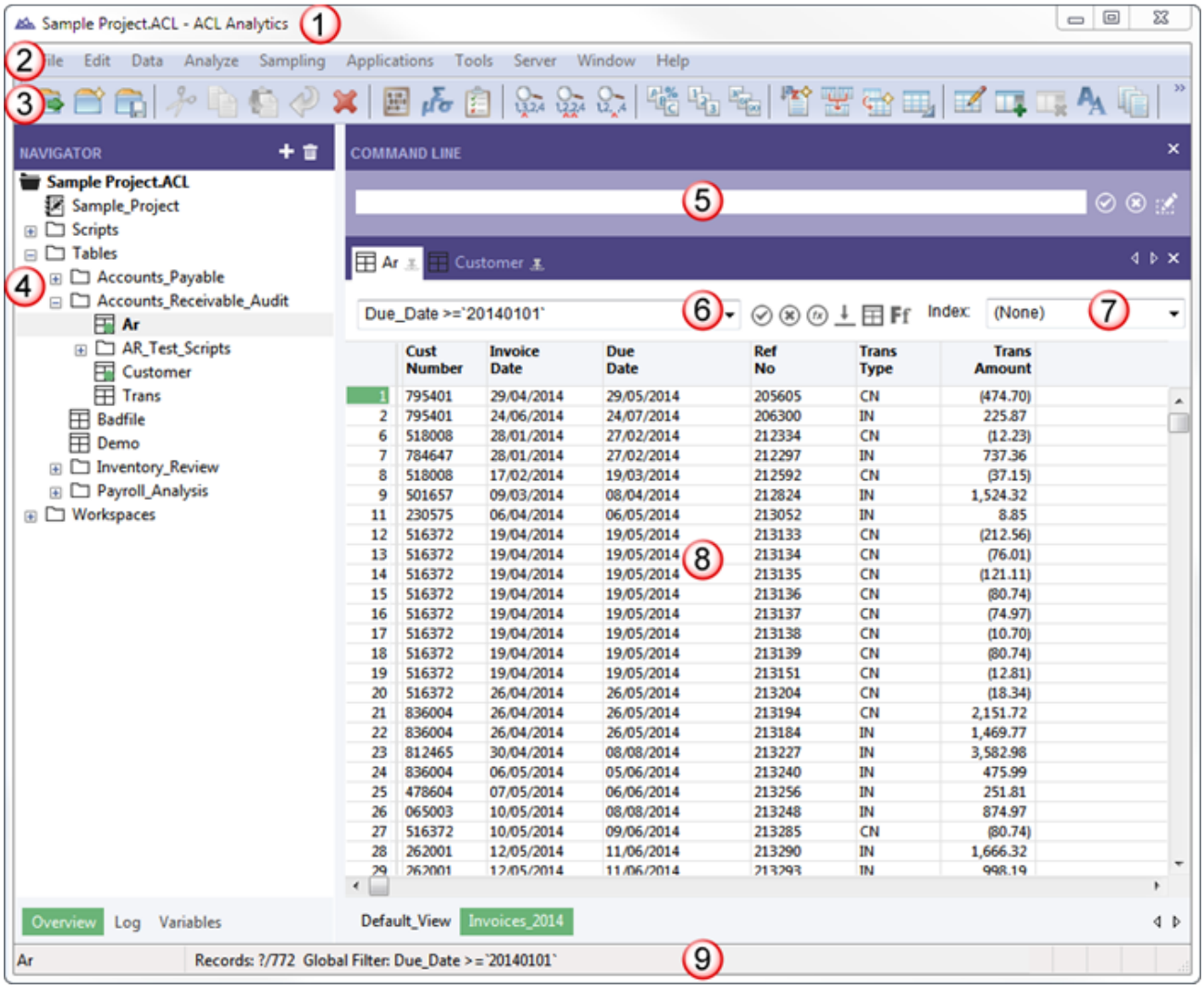
The Analytics user interface includes a number of elements that display specific types of information, and allow you to work with data:

- "The main Analytics user interface" below
- "Command dialog boxes" on page 110
- "Additional user interface elements" on page 114
  - Data Definition Wizard
  - Table Layout dialog box
  - Expression Builder dialog box
  - Dialog Builder
  - The Analysis App window

Understanding the organization and function of the various user interface elements helps you work effectively with Analytics.

The main Analytics user interface

The main Analytics user interface appears automatically when you open Analytics.



Number	Element	Description
1	**Title Bar**	Displays the name of the open Analytics project, and the ACL for Windows component name.
2	**Main Menu**	Provides access to most Analytics features, including menu commands for: ◦ Working with Analytics projects ◦ Performing data analysis ◦ Configuring options and connection settings
3	**Toolbar**	Buttons in the toolbar are shortcuts to common actions. Analytics enables buttons that are relevant to your current activity. To display or to hide the toolbar, select **Window > Tool bar**.

Number	Element	Description
		Note You can customize the buttons contained in the toolbar. For more information, see "Customize the Analytics toolbar" on page 152.
4	**Navigator**	Displays information in three tabs about the open Analytics project: ◦ **Overview** tab - displays all items that belong to the project You can right-click any project item to perform an action. To organize items in the **Overview**, right-click the project icon and select **New > Folder**. You can drag any project item into the folders that you create in the **Overview**. ◦ **Log** tab - displays the Analytics command log All actions you take associated with the project are recorded and organized chronologically in the log. Double-click log entries to open them, and right-click log entries to perform an action. ◦ **Variables** tab - displays the names, values, and data categories of any variables in the project The contents of the tab are dynamically updated as variables are created, deleted, or changed in value. Variable names are listed alphabetically. **Tip** To resize the **Navigator**, drag the divider between the **Navigator** and the display area. You can also double-click the divider to close or open the **Navigator**.
5	**Command Line**	Allows you to enter ACLScript commands. To display or to hide the command line, select **Window > Command Line**.
6	**Filter and Quick Search**	A text box and drop-down list that allow you to perform two different tasks: ◦ Apply a filter to the data in the View tab ◦ Enter one or more search terms to perform a quick search of the data in the View tab
7	**Index**	Allows you to apply existing indexes to the table and to see if an index is currently applied
8	**Display Area**	Displays different types of information in the following tabs: ◦ **View tab** (shown above) - displays the active Analytics table You can pin the View tab if you want to open more than one table. An additional View tab opens for each additional table you open. You can switch between tables using the individual View tabs. The record number column is at the far left of the View tab. The number of the currently selected record is highlighted in green. If multiple views exist for the same table, buttons at the bottom of the View tab let you switch between views.

Number	Element	Description
		◦ **Results tab** - displays the results of an analytical operation when output to screen or graph, or displays selected command log entries You can pin the Results tab to keep the content visible and cause subsequent results to appear in an additional Results tab. For operations with text and graph output, buttons at the bottom of the tab let you switch back and forth between the two formats. ◦ **Script Editor tab** - displays the contents of a new or existing script You can edit scripts manually, or use the tools available in Analytics, such as syntax capture and copying commands from the log, to edit a script. ◦ **Workspace Editor tab** - displays the field definitions in a new or existing workspace Workspaces allow you to share field definitions among Analytics tables. **Tip** To resize the display area, drag the divider between the display area and the **Navigator**. You can also double-click the divider to close or open the **Navigator**.
9	**Status Bar**	Displays information such as: ◦ The name of the active Analytics table ◦ The record count ◦ The details of any filters currently applied to the table ◦ The name of any currently running script

Command dialog boxes

When you select an Analytics command such as Summarize or Duplicates from the main menu, a command dialog box opens. These dialog boxes contain options that allow you to specify:

- The input field or fields for the operation
- Various options that affect the behavior of the operation
- The format of the output results

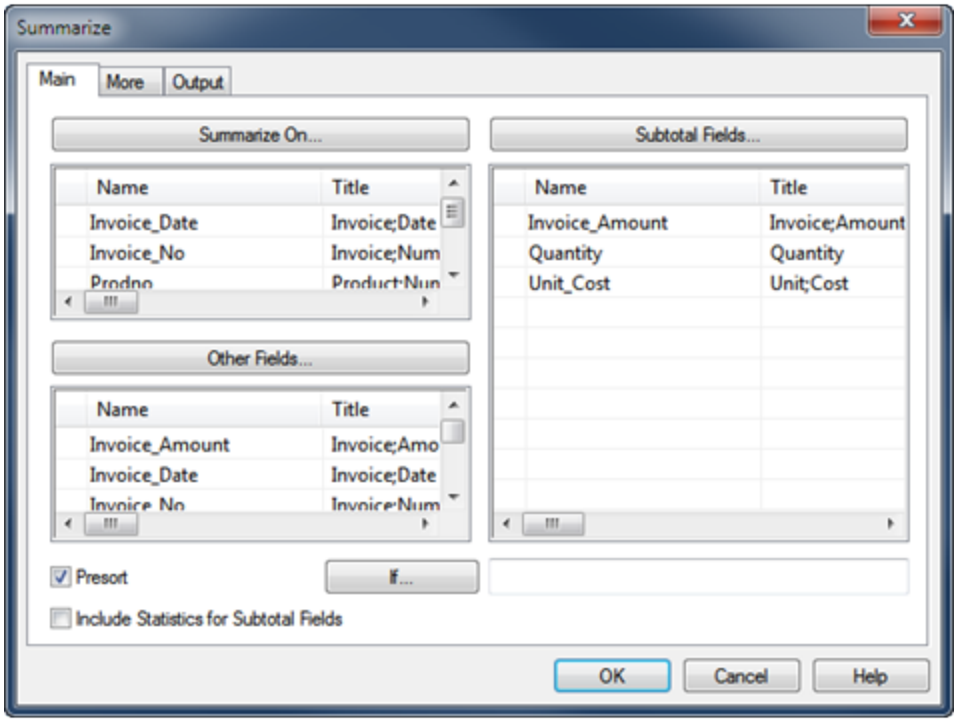
Depending on the operation, the options are organized on two or three tabs in the dialog box:

- **Main** tab
- **More** tab, or **Output** tab, or both

The figures below show examples of the three tabs.

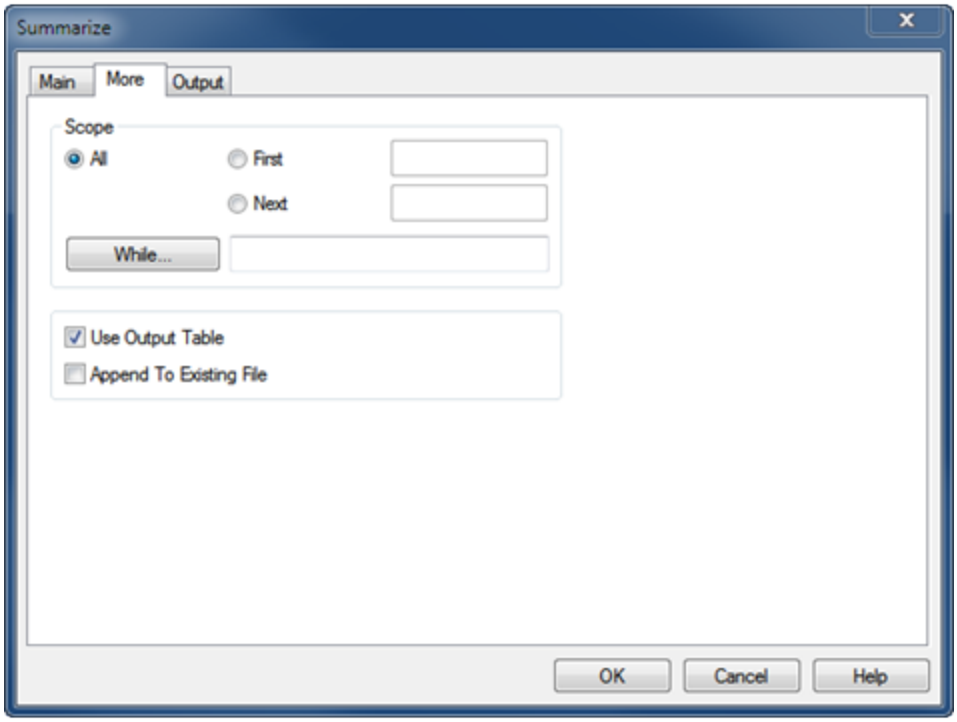
> **Note**
>
> The options in the dialog boxes vary somewhat depending on the operation you select. There are a number of standard options, explained below, that appear for most operations.
>
> Options that are not standard are explained elsewhere in the Analytics and ACLScript Help.

The Main tab



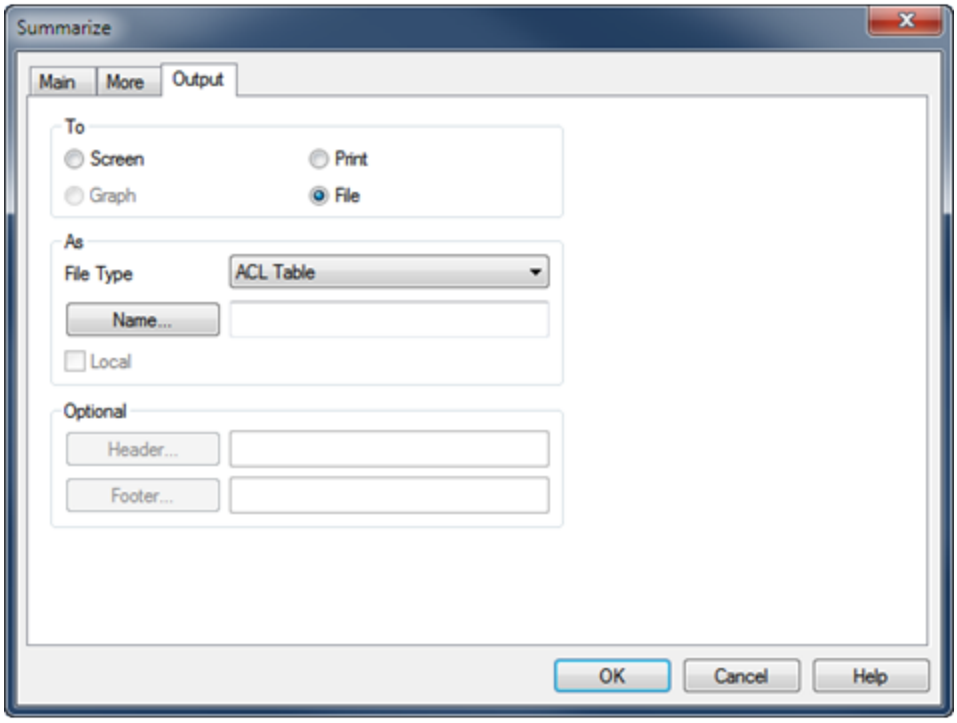
Main tab option	Allows you to . . .
field list or lists	Specify the input field or fields for the operation
Subtotal Fields	Specify one or more numeric fields to optionally subtotal as part of the operation
Presort	Specify that the input field is automatically sorted before the operation is performed
If	Specify an IF statement that excludes records that do not meet the specified condition from the operation
To (not shown)	Specify the name and location of the Analytics table that will contain the output results **Note** Appears on the **Output** tab (as **Name**) in command dialog boxes that have an **Output** tab

The More tab



More tab option	Allows you to . . .
Scope All | First | Next | While	Specify how many records in a table are processed by an operation: ◦ All - all records are processed ◦ First - the specified number of records are processed, starting at the first record in the table ◦ Next - the specified number of records are processed, starting at the currently selected record in the table ◦ While - records are processed, starting at the first record in the table, as long as the WHILE statement evaluates to true
Use Output Table	Specify that an Analytics table containing output results opens automatically upon completion of the operation Appears on either the **Main** tab or the **More** tab.
Append To Existing File	Specify that output results contained in an Analytics table or text file are added to the bottom of an existing Analytics table or text file

The Output tab



Output tab option	Allows you to . . .
To Screen | Print | Graph | File	Specify the format of the output results: ◦ Screen - displays the results in the Analytics display area ◦ Print - sends the results to the default printer ◦ Graph - creates a graph of the results and displays it in the Analytics display area ◦ File - saves or appends the results to an Analytics table or a file Some operations do not support all four output formats
File Type	Specify an Analytics table or a text file when you save output results to a file Depending on the operation, you may be able to save to either a table or a text file, or to only one of these options
Name	Specify the name and location of the Analytics table or text file that will contain the output results Appears on the **Main** tab (as **To**) in command dialog boxes that do not have an **Output** tab
Local	Specify whether to save an Analytics table with output results locally or to the server (only enabled when connected to a server table) Appears on either the **Main** tab or the **Output** tab
Header	Specify a text header to accompany the output results

Output tab option	Allows you to . . .
	Not supported for all format types
Footer	Specify a text footer to accompany the output results Not supported for all format types

Additional user interface elements

Analytics also includes the following user interface elements, which are used to define and import data, work with tables, build expressions, insert custom dialog boxes into scripts, and work with analytics and analysis apps:

- **Data Definition Wizard** - a page-based wizard that provides a standard way to access a variety of data sources, mostly file-based

  The **Data Definition Wizard** is automatically displayed when you create a new Analytics project, and when you add a new Analytics table to an existing project.

  For more information, see "Defining and importing data" on page 224.

- **Data Access window** - a visual interface that contains a number of data connectors you can use for accessing source data in either databases or files

  For more information, see "Defining and importing data" on page 224.

- **Table Layout dialog box** - used to define or modify Analytics table layouts

  Table layouts specify how Analytics should identify records in the data source and read individual fields.

  For more information, see "Table Layout dialog box" on page 699.

- **Expression Builder dialog box** - used to enter an expression in Analytics

  An expression is a statement that combines data fields, operators, functions, filters, and variables that Analytics evaluates and returns a value for.

  For more information, see "Expression Builder overview" on page 799.

- **Dialog Builder** - used to create custom dialog boxes in Analytics scripts

  Custom dialog boxes provide user interaction or feedback when a script is running.

  For more information, see "Creating custom dialog boxes" on page 1508.

- **The Analysis App window** - used to run analysis apps and create data interpretations

  The Analysis App window is a freestanding component of Analytics that provides a simple user interface for running analytics, and bundled sets of analytics called analysis apps. It also provides advanced data interpretations and visualizations, giving users a number of options for reviewing the results of analysis.

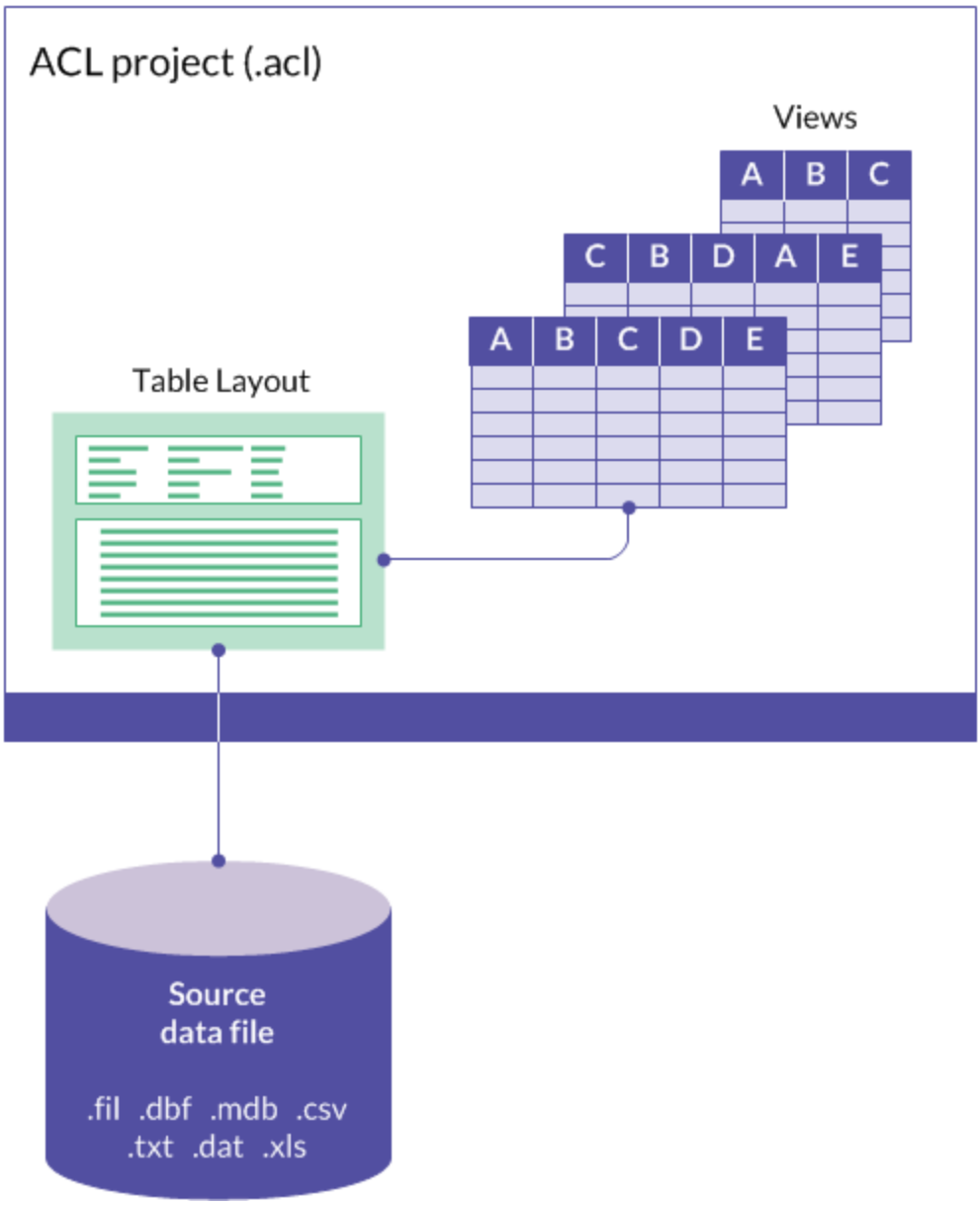
  For more information, see "Working with analysis apps" on page 2642.

The structure of Analytics tables

Data in Analytics is contained in tables. When you view and analyze data in Analytics, you are doing the work in an Analytics table. Analytics tables have a three-part structure:

- the source data file
- the table layout
- one or more views

Understanding how Analytics tables are structured is fundamental to working effectively in Analytics. The diagram below illustrates the structure of an Analytics table.



The source data file

The source data file contains raw data, without Analytics formatting information such as field names. It exists outside the Analytics project. It can reside in the same Windows folder as the Analytics project, or it can be located elsewhere on the same computer, or in a network location, or in a database.

Analytics source data file

In most cases, when you create a new Analytics table, Analytics copies and imports source data into a new Analytics source data file saved in a flat file format with a `.fil` extension. The .fil file serves as the basis for the Analytics table.

The flat file format has fixed-length fields, and records identified by a fixed length, or by carriage return and line feed codes, making it an easy format to interpret.

In some cases, such as .dbf and .txt files, Analytics can connect directly to the existing source data file, and a .fil file is not required or created.

The table layout

The table layout is a structured interpretation of the raw data in the source data file. The table layout contains metadata such as field definitions, which specify field names, the start position of fields, the length of fields, the data type of fields, and so on.

Analytics requires the table layout in order to make sense of the raw data. The table layout also specifies the location of the source data file. The table layout is part of the Analytics project file (.acl).

For more information, see "Structuring data with table layouts" on page 696.

One or more views

The view is an arrangement of named columns, with numbered records, that displays the data structured by the table layout. When you look at data in the main Analytics window, you are looking at a view.

You can create multiple views for the same table, each of them with a different selection and arrangement of columns. The view is part of the Analytics project file (.acl).

Views also format the data for Analytics reports.

For more information, see "Displaying data with table views" on page 767.

Customizing Analytics

Analytics is installed with a standard set of configuration settings that define the default behavior of the application. You can change any of these settings in the **Options** dialog box (**Tools > Options**) to modify application behavior, including:

- turning features on or off
- changing how data is displayed
- controlling some aspects of command output

Other customizable options

This section of the guide also includes information about changing font settings for views and reports, customizing the Analytics toolbar, and customizing the **Applications** menu to display user-defined menu items such as scripts and frequently used commands.

Script Editor

For information about customizing the Script Editor, see "Customizing the Script Editor" on page 1500.

Configuring Analytics options

Use the **Options** dialog box to configure a variety of options that control how Analytics features behave.

If required, you can configure the options differently for individual Analytics projects. For more information, see "How Analytics preferences files work" on page 146.

Revert all options to the default settings

To revert all configurable options to the Analytics default settings, click **Factory** at the bottom of the **Options** dialog box.

> **Caution**
> Clicking **Factory** sets all options on all **Options** tabs to their default settings, not just the options on the active tab.

Settings stored in a preferences file

Settings for the configurable options are stored in a preferences file (.prf file). For more information, see "How Analytics preferences files work" on page 146.

Steps

1. From the Analytics main menu, select **Tools > Options**.
2. Click the tab with the option that you want to change and modify the setting.

   The following tabs are used to group related options:

   - System tab
   - Interface tab
   - Table tab
   - View tab
   - Command tab
   - Date and Time tab
   - Numeric tab
   - Print tab
   - Application Font tab
3. Click **OK** to save your changes.

System options

Use the option in the **System** tab to control how memory is used for sorting and indexing operations.

Use Additional System Resources for Sorting and Indexing

When this option is selected, sorting and indexing operations can be faster, depending on the size of the file being sorted or indexed, and the available memory on your computer.

Selecting the option allows Analytics to use a memory mapped file for sorting and indexing operations. Using a memory mapped file can speed up the sorting or indexing of files that exceed the available RAM on your computer. For example, a 500-MB file might take the same length of time to sort regardless of how the option is set, but a 3-GB file might sort significantly faster if you enable the option.

Similar to sorting smaller files, sorting very large files of multiple gigabytes may not show any improvement in speed when the option is enabled. If you routinely sort very large files and performance is an issue, you should consider increasing your computer's RAM, increasing the Windows paging file size, or both.

Additional information

- If Analytics is unable to determine the number of records in a file being sorted or indexed, a memory mapped file is not used for the operation, even if **Use Additional System Resources for Sorting and Indexing** is selected.
- Using additional system resources for sorting and indexing may slow down other tasks while sorting or indexing is in progress.
- Unlike all other options in the **Options** dialog box, the setting for this option is not stored in the Analytics preferences file. The option applies only to the computer on which it is set.

  For more information about options stored in the preferences file, see "How Analytics preferences files work" on page 146.

Interface options

Use the options in the **Interface** tab to specify some of the basic behavior of Analytics.

Include Filters in Field Lists

Select this option if you want logical fields, including filters, to appear in field lists. Logical fields are normally excluded from field lists. Because filters are actually implemented as logical fields, you can use this option to have filters appear in field lists so you can apply commands to them.

Warn Before Overwriting Files

Select this option if you want Analytics to display a confirmation dialog box before overwriting any of the following:

- fields in table layouts
- Analytics tables
- files, including Analytics data files (.fil)

If you turn this option off, Analytics overwrites fields, tables, and files without asking for confirmation.

You can also use the `SET SAFETY` command in a script or on the command line to turn this option on or off. By turning this option off, you can avoid interruptions while executing a script that is designed to overwrite existing fields, tables, or files.

If you change the setting using the **Warn Before Overwriting Files** checkbox, the change remains in effect until you specifically change it again. If you use the `SET SAFETY` command to change the setting, the change remains in effect for the duration of the Analytics session only.

Changes to this setting are recorded in the log using the following syntax:

```
SET SAFETY {ON|OFF}
```

Enable ACL Server integration

Select this option to enable the functions that allow you to connect to AX Server.

Disable auto complete in scripts

Select this option to turn off keyword auto-completion in the **Script Editor**. On-screen help for function parameters cannot be disabled.

Disable Script Syntax Check Before Commit Scripts

Select this option to turn off script syntax checking when committing scripts to the Robots app in HighBond.

> **Note**
> This option controls script syntax checking only. It does not control analytic header validation, which is a separate process that cannot be disabled.

Beep(s) Upon Task Completion

This option indicates the number of beeps that sound when Analytics completes a task. Enter a number from 0 to 255. This option is especially useful if you expect a task to take a long time. The default is 0.

Changes to this setting are recorded in the log using the following syntax:

`SET BEEP value`

Table options

Use the options in the **Table** tab to specify how Analytics processes tables.

Automatically Profile on Open

When the **Automatically Profile on Open** option is turned on, Analytics automatically executes the Profile command on all numeric fields when you open a project, change tables, or change global filters.

Analytics retains the information and uses it to provide minimum and maximum values for histograms and stratifications, as well as absolute values for monetary unit sampling.

> **Note**
> Tables with numeric fields will open more slowly with this option on.

Delete Data File with Table

If you turn this option on, Analytics automatically deletes the associated data file when you delete a table. You can use this option to quickly remove unwanted files from your hard disk, but it will prevent you from accessing the data in the future.

> **Caution**
> Be careful when turning this option on. It may be an original data file that is deleted along with the table.
>
> Data files are deleted outright. They are not sent to the Windows Recycle Bin.

You can also use the `SET DELETE_FILE` command in a script or on the command line to turn this option on or off.

If you change the setting using the **Delete Data File with Table** checkbox, the change remains in effect until you specifically change it again. If you use the `SET DELETE_FILE` command to change the setting, the change remains in effect for the duration of the Analytics session only.

Changes to this setting are recorded in the log using the following syntax:

```
SET DELETE_FILE {ON|OFF}
```

Don’t Share Table Layouts

> **Note**
> To prevent accidental deletion of field definitions, the **Don't Share Table Layouts** checkbox is selected by default.

If you deselect this option, a single table layout can be shared by multiple data files or data sources with an identical record structure. The feature works with only those Analytics operations that can output results to an Analytics table with an identical record structure – extracting, sorting, sampling, and merging – and with copying table layouts.

When sharing of table layouts is permitted, multiple source data files (e.g., Analytics data files (.fil)) or data sources that have the same record structure share a single set of field definitions. When you add a physical or computed field to a shared table layout, add a column to an associated view, or add a view, the added field, column, or view is automatically added to all the Analytics tables that use the shared table layout. When you delete a field, column, or view, it is no longer available to any of the Analytics tables that use the shared table layout.

Generally, you should maintain a separate table layout for each data file. However, sharing a single table layout can save labor if multiple data files with the same record structure require an identical set of field definitions, and any updates to the table layout will apply to all the data files. For example, extracting records from an annual transactions table into twelve separate monthly tables produces a total of thirteen tables with the same record structure. If the **Don’t Share Table Layouts** checkbox is selected, each table has its own layout. If the **Don’t Share Table Layouts** checkbox is deselected, all the tables share the original table’s layout and the layout can be managed centrally.

Deleting a shared table layout from one of the tables that uses it does not perform a global deletion. The shared table layout is still available to the other tables that use it.

Sharing does not extend beyond individual Analytics projects. If you copy a table to another project, a new table layout is created, regardless of how **Don’t Share Table Layouts** is set.

Exact Character Comparisons

Use this option to control how Analytics compares character fields, expressions, or literal values.

> **Note**
> Blank spaces are treated like characters.

If the option is off

If the option is off, Analytics uses the shorter string when comparing two strings of unequal length. The comparison starts with the leftmost characters and moves to the right.

Exact Character Comparisons is off

True	False
◦ `"AB" = "AB"` ◦ `"AB" = "ABC"` ◦ `"AB" = "ABLMNOP"` ◦ `"AB " = "AB"`	◦ `"AB" = "ZZAB"` ◦ `"AB " = "ABC"` ◦ `" AB" = "AB"`

The examples with blank spaces

- `"AB " = "AB"` is True because the shorter string (`"AB"`) is used for comparison, and the blank space in the third position is not considered.
- `"AB " = "ABC"` is False because all three characters are compared and the blank space and the "C" in the third position are not equal.
- `" AB" = "AB"` is False because the shorter string (`"AB"`) is used for comparison, and the blank space and the "A" in the first position are not equal.

If the option is on

If the option is on, comparison strings must be exactly identical to be a match. When comparing two strings of unequal length, Analytics pads the shorter string with trailing blank spaces to match the length of the longer string.

Exact Character Comparisons is on

True	False
◦ `"AB" = "AB"` ◦ `"AB " = "AB"`	◦ `"AB" = "ABC"` ◦ `"AB" = "ABLMNOP"` ◦ `"AB" = "ZZAB"` ◦ `"AB " = "ABC"` ◦ `" AB" = "AB"`

The examples with blank spaces

- `"AB " = "AB"` is True because the shorter string (`"AB"`) is padded to match the length of the longer string (`"AB "`), and `"AB "` and `"AB "` match.
- `"AB " = "ABC"` is False because all three characters are compared and the blank space and the "C" in the third position are not equal.
- `" AB" = "AB"` is False because the shorter string (`"AB"`) is padded to match the length of the longer string (`" AB"`), and `"AB "` and `" AB"` do not match.

Getting rid of blank spaces

You can use the ALLTRIM( ) function to remove leading and trailing blank spaces and ensure that only text characters and internal spaces are compared.

For example: `ALLTRIM(" AB") = ALLTRIM("AB")` is True when the values are wrapped with ALLTRIM ( ), but False otherwise.

Exact Character Comparisons and filters

The **Exact Character Comparisons** setting affects how filters work:

- **The option is off** - `Address = "PO Box"` returns all addresses that start with "PO Box"
- **The option is on** - `Address = "PO Box"` returns only those records that have the exact string "PO Box" and nothing else in the Address field

Applicability

Some Analytics operations and functions are affected by the **Exact Character Comparisons** option and some are not:

Affected	Not affected
◦ **Locate If** operation ◦ MATCH( ) function ◦ BETWEEN( ) function	◦ Join operation ◦ Relate operation ◦ FIND( ) function ◦ FINDMULTI( ) function

Log entry

Changes to this setting are recorded in the log using the following syntax:

`SET EXACT {ON|OFF}`

Display Format on Open

If you turn this option on, Analytics automatically displays the current table layout and computed field definitions when you open a new table. The results appear in the command log.

Changes to this setting are recorded in the log using the following syntax:

`SET FORMAT {ON|OFF}`

Define Flat Files Manually

With this option selected, certain screens of the **Data Definition Wizard** are skipped when you create a table from a flat file and you complete the data definition in the **Table Layout** dialog box.

Buffer Size

This option specifies the size of the data block read. The default is 33K (kilobytes), which is the recommended buffer size for most applications.

Acceptable values range from 5 to 255. Changing the buffer size may provide small performance improvements in certain environments. You should only change this setting if you are advised to do so by Support.

Changes to this setting are recorded in the log using the following syntax:

```
SET READAHEAD value
```

Sort Memory

This option specifies the maximum amount of system resources to be allocated for sorting and indexing processes. The sort memory can be any value from 0 to 2000MB (megabytes), in 20MB increments. To optimize Sort performance, set the sort memory according to the available physical memory in the system. This enables Analytics to use the necessary amount of memory to sort a table up to this maximum, if required.

If the sort memory is left as 0, Analytics uses the system resources currently available.

Sort Order

This option sets the sort sequence for character fields.

Choose the locale from the drop-down list. The default is “System Default” for the non-Unicode edition of Analytics and “Mix Languages (UCA)” for the Unicode edition. By default, Analytics sorts data in ascending order based on the byte order of each character in its character set. The **Sort Order** option affects sort order when sorting or indexing, or performing a quick sort, and when testing sequential order.

Changes to this setting are recorded in the log using the following syntax:

```
SET ORDER values
```

View options

Use the options in the **View** tab to specify how Analytics displays views.

Hide Filtered Records

When the **Hide Filtered Records** option is on, filtered tables display only those records that are included by the filter.

If this option is turned off, filtered tables continue to display all records and records excluded by the filter are highlighted.

Show Grid Lines

When this option is on, grid lines are displayed in views.

Test Column Widths

When this option is on and you change the width of a column, Analytics prompts you to save or discard your work when you close a view.

If you turn this option off and the only changes to a view are column width changes, the changes are discarded.

Show Right Edge of Page

When this option is on, Analytics displays a dotted line in the view to indicate the right margin. The margin is based on the print setup options for the report.

Display Invalid Data as Blanks or Zeros

When this option is off, Analytics accepts all invalid characters in a field.

If you turn this option on, Analytics replaces invalid character data with blanks and invalid numeric data with zeros, from the first invalid character to the end of the field. This option affects all fields except text fields: Analytics automatically replaces invalid data with blanks in text fields.

The **Display Invalid Data as Blanks or Zeros** option is most useful with variable-length data files, because it allows Analytics to correctly display them.

Changes to this setting are recorded in the log using the following syntax:

```
SET CLEAN {ON|OFF}
```

Redraw Seconds

This option displays the maximum amount of time in seconds that Analytics takes to redraw the view. If redrawing takes longer than the specified amount of time, Analytics interrupts processing and displays a message. The maximum you can specify is 100 seconds. The default time is 10 seconds.

You may need to increase the redraw time when using restrictive filters that select very few records, especially when working with very large files. When you increase the redraw time, you may have to wait longer to view the data. To reduce the waiting time, turn off the **Hide Filtered Records** option.

Global Page Title

Whatever you enter in this text box will appear, left justified, under the page number at the top of each page you print. You can choose to leave this box blank.

Authorized group licensed versions of Analytics have customer-specified text in the **Global Page Title** which cannot be modified.

Changes to this setting are recorded in the log using the following syntax:

```
SET DESIGNATION value
```

Command options

Use the options in the **Command** tab to specify how Analytics executes commands.

Autoexecute Commands

If you turn this option on, Analytics immediately executes certain commands using the selected field in the view as input. You cannot edit the command or apply a local filter. The option applies only to some command, and the selected input field must be the data type required by the command.

Automatic Output Filenames

If you turn this option on, Analytics auto-generates names for command output files. The auto-generated name contains the command name and an incremental number starting at 01. You can accept the name, or specify a more meaningful name.

Use Output Table

If you turn this option on, the **Use Output Table** checkbox, located in command dialog boxes that create tables, is selected by default. When the checkbox is selected and you execute a command, Analytics automatically closes the active table and opens the table created by the command. If you do not want the output table to open, you can deselect the checkbox before running a command.

Show Group Tests in Log

When you run a script, this option affects the display of group results as they appear in the command log. Analytics displays the group results of If, While, For, and Next tests beside the commands to which they apply. Because more than one test in a group can apply to each command, the syntax can get very long.

If you turn this option off, Analytics does not display the tests when a script is run.

Changes to this setting are recorded in the log using the following syntax:

```
SET TEST {ON|OFF}
```

Suppress XML Output for Command Results

When this option is selected, command output to screen appears as plain text rather than formatted text.

Changes to this setting are recorded in the log using the following syntax:

`SET SUPPRESSXML {ON|OFF}`

Return Matches for Null Fields

This option controls how Analytics interprets null character values. A character value is “null” when it is empty and contains no data. For example, `NAME=""` is an expression with a null character value: there is nothing between the quotation marks.

When this option is selected, Analytics interprets a null as a wildcard. For example, the view filter `NAME = ""` displays records with any value in the NAME field.

When this option is not selected, Analytics interprets a null literally as a null. For example, the view filter `NAME = ""` displays only records with no data in the NAME field.

Intervals

This option indicates the number of intervals chosen by default for a stratification or histogram. Enter a number from 1 to 255. The default is 10.

Error Limit

This option sets the default number of errors after which Analytics stops processing the Sequence or Verify commands. Enter a number from 1 to 255. The default is 10.

Retry Command Attempts

> **Note**
> Applies to Analytics scripts only. Does not apply to the Analytics user interface.

Specifies the number of times Analytics attempts to import or export data if the initial attempt is unsuccessful. Enter a number from 0 to 255. If you enter 0, no additional attempts are made after an initial failure. The default is 0.

There is no waiting period between retry attempts. Each successive attempt is made immediately after a preceding failure.

The ability to specify retry attempts is useful when connecting to databases or cloud data services, which can be temporarily unavailable.

Changes to this setting are recorded in the log using the following syntax:

`SET RETRY num`

Applies to the following commands:

Import	◦ ACCESSDATA ◦ IMPORT GRCPROJECT ◦ IMPORT GRCRESULTS ◦ REFRESH (for tables initially created using ACCESSDATA or IMPORT SAP only)
SAP (Direct Link)	◦ IMPORT SAP ◦ RETRIEVE
Export	◦ EXPORT . . . ACLGRC (export to HighBond Results)

Maximum Categories

This option specifies the maximum number of unique values that can occur in a character key field used as input for the Train command. Enter a number from 1 to 255.

Notify Settings

Retry Attempts

This option specifies the number of times the Notify operation will attempt to send an email if the initial attempt is unsuccessful. Enter a number from 0 to 255. If you enter 0, no additional attempts are made after an initial failure. The default is 5.

One possible reason for the Notify operation failing to send an email is that the email server is unavailable.

Retry Interval (seconds)

This option specifies the amount of time in seconds between additional attempts to send an email after an initial failure. Enter a number from 1 to 255. The default is 10 seconds.

Stop Script if Notify Fails

If you turn this option on, Analytics stops processing a script if the Notify operation fails. The script stops after the initial failure, or after the specified number of **Retry Attempts**, if none of the attempts are successful. The default setting is off, which allows a script to continue even if a Notify operation fails.

An invalid email recipient is not considered a failure of the Notify operation and does not cause a script to stop regardless of how **Stop Script if Notify Fails** is set.

Date and Time options

Use the options in the **Date and Time** tab to specify how dates, datetimes, and times are displayed in Analytics, and to configure several other options associated with dates and times.

Date Settings

Day, Month, Year

Use the **Day**, **Month**, and **Year** text boxes to specify the characters that represent these components of date and datetime formats. The default values are 'D' for **Day**, 'M' for **Month**, and 'Y' for **Year**, but you can specify different characters for languages other than English. The characters you specify must be uppercase, they must all be different, and 'D', 'M', and 'Y' can only be used in their default positions.

Date Display Format

This option lets you specify how Analytics displays dates, and the date portion of datetimes, in views, reports, and exported files. It also controls the format of log entry dates. You can select one of the formats in the **Date Display Format** drop-down list, or you can create your own date display format.

> **Note**
> This option has no effect on how Analytics reads dates from data sources. To specify how Analytics reads dates, use the **Data Definition Wizard**, or the **Format** field in the **Table Layout** dialog box. For more information, see "Formats of date and time source data" on page 347.

Formats in the Date Display Format drop-down list

Select This Display Format:	To Display This:
DD/MM/YY	31/12/14
DD/MM/YYYY	31/12/2014
MM/DD/YY	12/31/14
MM/DD/YYYY	12/31/2014
MMM DD, YYYY	Dec 31, 2014

Select This Display Format:	To Display This:
YYYYDDD	2014365
YYYY-MM-DD	2014-12-31

Create your own date display format

To create your own date display format, enter your choice in the **Date Display Format** text box using a combination of characters from "Date display format characters" below (assumes the default **Day**, **Month**, and **Year** format characters).

The following guidelines apply:

- The day, month, and year components can appear without spaces, or they can be separated using blank spaces or any punctuation mark.
- The components can appear in any order.
- One or two of the three components can be omitted.
- The components cannot be less than two characters long.
- You must use uppercase characters to specify the format.
- The entire date display format, including any spaces and punctuation marks, cannot exceed 12 characters.

Date display format characters

Specify This Display Format:	To Display This:
DD	Day (1 - 31)
DDD	Day (1 - 366)
MM	Month (1 - 12)
MMM	Month (Jan - Dec)
YY	Year (00 - 99)
YYYY	Year (1900 - 9999)

> **Note**
> If you specify a date display format that does not display all the available source data, quick filtering by date or datetime is disabled. For example, if you specify the format `MMM YYYY` for dates that have day, month, and year data, quick filtering on a date or datetime value in a view returns zero results.

Changes to the date display format are recorded in the log using the following syntax:

`SET DATE value`

Start of Century

Many data files use only two digits to represent the year, which means the century in which the year occurs is unspecified. The two-digit year denoting the earliest year assigned to the 20th century can vary from one set of data files to the next. This year is often called the start-of-century year or the pivot year.

The pivot year applies to two-digit years only, and does not affect data that uses four digits to represent the year. Analytics can read four-digit years from 1900 to 9999.

The default **Start of Century** setting is 40. With this setting, Analytics interprets two-digit years 40 to 99 as 1940 to 1999, and two-digit years 00 to 39 as 2000 to 2039.

To change the pivot year, enter a number from 0 to 99. For example, if you want to set 1950 as the pivot year, enter 50 in the **Start of Century** text box. The table below provides examples of different pivot years.

Start of Century setting	Year in source data	Interpreted as
00	00 to 99	1900 to 1999
40	40 to 99, 00 to 39	1940 to 1999, 2000 to 2039
50	50 to 99, 00 to 49	1950 to 1999, 2000 to 2049
99	99, 00 to 98	1999, 2000 to 2098

When working with data files that use a different pivot year from the **Start of Century** year, you can use an expression to create a computed field that correctly interprets the two-digit year or converts it to a four-digit year.

Changes to the **Start of Century** setting are recorded in the log using the following syntax:

```
SET CENTURY value
```

Aging Periods

This option sets the default aging periods for the **Age** dialog box. If you use a specific set of aging periods frequently, you can enter the set in the **Aging Periods** text box and Analytics uses the setting as the default aging periods in the **Age** dialog box. If necessary, you can still override the periods in the **Age** dialog box.

Enter the periods in days, separated by commas without spaces. You can set as many aging periods as you want.

Changes to this setting are recorded in the log using the following syntax:

```
SET PERIODS values
```

Abbreviations for Month Names

This option sets the default abbreviations for MMM-format month names. Month abbreviations must be three characters long, in the correct order starting with January, and separated by commas without spaces.

This option affects the way Analytics reads dates from a table, and displays dates in views, reports, and output files. For example, if MMM-format dates in the input file are in French, you would specify French month abbreviations:

```
Jan,Fév,Mar,Avr,Mai,...
```

Analytics would then correctly interpret the string `Fév` as the second month, and `Avr` as the fourth. If you also choose a **Date Display Format** that uses MMM to display the month, Analytics uses the abbreviations you provide to display the abbreviated month names in views, reports, and output files.

Changes to this setting are recorded in the log using the following syntax:

```
SET MONTHS values
```

Time Settings

Hour, Minute, Second

Use the **Hour**, **Minute**, and **Second** text boxes to specify the characters that represent these components of time and datetime formats. The default values are 'h' for **Hour**, 'm' for **Minute**, and 's' for **Second**, but you can specify different characters for languages other than English. The characters you specify must be lowercase, they must all be different, and 'h', 'm', and 's' can only be used in their default positions.

Time Display Format

This option lets you specify how Analytics displays times, and the time portion of datetimes, in views, reports, and exported files. You can select one of the formats in the **Time Display Format** drop-down list, or you can create your own time display format.

> **Note**
>
> This option has no effect on how Analytics reads times from data sources. To specify how Analytics reads times, use the **Data Definition Wizard**, or the **Format** field in the **Table Layout** dialog box. For more information, see "Formats of date and time source data" on page 347.

Formats in the Time Display Format drop-down list

Select This Display Format:	To Display This:	To Display This:
hh:mm	23:59	11:59
hh:mm P	11:59 P	11:59 A
hh:mm PM	11:59 PM	11:59 AM
hh:mm:ss	23:59:59	11:59:59
hh:mm:ss P	11:59:59 P	11:59:59 A
hh:mm:ss PM	11:59:59 PM	11:59:59 AM
hh:mm:ss±hh:mm	23:59:59-05:00	11:59:59-05:00

Create your own time display format

To create your own time display format, enter your choice in the **Time Display Format** text box using a combination of characters from "Time display format characters" on the next page (assumes the default **Hour**, **Minute**, and **Second** format characters).

The following guidelines apply:

- The hour, minute, and seconds components can appear without spaces, or they can be separated using blank spaces or any punctuation mark.
- The components must appear in hour, minutes, and seconds order.
- The seconds component can be omitted. The hour and minutes components cannot be omitted.
- The components must be two characters long.
- You must use lowercase characters to specify the format.
- Including the optional AM/PM indicator switches the time display from the 24-hour clock to the 12-hour clock. The AM/PM indicator can be positioned anywhere after the hour component, and can be prefaced by a space, if desired.
- The UTC offset must be prefaced with either a plus sign (+) or a minus sign (-).
- The minutes component can be omitted from the UTC offset. Do not omit the minutes component if any of the time data you are displaying contains UTC offsets that are not whole hours.
- The entire time display format, including any spaces, punctuation marks, and the plus or minus sign, cannot exceed 14 characters.

Time display format characters

Specify This Display Format:	To Display This:
hh	Hour (00 - 23)
mm	Minute (00 - 59)
ss	Second (00 - 59)
A or P	AM/PM indicator (A and P)
AM or PM	AM/PM indicator (AM and PM)
+ or -	UTC offset indicator (+ and -)

> **Note**
> If you specify a time display format that does not display all the available source data, quick filtering by datetime or time is disabled. For example, if you specify the format `hh:mm` for times that have hour, minute, and seconds data, quick filtering on a datetime or time value in a view returns zero results.

Changes to the time display format are recorded in the log using the following syntax:

`SET TIME value`

Display Times with UTC Offset as UTC

Analytics can accept local time data that includes a UTC offset (explained below), such as `10:30:15-05:00` (`-05:00` is the UTC offset). UTC is Coordinated Universal Time, the time at zero degrees longitude, and the UTC offset is a time zone indicator.

The **Display Times with UTC Offset as UTC** option lets you specify whether Analytics converts the local time to UTC without a UTC offset (the default setting), or displays UTC-based data as local time with a UTC offset. For example, here are the two different ways of displaying the same piece of UTC-based data:

- `31/12/2014 15:30:15`

  (**Display Times with UTC Offset as UTC** checked, default setting)
- `31/12/2014 10:30:15-05:00`

  (**Display Times with UTC Offset as UTC** unchecked)

When **Display Times with UTC Offset as UTC** is checked, Analytics incorporates the UTC offset in the main piece of time data, and adjusts the main piece of time data by an appropriate number of hours. In the example above, conversion to UTC increments the local time data by 5 hours.

Conversion of local time to UTC is for display purposes only, and does not affect the source data, which continues to contain the UTC offset. You can change back and forth between the two different display modes whenever you want to.

When Analytics performs calculations on local time data with a UTC offset, the UTC offset is automatically incorporated and the calculation is performed on the UTC equivalent of the local time. If **Display Times with UTC Offset as UTC** is checked, you see the actual time data that is being used in a calculation, which can make the results easier to understand. For more information, see "How UTC offsets affect datetime expressions" on page 830.

About UTC

UTC is a global time standard that has replaced Greenwich Mean Time (GMT). For most purposes, the two standards are equivalent. The final portion of UTC-based time data (for example, `-05:00`, or `+01:00`) is a **UTC offset** that indicates how far behind or ahead the local time value is compared to UTC. For example:

- `31/12/2014 10:30:15-05:00` represents December 31, 2014, 10:30:15 AM, Eastern Standard Time (North America).
- `31/12/2014 15:30:15` (UTC) represents the same point in time at zero degrees longitude.

For UTC-based datetime data, if conversion to UTC goes forward or backward across the boundary of midnight, the date is adjusted by one day.

> **Note**
> The UTC offset is also referred to as the **time zone offset**, although the two are not exactly the same. More than one time zone can have the same UTC offset.

How Analytics displays UTC-based and non-UTC time data

UTC-based time data

Source data	'Display Times with UTC Offset as UTC' selected (default setting)	'Display Times with UTC Offset as UTC' not selected
Time Display Format = hh:mm:ss		
31/12/2014 10:30:15-05:00	31/12/2014 15:30:15	31/12/2014 10:30:15-05:00
01/01/2015 00:30:15+01:00	31/12/2014 23:30:15	01/01/2015 00:30:15+01:00
Time Display Format = hh:mm:ss+hh:mm		
31/12/2014 10:30:15-05:00	31/12/2014 15:30:15+00:00	31/12/2014 10:30:15-05:00
01/01/2015 00:30:15+01:00	31/12/2014 23:30:15+00:00	01/01/2015 00:30:15+01:00

Non-UTC time data

Source data	'Display Times with UTC Offset as UTC' selected (default setting)	'Display Times with UTC Offset as UTC' not selected
Time Display Format = hh:mm:ss		
31/12/2014 10:30:15	31/12/2014 10:30:15	31/12/2014 10:30:15
Time Display Format = hh:mm:ss+hh:mm		
31/12/2014 10:30:15	31/12/2014 10:30:15+00:00	31/12/2014 10:30:15+00:00

Numeric options

Use the options in the **Numeric** tab to specify how Analytics processes and displays numeric data.

Stop on Numeric Overflow

When you select this option, Analytics stops processing when a numeric overflow occurs. If the results of mathematical operations, including intermediate calculations, exceed 22 digits, Analytics may stop processing. In the view, affected fields display ###ERR###.

When you turn this option off, Analytics continues to process, but it truncates excess digits, starting from the left, which produces inaccurate calculations. When the user attempts division by zero, Analytics substitutes a large number for the result.

Changes to this setting are recorded in the log using the following syntax:

```
SET OVERFLOW {ON|OFF}
```

Verify Data

If you turn this option on, every time you process a field while a table is open, Analytics automatically checks whether the contents of a data field correspond to the field's data type in the table layout (Character, Numeric, Datetime, Logical, etc.). Processing stops when an error occurs, unless the **Blank Invalid Data** option is also on.

If you turn this option off, Analytics does not test for data validity, therefore improving processing speed.

Changes to this setting are recorded in the log using the following syntax:

```
SET VERIFY {ON|OFF}
```

Blank Invalid Data

This option is only available when the **Verify Data** option is turned on. If you turn **Blank Invalid Data** on, Analytics automatically replaces invalid character data with blanks and invalid numeric data with zeros.

If you enable this option and process a field that contains invalid data, Analytics creates an error log in the same folder as your Analytics project files and displays the message: "Invalid data encountered in file, values zeroed. See file ERROR.LOG". To view or print the log, you can use a word processor or text editor to open it in the directory in which you store your working files. You can access the error log while in Analytics, by using the Type or Dump commands.

Changes to this setting are recorded in the log using the following syntax, where BLANK indicates that the option is selected and ON means that the **Verify Data** option is selected, but **Blank Invalid Data** is not:

```
SET VERIFY (BLANK|ON}
```

Expression Field Width

This option specifies the default display width in characters for numeric computed fields or ad hoc numeric expressions when Analytics cannot determine the maximum width. The default is 12 characters, based on Analytics's default application font.

Changes to this setting are recorded in the log using the following syntax:

```
SET WIDTH characters
```

Decimal Place Symbol

Analytics uses a period as the default decimal place character. You can change the default setting to either a comma or a space by entering the new character in the text box. Among the three separators (decimal, thousands, and list), the decimal separator must be unique.

Changes to this setting are recorded in the log as a change to the **Default Numeric Format**.

Default Numeric Format

By default, Analytics displays numbers using a numeric format that does not use thousands separators and denotes negative numbers with a leading minus sign.

You can modify the **Default Numeric Format** option to display numeric values using a different format. You can select one of the predefined formats from the drop-down list, or create a format of your own. For example, you may want to select a format that specifies a thousands separator or uses a different indicator for negative numbers.

The format you specify is an application-wide default format that applies to all numeric fields and columns that do not have field-level or column-level formatting specified. You may want to avoid specifying a format with a currency sign because all numeric fields, including those that are not currency, will have the currency sign. If required, you can format currency fields at the field level or column level. For more information about field-level and column-level numeric formatting, see "Format numeric values in a view" on page 781.

Changes to this setting are recorded in the log using the following syntax:

```
SET PICTURE format
```

Thousands Separator

Analytics uses a comma as the default thousands separator for numeric output. You can change the default setting to either a period or a space by entering the new character in the text box. The thousands separator cannot be the same as the decimal separator.

List Separator

Analytics uses a comma as the default list separator, which is used primarily to separate function parameters. You can change the default setting to either a semi-colon (;) or a space by entering the new character in the text box. The list separator cannot be the same as the decimal separator.

Print options

Use the options in the **Print** tab to specify the default print settings for reports and margin settings for printed output.

Include Report History with Reports

When this option is selected, a report history is added as the last page in the report. The report history includes the Analytics project, table, and data file names, the REPORT command used to generate the report, any table layout notes, and the table history.

Include Field Definitions in Table History

When this option is selected, field definitions for each physical data field and computed field in the table layout are added to the report. The field definitions include any field notes. This option has no effect unless the **Include Report History with Reports** option is also selected.

Include View Note in Report History

When this option is selected, any view notes associated with the active view are added to the report. This option has no effect unless the **Include Report History with Reports** option is also selected.

Margins

The **Left Margin**, **Top Margin**, **Right Margin**, and **Bottom Margin** text boxes allow you to specify the margins for all printed output. To change a value, enter the new value in the text box, or click the up and down arrows beside the text box to increase or decrease the value.

If you specify a margin that exceeds your printer's printable area, Analytics uses the maximum of your printer's printable area as the margin.

Changes to each of the individual margin settings are recorded in the log using the following syntax:

```
SET MARGIN {LEFT|RIGHT|TOP|BOTTOM} value
```

Application Font options

Use the options in the **Application Font** tab to specify the fonts used to display data in all windows, except the View tab in the display area.

Fixed-width Font

Analytics uses fixed-width fonts for information displayed in the **Table Layout**, **Script**, and **Workspace** windows. The default fixed-width font is Courier New. You can choose another font from the list box.

Proportional Font

Analytics uses proportional fonts in views and reports, and to display information such as the project file name, the table, and the record count in the status bar. The default proportional font is Arial. You can choose another font from the list box.

Language Version

Analytics allows letters, numbers, and the underscore character to be used in field names. The default Standard language version setting accommodates Western European characters for field names. The Thai setting allows Thai characters to be used in addition to English.

Script Editor Settings

Analytics allows the **Script Editor** background and font styles to be customized. You can choose the **Script Editor** background color, and colors and styles of the default text, comments, and command, parameter, and function keywords. For more information about customizing the **Script Editor**, see "Customizing the Script Editor" on page 1500.

How Analytics preferences files work

> **Note**
> Preferences file behavior was changed in version 10.0 of Analytics. This topic explains how preferences files work in version 10.0 and later.

The settings for the configurable options in Analytics – that is, the **Options** dialog box settings – are stored in a preferences file (.prf file) called `aclwin15.prf` (non-Unicode edition) or `acl15.prf` (Unicode edition).

Any changes you make in the **Options** dialog box are automatically saved to the .prf file. The changes remain in effect unless you specifically change them again.

Global versus project-specific preferences files

A single, global .prf file can govern the behavior of Analytics and all Analytics projects that you open, or you can associate different .prf files with individual Analytics projects as a way of customizing preferences on a project-by-project basis. For example, you can specify that different projects use different date display formats, or that one project deletes the source data file when you delete a table layout, and another project does not.

Importance to Analytics script writers

Understanding preferences files can be important for Analytics script writers if they provide their scripts to other Analytics users and need to control Analytics preferences settings on the other users' computers.

Global preferences file

When you install Analytics, a .prf file with the default configuration settings (**Factory** settings) is created in the following location:

- **The application data folder** – `C:\Users\< user account name >\AppData\Local\ACL` (Windows 10 location)

> **Note**
> The application data folder may be hidden by the Windows operating system. If required, enable the Windows folder option to show hidden files and folders.

The .prf file in the application data folder contains the global preference settings for Analytics. Any changes you make in the **Options** dialog box are saved to this global .prf file, unless you are using a project-specific .prf file.

The global .prf file is used when:

- you open Analytics without opening an Analytics project
- you open a project that does not have a project-specific .prf file
- you close a project without closing Analytics.

Per-user global preference settings

If more than one user account accesses Analytics on the same computer, separate .prf files exist in the application data folders for each user account, and different users can have their own global preference settings.

Automatic regeneration of global .prf file

If the global .prf file in the application data folder is deleted, renamed, or cannot be used for some other reason, a new .prf file with the default configuration settings is automatically created in the application data folder when you open Analytics.

Project-specific preferences files

If you want to customize the preference settings for one or more Analytics projects, you can manually copy the global .prf file from the application data folder to the folder containing the individual Analytics project. The copied .prf file now becomes a project-specific .prf file.

> **Caution**
> If you copy the global .prf file, be careful not to inadvertently move the file rather than copy it. If you move the file, any global preference settings you have created will be lost, and replaced by the default configuration settings.

> **Note**
> If you have different versions of Analytics installed side-by-side, make sure to copy the correct version of the .prf file.

The Analytics project file with the .acl extension and the project-specific .prf file must be in the same folder for the association between the two to take effect. When the project is open, the preference

settings specified in the project-specific .prf file are used. Any changes you make in the **Options** dialog box are saved to the project-specific .prf file rather than the global .prf file.

The benefit of project-specific .prf files

The benefit of project-specific .prf files is that you can customize preferences based on the requirements of specific projects and the scripts within a project.

If you send an Analytics project to another user, you can also send the project-specific .prf file to ensure that when the user runs any scripts within the project the results are consistent with the results on your own computer.

At the same time, because the .prf file is project-specific, it does not affect the other user's global preference settings.

Reverting to the global .prf file

At any time, you can revert to using the global .prf file for a project by deleting or renaming the project-specific .prf.

Identifying which preferences file is being used

Analytics displays the path for the currently active .prf file at the bottom of the **Options** dialog box. If the path to the application data folder is being displayed, the global .prf file is being used. If a path to an Analytics project folder is being displayed, a project-specific .prf file is being used.

Reverting to the default configuration settings

You can revert to the default configuration settings (**Factory** settings) at any time by clicking the **Factory** button at the bottom of the **Options** dialog box. Clicking **Factory** sets all options on all **Options** tabs to their default settings, not just the options on the active tab. The reversion to the default settings applies only to the currently active .prf file shown at the bottom of the **Options** dialog box.

Preferences file order of precedence

An Analytics project is open

When you open an Analytics project, a .prf file is loaded using the following order of precedence:

1. Project-specific .prf file in the folder containing the Analytics project
2. If no project-specific .prf file is found, load the global .prf file in the application data folder
3. If no global .prf file is found, automatically recreate the global .prf file in the application data folder using the default configuration settings (**Factory** settings) and load the recreated file

No Analytics project is open

When you open Analytics without opening a project, a .prf file is loaded using the following order of precedence:

1. Global .prf file in the application data folder
2. If no global .prf file is found, automatically recreate the global .prf file in the application data folder using the default configuration settings (**Factory** settings) and load the recreated file

Creating preference settings for the duration of an Analytics session

You can use the SET command to create temporary preference settings that remain in effect for the duration of an Analytics session only. For example, `SET DATE "DD MMM YYYY"` temporarily changes the date display format.

This behavior applies whether you use the SET command in the Analytics command line or in an Analytics script.

As soon as you close Analytics, the settings revert to whatever is stored in the applicable .prf file. The SET command never makes any change to a .prf file.

For more information, see "SET command" on page 1960.

Change font settings for views and reports

Use the **Select View Fonts** dialog box to customize the font settings used to display text in views and reports. You can either customize the font used for each area of the view or report individually, or you can change the font settings for all areas at once.

Any changes you make to the font settings affect only the current view, and reports based on the view. The settings are added to the Analytics project and persist between Analytics sessions. If you create a new view, it replicates the font settings of whatever view is currently open. If required, you can change the font settings for the new view immediately after creating it.

1. Do one of the following:
   - Click **Change Font**  in the display area.
   - Right-click in the view and select **Select View Fonts**.
2. In the **Select View Fonts** dialog box, click one of the following buttons:
   - **All** – The selected font settings are applied to all areas of the view and report unless more specific settings are applied to individual areas.
   - **Header** – The font settings for the report header.
   - **Titles** – The font settings for column headers.
   - **Data** – The font settings for view and report data.
   - **Totals** – The font settings for report subtotals.
   - **Footer** – The font settings for the report footer.
3. In the **Font** dialog box, make any necessary changes to the font settings and click **OK**.
4. Repeat steps 2 and 3 if you want to change the font settings for any additional areas.
5. Click **OK**.

Change font size in views

You can temporarily increase or decrease the font size in the views in an Analytics project. The change in size affects all views in the project, and persists until you reset the size or exit Analytics.

1. Click into an open view.
2. Do one of the following:
   - Press **Ctrl++** repeatedly to increase the font size (Ctrl + the Plus key on the number pad).
   - Press **Ctrl+-** repeatedly to decrease the font size (Ctrl + the Minus key on the number pad).
   - Press **Ctrl+0** to reset the font size to the default size (Ctrl + 0 on the number pad).

> **Note**
> You must use the Plus, Minus, and 0 keys on the number pad, not on the main keyboard. On laptops, press **Fn+Ctrl+** the appropriate key on the number pad.

Customize the Analytics toolbar

Customize the Analytics toolbar by adding buttons for the features that you use most often, and removing those you use less frequently. You can also reorder the buttons in the toolbar, and add separators to group related buttons.

1. If required, select **Window > Tool bar** to display the toolbar.
2. Double-click an empty spot on the toolbar to open the **Customize Toolbar** dialog box.
3. Complete any of the following steps:
   - Select a button in the **Available toolbar buttons** list and click **Add** to add it to the toolbar.
   - Select a button in the **Current toolbar buttons** list and click **Remove** to remove it from the toolbar.
   - Select the **Separator** button in the **Available toolbar buttons** list and click **Add** to insert a vertical line to visually group related buttons. You can add as many separators as you need.
   - Select a button in the **Current toolbar buttons** list and click **Move Up** or **Move Down** to change the location of a button. The order of the buttons from top to bottom corresponds to their location from left to right on the toolbar.

   The toolbar dynamically updates as you make changes.
4. If required, you can click **Reset** to reverse all the changes you have just made to the toolbar.

   > **Note**
   > Once you click **Close** the changes are saved and **Reset** does not reverse them. You can revert to the default toolbar settings by selecting **Tools > Options > Factory**.

5. Click **Close** to save your changes.

Adding custom items to the Analytics main menu

You can add custom items such as your own Analytics scripts, and frequently used commands, to the Analytics main menu. The custom items appear under the **Applications** menu. This capability is especially useful if you create scripts for others to use, and you want a single, easy-to-use location for accessing the scripts.

Project-level or global access

You can create custom menu items that are restricted to individual Analytics projects, or you can make the items available globally, whenever Analytics is open.

- **project-level access** – locate the text file (*.mnu) containing the custom menu items in the same folder as the Analytics project (*.acl)
- **global access** – locate the text file (*.mnu) containing the custom menu items in the same folder as the Analytics executable (ACLWin.exe)

You can also combine approaches, and create both project-level and global custom menu items.

> **Note**
> If you want other Analytics users to have the custom menu items, give them the *.mnu file with instructions about where to locate the file.

The configurable menu file (*.mnu)

You use one or more text files with a .mnu file extension to build your custom menu items.

Sub-menu entries

Each *.mnu file creates a separate sub-menu entry under the **Applications** menu. For example, `Account scripts.mnu` creates the **Account scripts** sub-menu entry and this menu structure: **Applications > Account scripts**.

Sub-menu entries appear in alphanumeric order on the **Applications** menu.

Custom menu items

Contained within each sub-menu entry are the individual custom menu items. For example, you might have two items that run the following scripts:

- Accounts Payable Analysis

  (**Applications > Account scripts > Accounts Payable Analysis**)
- Accounts Receivable Analysis

  (**Applications > Account scripts > Accounts Receivable Analysis**)

Custom menu items appear on sub-menus in the order that the items are listed in the *.mnu file.

Using one or more *.mnu files, you can build multi-level, cascading sub-menus to meet your requirements.

> **Tip**
> Users can become disoriented by too many sub-menu levels. A best practice is to limit sub-menu levels to three.

Maximum number of custom menu items

The **Applications** menu can contain a maximum of 179 custom menu items. The maximum applies to the total number of custom menu items across all sub-menu levels and all *.mnu files.

Menu file syntax

> **Note**
> The required syntax in the menu file must be specified precisely. Even a single extra space can cause Analytics to ignore the menu file and the custom menu items do not appear.
>
> Follow the requirements below exactly.

> **Tip**
> Create or edit your menu files in a text editor such as Notepad++ with all non-printing characters displayed so that you can see exactly what characters are contained in the file.
>
> Use a monospaced or fixed-width font so that individual characters align vertically.

A sample menu file, `Template.mnu`, is included in the `Sample Data Files` folder installed with Analytics.

- `Template.mnu` creates the sub-menu entry **Template** in the **Applications** menu in `Sample Project.acl`, and in the three other sample Analytics projects contained in the `Sample Data Files` folder.
- The **Template** sub-menu entry contains six custom menu items at the first level.
- One of the first-level custom menu items, **Margin Analysis**, contains four custom menu items at the second level.
- Most of the custom menu items in `Template.mnu` are simply placeholders to illustrate the concept of menu files.

The content of `Template.mnu` is reproduced below, with accompanying syntax requirements.

Content of Template.mnu

```
MAIN MENU                          6                                        .
Margins Analysis                   8 menu_def                               .
Inventory Analysis                 PAUSE 'SAMPLE INVENTORY ANALYSIS BATCH'  .
Accounts Payable Analysis          PAUSE 'LAUNCH YOUR A/P BATCH(ES)'        .
Accounts Receivable Analysis       PAUSE 'DO A/R BATCH(ES) HERE'            .
Internal Audit Functions           PAUSE 'SAMPLE INTERNAL AUDIT PROCESSES'  .
Quit ACL                           QUIT                                     .
                                                                            .
MARGINS ANALYSIS                   4                                        .
Exception Listing                  PAUSE 'DO Batch  where margin<=0'        .
High Margin Products               PAUSE 'Sample Batch top 5 margins'       .
Low Margin Products                PAUSE 'Calculate lowest 5 margins'       .
Margin Statistics                  STATISTICS                               .
```

Menu file syntax requirements

Property	Requirement
Line length	Each line of the menu file must be exactly the same length. Although not required, it is good practice to use a period (.) to visually mark the end of each line, immediately before the line break.
Line numbering	The lines in the menu file are counted from zero (0). Keep this numbering scheme in mind whenever you specify line number references in the menu file syntax. If the text editor you are using displays and counts line numbers beginning at 1, you need to decrement the text editor line number by 1 when you specify menu file line number references. In the example above, the **Margins Analysis** menu item appears on line 1, and the **MARGINS ANALYSIS** sub-menu syntax block appears on lines 8 through 12.
Blank lines	Blank lines can appear between syntax blocks but not within syntax blocks. Blank lines, composed of space characters, must be the same length as the other lines in the menu file. Although not required, one or more blank lines between syntax blocks provides visual separation in the menu file.
Syntax blocks	Syntax blocks define each group of custom menu items. You can use multiple syntax blocks to create multiple menu levels. ◦ The left side of the block contains the names of the menu items, one per line. These are the names that appear on the menu in Analytics. ◦ Names can be a maximum of 35 characters.

Property	Requirement
	◦ The right side of the block contains either an ACLScript command or a line reference to a lower-level block of syntax. ◦ Lines on the right side of the block must all start at character position 37. ◦ Use only space characters to align text elements. Do not use tab characters. **Note** Even one tab character in a menu file will cause the file to be ignored. Use a text editor that can display tab characters so you can check for their presence.
Block heading	Each syntax block begins with a single heading line. The left side of the line contains the block identifier in uppercase and the right side contains the number of lines within the block. In the example above, line 0 contains the block identifier **MAIN MENU** and specifies that there are 6 lines in the block. The heading line is not counted. Block identifiers are optional. They keep syntax blocks organized within the menu file. They do not appear anywhere in the Analytics **Applications** menu structure. If you omit a block identifier, the specified number of lines in the block must still start at character position 37.
Reference to a lower-level block of syntax	A reference from a menu item to a lower-level block of syntax takes the form `num menu_def`. `num` specifies the line number on which the lower-level block of syntax starts - that is, the heading line of the lower-level syntax block. In the example above, line 1 contains the **Margins Analysis** menu item, which refers to the line on which the **MARGINS ANALYSIS** lower-level syntax block starts (`8 menu_def`).
Custom menu items	Custom menu items can specify: ◦ any valid ACLScript command ◦ a line reference to a lower-level block of syntax To create a custom menu item that runs an Analytics script, specify `DO SCRIPT script_name`. For example: `Calculate Median Value script  DO SCRIPT Calculate_Median_Value` **Note** The script must be included in the Analytics project in which the custom menu item appears. Short commands can be entered directly in the .mnu file. Longer commands with multiple parameters should be saved in a script, which can be referenced using the `DO SCRIPT` command.

Create or edit a menu file

The easiest way to create a menu file is to copy the Analytics sample menu file (`Template.mnu`) and then modify it.

Keep the following points in mind when you edit a menu file that is already in use:

- Before you edit any menu file, make a backup copy of it.
- If you add or remove lines, make sure to adjust any line number references appropriately.
- Wherever possible, add new items at the end of the menu file in order to maintain the existing line references.

1. Copy `Template.mnu` from the Analytics `Sample Data Files` folder to a working location.

   > **Caution**
   > Do not edit the original template file. If you run into problems you can recopy the original file and start again.

2. Rename the copied file to something appropriate.

   The name you give the file becomes the name of the sub-menu entry on the Analytics **Applications** menu.

   > **Note**
   > If you are creating a menu file from scratch, change the file extension to `.mnu`.

3. Open the renamed file in a text editor such as Notepad++ and edit it to build sub-menus and custom menu items.

   Follow the "Menu file syntax requirements" on page 155 above exactly.

4. Do one of the following:
   - Save the file in the folder containing the Analytics project in which you want the custom menu items to appear.
   - Save the file in the Analytics executable folder to make the custom menu items available in all Analytics projects opened on the computer.

   > **Tip**
   > You can create both project-level and global menu files, if required.

5. Close and reopen Analytics to refresh the **Applications** menu.

   The sub-menu entry and the custom menu items should now be available in the **Applications** menu.

   If the sub-menu and the custom menu items do not appear, carefully check the content of your menu file against the syntax requirements above. Make all non-printable characters visible in the text editor you are using. An extra space at the end of a line, or a tab character anywhere in the menu file, causes the file to be ignored.

Running commands from the Analytics command line

Most of the functionality in Analytics that is completed by selecting options from menus and entering the required information in dialog boxes can also be completed by running commands from the command line.

Analytics includes a language called ACLScript that is used throughout the application to process commands and record analysis steps. For example, when you select **Analyze > Count** from the main menu and complete the required information in the **Count** dialog box, Analytics automatically converts the information you entered to a command statement that is used to run the command and record the action in the log.

Guidelines for using the command line

- You can only enter one command at a time.
- Commands are not case-sensitive. You can use uppercase or lowercase characters.
- Commands can include a set of required and optional parameters.
- When the text for a command is long, Analytics expands the **Command Line** text box and wraps lines to display the entire command. Click outside the text box to collapse it and click inside the text box to display the entire command again.
- You can abbreviate commands, functions, and keywords in ACLScript. The abbreviation must include the leading characters of the command, function, or keyword. The abbreviation can be as short as you like, provided it uniquely identifies the term. In most cases you will need to enter at least the first three characters.

For detailed information about the syntax required for each ACLScript command, see "Commands overview" on page 1526.

Steps

1. If the **Command Line** text box is not visible, select **Window > Command Line**.
2. Enter the command text using one of the following methods:
   - Type in the command using the required syntax.
   - Click an entry in the **Log** tab in the **Navigator** to add the command to the command line. You can run the command as is, or edit it before running the command.

- Copy the command syntax from an existing Analytics script, or other text file, and paste it in the **Command Line** text box. You can run the command as is, or edit it before running the command.

3. Optional. If the command has a dialog box associated with it in the Analytics user interface, click **Edit Command**  to display the associated dialog box, which you can use to modify the parameter settings for the command.
4. Click **Run**  or press **Enter** to run the command.

   The **Run**, **Clear Entry**, and **Edit Command** options are also available by right-clicking in the **Command Line** text box.

Printing display area information

You can print the information displayed in the display area in the active View tab, Script Editor tab, Results tab, or Workspace Editor tab. If you print the information in the View tab, the default report settings for the view are used.

To print information in the display area:

1. Open the view, workspace, or script you want to print.
2. Select **File > Print**.
3. If the information in the display area is more than one page long, you have the option to specify a page range. To specify a range, select **Pages** and enter the start page number and end page number to print.
4. If you want to change the printer, or any printer properties (paper size, page orientation, etc.), click **Setup** and make any necessary changes in the **Page Setup** dialog box and click **OK**.
5. Click **Print**.

Send email notifications from Analytics

You can send email notifications, which can optionally include file attachments, directly from Analytics. This functionality is primarily used in scripts to notify users when processing has completed, or an error occurs.

You can also send messages from Analytics if you know the required information about your mail server configuration and the mail server security configuration does not prevent it.

> **Note**
>
> The email notification feature can be used with any mail server that supports SMTP (Simple Mail Transfer Protocol), which is used by Microsoft Exchange and many other mail servers. Email notification uses port 25, so this port must be open on the mail server or the command fails.
>
> If email notification fails with an error message, contact your IT department to find out if port 25 is blocked on your network.

1. Select **Tools > Notify by Email**.
2. Complete the following information:
   - **Sender** - Enter the email address to send the message from.
   - **Password** - Enter the email account password.
   - **Mailbox Path** - Enter the hostname or IP address of your SMTP mail server.

     If you are using a local mail system, enter the path to a local mailbox or click **Browse** to open the **Browse for Folder** dialog box.
   - **To** - Enter the email addresses of recipients. Separate the names and addresses of multiple recipients with commas.

     > **Note**
     >
     > Enter a maximum of 1020 characters.

   - **Cc** - Optional. Enter the email addresses of "carbon copy" recipients. Separate the names and addresses of multiple recipients with commas.

     > **Note**
     >
     > Enter a maximum of 1000 characters.

   - **Bcc** - Optional. Enter the email addresses of "blind carbon copy" recipients. Separate the names and addresses of multiple recipients with commas.
   - **Subject** - Enter the text of the subject line.
   - **Text** - Enter the text of the message.

- **Attachment** - If you want to include an attachment, specify the path and filename of the file, or click **Browse** to open the **Select File** dialog box.

3. Click **OK**.

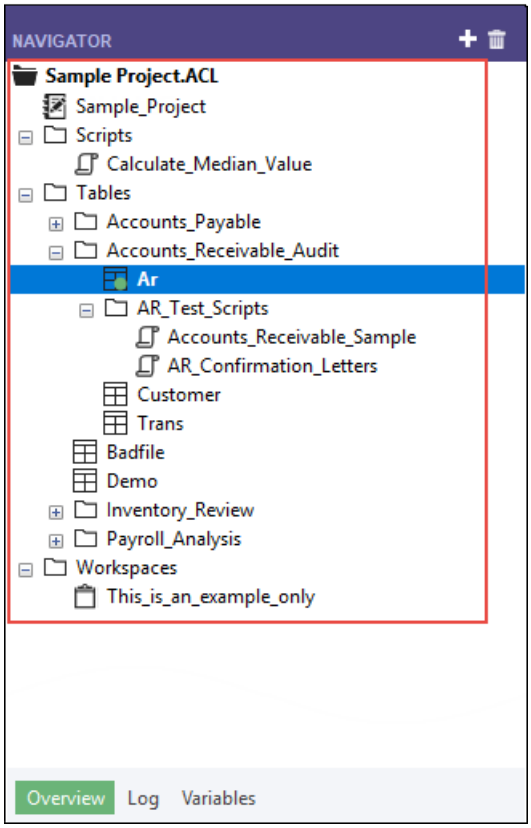
Analytics projects

Analytics projects are the highest level of organization in Analytics, and they store the information associated with a data analysis project.

The main Analytics project file (.ACL) stores most of the project information. A set of additional files store particular types of information about the project, such as the log or indexes. Data is stored outside the project in native Analytics data files, or in a database.

The Analytics project you are currently working with is displayed in the **Overview** tab in the **Navigator**. The contents of the log are displayed in the **Log** tab. Only one project can be open at a time.

`Sample Project.ACL` appears below in the **Navigator**.



Navigator: Overview tab

Analytics projects contain several different item types. You can view and work with these items in the **Navigator**.

The table below lists the item types that can appear in the **Navigator** treeview.

Icon	Item type	Description
	Table	An Analytics table, which consists of two parts: a table layout and an associated data source. The table layout contains information about how to display the data, such as record length and field names. The data source is a file or data set (e.g., database table) that contains the content of the table. The data source exists outside the Analytics project.
	Server Table	A table with a table layout that resides locally in Analytics, and an associated data source on a server. The table layout connects to the data source using a database profile and/or server profile.
	Script	A series of ACLScript commands that can be run from within the Analytics project.
	Server Script	An ACLScript file (.aclscript, formerly .bat) that is located on a server.
	Workspace	An Analytics project item that contains one or more field definitions that have been saved for reuse with other tables.
	Project	The top-level entry in the treeview is the Analytics project. Projects are stored in physical files with a .ACL file extension.
	Log	A record of the commands issued when working with the Analytics project.
	Folder	A folder inside the Analytics project. These folders exist only in the Analytics project file (.ACL). They are not physically created as Windows folders.

Navigator: Log tab

The table below lists the types of entries that can appear in the **Log** tab in the **Navigator**.

Icon	Entry type	Description
	Group	A group of log sessions within a specific date range.
	Session entry	Individual sessions indicated by date and time. Sessions are created whenever you open the project, or when you create a session manually.
	Command successful	Identifies a command completed successfully.
	Command failed	Identifies a command that failed.

Analytics project files

The following file types are used to record information for Analytics projects. When you back up or archive a project, you must ensure that you copy all of the files to restore the complete project.

File type (extension)	Description
Analytics Project file (.ACL/.acl)	The Analytics project file is where all of the critical information for your data analysis project is stored: ◦ table layout and view definitions ◦ scripts ◦ project folders ◦ command syntax that updates tables using the **Refresh from Source** command. ◦ table history ◦ workspaces
Analytics Project auto-save file (.ac)	A temporary autosave file is created each time the project is opened. The purpose of the file is to record all unsaved changes to the Analytics project, so that the changes can be recovered if Analytics closes unexpectedly. If the project is saved and closed normally, the .ac file is deleted, otherwise you are prompted to restore your project from this file.
Analytics data file (.fil)	In many cases, when you define an Analytics table from a data source, the data is copied from the data source into a new Analytics data file with a .fil file extension. For a list of data source types that copy data to .fil files, see "Data sources you can access with Analytics" on page 227.
Log file (.log)	The log file records all commands executed by Analytics while the project is open. The default log is assigned the same name as the Analytics project file, with a .log extension. If necessary, you can specify a custom log file name.
Log index file (.lix)	An index file used to associate log entries with particular sessions. Sessions are created each time you open a project, and can also be created manually at any time.
Index file (.inx)	An index file is created when you index an Analytics table. The file name is the same as the name of the index in Analytics, with an .inx extension. An index file is also created when you use the **Quick Sort Ascending** or **Quick Sort Descending** commands on a table. The filename for indexes created by quick sort commands is ACLQSTMP.inx

Additional Analytics file types

Four additional file types can be created from an Analytics project, or imported into an Analytics project.

These file types are not required by the project, however if they exist, you may want to include them in any backup process.

File type (extension)	Description
Table layout file (.layout)	An external copy of an Analytics table layout.
View file (.rpt)	An external copy of an Analytics view.
Analytics script file (.aclscript)	An external copy of an Analytics script or analytic.
Workspace file (.wsp)	An external copy of an Analytics workspace.

Analytics analysis app files

Analysis apps are bundled sets of analytic scripts. Analytic scripts are regular scripts written using the ACLScript language, with the addition of an analytic header that allows the script to run in AX Client, AX Web Client, or the Analysis App window.

Analytic scripts are created and tested in Analytics and in order to be run in the Analysis App window they must be packaged and saved outside Analytics as an analysis app package with an **.aclapp** file extension.

When the .aclapp file is opened in the Analysis App window it is automatically installed as an analysis app file with an **.aclx** file extension.

For more information, see "Working with analysis apps" on page 2642.

Working with Analytics projects

Analytics projects provide a way to group and organize all the tables and processing associated with a data analysis project. You can create as many Analytics projects as you need. The data analysis completed in Analytics is recorded in the command log in the Analytics project from the moment a project is created.

Create a new Analytics project

When you create a new project, a best practice to create a new Windows folder for the project, and maintain a one-to-one relation between Analytics projects and Windows folders. Use the Windows folder, and subfolders as required, to store:

- The Analytics project file (.acl) and all associated files, such as index files (.inx)
- Source data files
- Analytics data files (.fil)
- Results files produced from analysis performed in Analytics

You can create a new project from within Analytics, or from the ACL for Windows main screen.

Show me how

> **Note**
>
> The combined length of the Analytics project path and the project name, including the file extension (.acl), cannot exceed 259 characters.

Create a new project from Analytics

1. From the Analytics main menu, select **File > New > Project**.
2. In the **Save New Project As** dialog box, select a folder to save the project in, enter a file name, and click **Save**.

   The new project is created and the Data Definition Wizard opens. You can proceed through the wizard to create a new Analytics table in the new project, or click **Cancel** if you do not want to create a table.

Create a new project from ACL for Windows

1. In the **ACL for Windows** main screen, click **New Analytic Project**.
2. In the **Save As** dialog box, select a folder to save the project in, enter a file name, and click **Save**.

   The new project is created and opened in Analytics.

Open an existing Analytics project

When you open an existing Analytics project a new session is created, and all commands processed by Analytics are recorded in the log.

You can open an existing project from within Analytics, or from the ACL for Windows main screen.

Show me how

Open an existing project from Analytics

1. From the Analytics main menu, select **File > Open Project**.
2. In the **Project** dialog box, navigate to an Analytics project file (.acl), select the file, and click **Open**.

   The project opens in Analytics. If another project is open in Analytics, you will be prompted to save any changes to that project, and it will be closed before the selected project is opened.

Open an existing project from ACL for Windows

1. In the **ACL for Windows** main screen, click **Open Analytic Project**, or select an Analytics project (.acl) under **Recent Analytics Files**.
2. If you clicked **Open Analytic Project**, navigate to an Analytics project file (.acl), select the file, and click **Open**.

   The project opens in Analytics.

Save an Analytics project

The first time you save an Analytics project, you need to specify the filename and location. You can save the latest version of the project in the same location using the **Save Project** menu command. If you want to save a copy of the project, use the **Save Project As** menu command and choose a different filename and location.

Show me how

1. Do one of the following:
   - If you want to save the current version of the project select **File > Save Project**.
   - If you want to save the current version of the project with a different name, select **File > Save Project As**, and then enter the new filename and choose the location in the **Save Project As** dialog box and click **Save**.

     If you select this option, the project is saved with the new name, but the new project is not opened in Analytics.
2. If there are any project items that have been modified since the project was last saved, you will be prompted to save them. Click **OK** in the confirmation dialog box(es).

View Analytics project properties

You can view a number of properties associated with an Analytics project, and quickly navigate to the folder containing the project.

Show me how

1. Right-click the Analytics project in the **Overview** tab in the **Navigator**.

   The Analytics project is the top-level entry in the treeview.

2. Select **Properties**.
3. In the **Project Properties** dialog box, click on the following tabs to view or modify project properties:
   - **General** – This tab displays the basic properties of the project file: the file name, the file location, the last modification date and time, and the physical size of the project file.

     You can click **Open file location** to navigate directly to the folder containing the Analytics project file (.acl).
   - **Notes** – This tab displays any notes associated with the project. You can modify existing notes or add new notes. For more information, see "Add or edit Analytics project notes" on page 177.
   - **Views** – This tab displays all the views in the project, using the form *view name*[*table layout name*]. You can maintain the views in the project from this tab. For more information, see "Working with views" on page 770.
4. Click **OK** to close the dialog box and save any changes you have made.

Copy a project item from another Analytics project

You can copy Analytics project items into your current project from any Analytics project on your computer or on an accessible network drive.

You can copy multiple project items simultaneously if they are of the same type – for example, you could copy multiple scripts simultaneously. If you want to copy items of different types – for example, scripts and tables layouts – you need to repeat the procedure below for each item type.

If you want to import a project item that exists as a separate file outside an Analytics project, see "Import a project item" on the next page.

Show me how

1. Open the project that will contain the copied item or items.
2. In the **Overview** tab of the **Navigator**, right-click the Analytics project entry, or a project folder, and select **Copy from another Project > [Project Item Type]** where Project Item Type is one of the following options:
   - Table

- View
- Script
- Workspace

> **Note**
> When you copy a table, you are copying the table layout only, not the source data file (.fil).

3. In the **Locate Project File** dialog box, locate and select the Analytics project you want to copy the project items from and click **Open**.
4. In the **Import** dialog box, complete any of the following tasks to add project items to the **To *project_name*** list:
   - Double-click an individual project item.
   - **Ctrl+click** multiple project items and then click the right-arrow button.
   - Click **Add All** to add all the project items.

   You can remove project items from the **To *project_name*** list by double-clicking an individual project item, by using **Ctrl+click** to select multiple project items and then clicking the left-arrow button, or by clicking **Clear All**.
5. Click **OK** to copy the project item or items into the current project.

   If an item with the same name already exists in the project, the copied item is given an incrementing numeric suffix.

Import a project item

You can import Analytics project items that exist as separate files outside an Analytics project – for example, an Analytics script saved as an .aclscript file, or a table layout saved as a .layout file. You can import only one project item at a time.

If you want to import a project item from another Analytics project, see "Copy a project item from another Analytics project" on the previous page.

Show me how

1. In the **Overview** tab of the **Navigator**, right-click the Analytics project entry and select **Import Project Item > [Project Item Type]** where Project Item Type is one of the following options:
   - Table
   - View
   - Script
   - Workspace
2. In the **Project** dialog box, locate and select the appropriate file type and click **Open**.

   File types and project items correspond as follows:

File extension	Project item
.layout	table layout

File extension	Project item
.rpt	view
.aclscript	script
.wsp	workspace

3. Click **OK** in the confirmation dialog box.

   The project item is imported into the current project. If an item of the same type, with the same name, already exists in the project, the imported item is given an incrementing numeric suffix.

Export a project item

You can export Analytics project items as separate files saved outside the Analytics project – for example, an Analytics script can be saved as an .aclscript file, and a table layout can be saved as a .layout file. Project items exported as separate files can later be imported into any Analytics project. You can export only one project item at a time.

Project items and file types correspond as follows:

Project item	File extension
table layout	.layout
view	.rpt
script	.aclscript
workspace	.wsp

Show me how

> **Note**
>
> Limit the item name to 64 alphanumeric characters, not including the file extension, to ensure that the name is not truncated when the item is imported back into Analytics.
>
> The name can include the underscore character (_), but do not use any other special characters, or any spaces, or start the name with a number. Special characters, spaces, and a leading number are all replaced by the underscore character when the item is imported.

Export a table layout, a script, or a workspace

1. In the **Overview** tab of the **Navigator**, right-click the item and select **Export Project Item**.
2. In the **Save As** dialog box, choose a location to save the item, rename the item if required, and click **Save**.

3. Click **OK** in the confirmation dialog box.

   The project item is exported to the location you specified.

Export a view

1. Open the table associated with the view.
2. In the **Overview** tab of the **Navigator**, right-click the table and select **Properties > Views**.
3. Select the view, and click **Export**.
4. In the **Save As** dialog box, choose a location to save the view, rename the view if required, and click **Save**.
5. Click **OK** in the confirmation dialog box.

   The view is exported to the location you specified.

Using the Analytics command log

Analytics includes a powerful logging feature that serves two main purposes:

- **Tracking analysis**

  The log records the exact sequence of commands executed during each Analytics session, and saves them as part of the Analytics project. Recording the sequence of commands ensures that audit steps can be verified and replicated whenever necessary.

- **Recording Analytics command syntax**

  The log records the ACLScript syntax used to execute each command. Having access to the exact syntax allows you to:

  - easily rerun commands from the log instead of manually recreating them through the user interface
  - create scripts based on selected log entries
  - view and learn the ACLScript syntax associated with commands accessed through the user interface

Add sessions to the log

A new session is automatically created in the log each time you open an Analytics project. Each session includes the date and a timestamp indicating when the session started.

You can also manually add a session to the log whenever you want to create a group of log entries. For example, you might want to add a distinct session for each stage of audit analysis you perform.

When you manually add a session you have the option of including a session name.

1. Right-click in the **Log** tab and select **Add New Session**.
2. (Optional) In the **Session Name** text box, type a name to identify the session.

   Session names can be a maximum of 30 characters.

3. Click **OK**.

Add comments to the log

You can manually add comments to the Analytics command log. Use comments to document steps in your analysis, and to add explanations or descriptions that are relevant to the audit project. When you add a comment it is added as the last item in the log.

1. Select **Tools > Comment**.
2. Enter the text of the comment in the **Comment** text box and click **OK**.

   Single-line comments are displayed in the log treeview. To view multiline comments, double-click the comment entry in the treeview.

Search the log

You can search the log for particular command names or strings in log entries and session entries.

1. Click the **Log** tab in the **Navigator**.
2. Right-click the log entry where you want to start the search and click **Find**.
3. In the **Find** dialog box, in the **Find what** field, enter the search string.
4. Specify any of the **Find** options, as required:

Option	Description
Match whole word only	Only exact matches are found. For example, a search for "SET L" does not match "SET LEARN" if this option is selected, but it would otherwise.
Match case	Only matches with exactly matching upper and lower case are found. For example, a search for "Comment" does not match "COMMENT" if this option is selected, but it would otherwise.
Up, Down	Specifies the search direction.

5. Click **Find Next**.

   If a match is found, the first log entry that contains the search string is highlighted. Click **Find Next** to navigate to the next match.

Copy log entries

You can copy log entries to the clipboard, and then paste the entries into the **Script Editor** or **Workspace Editor** in Analytics, or into another application. Copying entries from the log is an alternative to exporting the entries to a new file or a script.

1. Click the **Log** tab in the **Navigator**.
2. Select the checkbox beside each log entry that you want to copy.

   You can select:

   - individual entries
   - log sessions
   - date ranges
   - the entire log

   If you select a log session or date range, all sub-entries are also automatically selected.

3. Right-click in the **Log** tab and select **Copy**.

   The log entries are copied to the Windows clipboard.

4. Paste the log entries into a destination.

Export log entries

You can export Analytics log entries, or the entire contents of the log, to an external file, or to a new Analytics script in the current project.

1. Click the **Log** tab in the **Navigator**.
2. Select the checkbox beside each log item that you want to export.

   You can select:

   - individual entries
   - log sessions
   - date ranges
   - the entire log

   If you select a log session or date range, all sub-entries are also automatically selected.

3. Right-click in the **Log** tab and select **Save Selected Items > [Export Type]** where Export Type is one of the following options:

HTML	an HTML file (.htm)
Log File	a new Analytics log file (.log)
Script	a new Analytics script in the current project
WordPad	a temporary new file in WordPad
Text	a text file (.txt)

4. Specify a file name or a script name and click **Save** or **OK**.

   To save a temporary WordPad file, use **Save as** in WordPad.

Delete log entries

You can delete the following entries from the log:

- individual entries
- log sessions
- date ranges
- the entire log

If you select a log session or date range, all sub-entries are also automatically selected.

When you delete a portion of the log, or the entire log, the following comment is automatically inserted at the point of deletion: **A range of the Log has been deleted**.

1. Click the **Log** tab in the **Navigator**.
2. To delete all entries in the log:
   a. Right-click in the **Log** tab and select **Delete Entire Log**.
   b. Click **OK** in the confirmation dialog box.
3. To delete individual entries:
   a. Select the checkbox beside each entry, session, or date range you want to delete.
   b. Right-click and select **Delete Selected Items**.
   c. Click **OK** in the confirmation dialog box.

Rerun commands from the log

You can select any single-line command in the log and rerun it from the command line.

You can also select and rerun the multiline version of the DEFINE FIELD...COMPUTED command. Other multiline commands, such as GROUP, cannot be rerun from the command line and can only be run in scripts.

When you rerun commands you can run them as is, or modify them before running them.

1. Click the **Log** tab in the **Navigator**.
2. Click the log entry with the command you want to rerun.

   The command prefills the **Command Line** near the top of the Analytics interface, just below the toolbar.

   > **Note**
   > If the **Command Line** isn't visible, select **Window > Command Line** from the Analytics main menu.

3. If required, edit the command in the **Command Line** text box.
4. Click **Run**  .

Using notes in Analytics projects

There are several different types of notes you can create in Analytics to record information about specific project items. Notes are particularly useful for recording details of a process that is repeated on a regular basis. When more than one person will be working with an Analytics project, notes make it easier for others to understand the procedures you design.

You can add notes to the following items:

- Analytics project
- table layout
- view
- record
- field
- script
- workspace

Add or edit Analytics project notes

You can add a note to an Analytics project to record any details about the project that you want to keep for future reference, or document for other users. You can edit the content of a project note at any time.

Show me how

1. Right-click the Analytics project in the **Overview** tab in the **Navigator**.

   The Analytics project is the top-level entry in the treeview.

2. Select **Properties**.
3. In the **Project Properties** dialog box, click the **Notes** tab.
4. Enter a new note or edit the existing note.

   To delete the note, delete all the text.

5. Click **OK** to close the dialog box and save your changes.

Add or edit table layout notes

You can add a note to a table layout to record information such as when or how the data source was accessed, the computed fields that are defined, or the analysis steps that need to be completed on the table. You can add a table layout note in either the **Overview** tab of the **Navigator**, or the **Table Layout** dialog box. You can edit the content of a table layout note at any time.

If you maintain table layout notes in the **Navigator**, you do not need to open the table to add, edit, delete, or read the note.

Table layout notes appear in printed Analytics reports if **Include Report History with Reports** is selected in the **Options** dialog box (the default setting). For more information, see "Print options" on page 144.

Show me how

Add or edit a note from the Overview tab

1. Right-click the table in the **Overview** tab in the **Navigator** and select **Properties**.
2. In the **Table Properties** dialog box, click the **Notes** tab.
3. Enter a new note or edit the existing note.

   To delete the note, delete all the text.
4. Click **OK** to close the dialog box and save your changes.

Add or edit a note from the Table Layout dialog box

1. Select **Edit > Table Layout**.
2. Click the **Table Layout Options** tab.
3. Click **Edit Table Layout Note**  .
4. Enter a new note or edit the existing note.

   To delete the note, delete all the text.
5. Click **Close**  .

   The note is automatically saved.
6. Click **Close**  to exit the **Table Layout** dialog box.

Add or edit view notes

You can add a note to a view to provide additional information about the view. The note is specific to that particular view, and is copied to any new views created from that view. You can edit the content of a view note at any time.

View notes appear in printed Analytics reports if you select **Include View Note in Report History** in the **Options** dialog box. For more information, see "Print options" on page 144.

Show me how

1. At the bottom of the View tab, right-click the button for the view you want to add a note to and select **Properties**.
2. In the **View Properties** dialog box, click the **Notes** tab.
3. Enter a new note or edit the existing note.

   To delete the note, delete all the text.
4. Click **OK** to close the dialog box and save your changes.

Add or edit record notes

You can add a note to a record to provide additional information about the record, or to create a link to a related file. A note added to a record is available in any views of the table that include the record. You can edit the content of a record note at any time.

Note icon

Records that have a note attached are identified by a note icon next to the record number in the view  . Tables that have one or more records with a note are identified in the **Overview** tab in the **Navigator** with a note icon in the bottom left corner of the standard table icon  .

The RecordNote field

When you add the first record note in a table, Analytics automatically adds a field called **RecordNote** to the table layout, which is used to contain record notes. You can display record notes in views, or include them in printed Analytics reports, by adding the **RecordNote** column to the view. Once you have added the **RecordNote** column, you can double-click values in the column to quickly and easily maintain record notes.

Steps

Show me how

> **Tip**
> To add or edit multiple record notes simultaneously, use the NOTES command.

1. Right-click the appropriate record number in the record number column in the View tab (the grey, first column on the far left) and select **Edit Note**.
2. Enter a new note or edit the existing note.

   To delete the note, delete all the text.
3. If you want to create a link to a related file, do the following:
   a. Position the cursor at the location in the note where you want to insert the link.
   b. Click **File Reference**  .
   c. Select the appropriate file in the **Open** dialog box and click **Open**.

      A link to the file is added to the note using the following syntax:

      `file:///<path_to_file>`
4. Click **OK** to close the dialog box and save your changes.
5. If you want to display record notes in a view, or include them in printed Analytics reports, do the following:

a. Right-click in the display area and select **Add Columns**.
b. In the **Available Fields** list, double-click **RecordNote** and click **OK**.

Save record notes

If required, you can save record notes to a text file or to another Analytics table.

Whenever you update the data in an Analytics table (**Refresh from Source**) all record notes in the table are automatically deleted. You can save the record notes prior to updating the table, or you can use the Analytics option that allows you to save the notes in the process of updating the table. The saved notes are formatted slightly differently depending on how you save them.

Show me how

Save record notes to a text file prior to updating a table

1. Enter the following syntax in the command line:

   `LIST Recno() RecordNote to <file_name.txt>`

   For example, `LIST Recno() RecordNote to Ap_trans_record_notes.txt`

   The name of the text file must not include any spaces.

2. Click **Run**  .

Save record notes to another Analytics table prior to updating a table

1. Perform a standard extract by fields and select the **RecordNote** field.
2. Select at least one other field to extract, or create the expression “Recno( )” and add the expression as an additional field to extract.

   Analytics does not allow you to extract the **RecordNote** field by itself.

Save record notes to a text file while updating a table

In the process of updating a table, click **Yes** when prompted to save notes to a file.

The record notes are saved to a file called <*table_name*.txt>. The file is located in the same folder as the Analytics project.

Delete record notes

You can delete record notes individually or selectively, or delete all the record notes in a table at once.

Show me how

Delete record notes individually

1. Right-click the appropriate record number in the record number column in the View tab (the grey, first column on the far left) and select **Edit Note**.
2. Delete all content from the **Edit Note** dialog box, ensuring that you delete any spaces or line breaks that precede or follow text, and click **OK**.

> **Note**
>
> Individually deleting all the record notes in a table does not delete the auto-generated **RecordNote** field from the table layout, which means the note icon continues to appear with the table icon in the **Overview** tab in the **Navigator**.
>
> If your intention is to delete all the record notes in a table, use the method for deleting all record notes, which also deletes the **RecordNote** field.

Delete record notes selectively

1. Enter the following expression in the command line, using an IF statement that identifies the records with notes you want to delete:

   `NOTES IF <appropriate filter criteria> CLEAR`

   For example, `NOTES IF Location = "03" CLEAR` deletes any notes for Location #3 records.

2. Click **Run**  .

Delete all the record notes in a table at once

1. If the **RecordNote** column appears in the view, remove it by right-clicking the column header and selecting **Remove Selected Columns**.
2. Select **Edit > Notes > Delete All Notes from Table**.
3. Click **OK** in the confirmation dialog box.

   All record notes are deleted, the **RecordNote** field is deleted from the table layout, and upon a screen refresh, the note icon disappears from the table icon in the **Navigator**.

Add or edit field notes

You can add a note to a field to provide additional information about the field. The note appears in the **Note** column in the **Edit Fields/Expressions** tab in the **Table Layout** dialog box. You can edit the content of a field note at any time.

Field notes appear in printed Analytics reports if you select **Include Field Definitions in Table History** in the **Options** dialog box. For more information, see "Print options" on page 144.

Show me how

1. Select **Edit > Table Layout**.
2. In the **Edit Fields/Expressions** tab, double-click the field you want to add a note to.
3. Click **Edit Field Note**  .
4. Enter a new note or edit the existing note.

   To delete the note, delete all the text.
5. Click **Close**  .

   The note is automatically saved.
6. Click **Accept Entry**  .

Add or edit script notes

You can add a note to a script project item to record any general details about the script that you want to keep for future reference, or document for other users.

Notes added to a script project item are external to the script itself, and are not the same as inline comments added to the body of a script. You can edit the content of a script project item note at any time.

You do not need to open the script to add, edit, delete, or read the note.

Show me how

1. Right-click the script in the **Overview** tab in the **Navigator**.
2. Select **Properties**.
3. In the **Script Properties** dialog box, click the **Notes** tab.
4. Enter a new note or edit the existing note.

   To delete the note, delete all the text.
5. Click **OK** to close the dialog box and save your changes.

Add or edit workspace notes

For more information, see "Add or edit a workspace note" on page 763.

Printing Analytics project information

You can print a report from Analytics with complete and detailed information about the following items in the current Analytics project:

- Table layouts, including field definitions and filter syntax
- Views
- Scripts
- Indexes
- Workspaces
- Preference settings
- Project notes
- Command log

You can choose to print information about some or all of the items. The report can be used to create a permanent record of the project settings in use when an analysis was completed, or to assist in troubleshooting problems with an Analytics project.

> **Note**
> If you choose to print the command log, the entire log is printed, which could be numerous pages depending on the size of the log.

To print Analytics project information:

1. Select **File > Print Project Contents**.
2. In the **Print Documentation** dialog box, select the font point size to use for the generated report from the **Font Size** drop-down list.
3. Select one or more of the following item types to print the information associated with all occurrences of the item type in the project:
   - **All Table Layouts**
   - **All View Definitions**
   - **All Script Definitions**
   - **All Index Definitions**
   - **All Workspace Definitions**

   If you do not want to print information for all occurrences of an item type, leave the associated checkbox blank and use **Ctrl+click** or **Shift+click** to select individual items.
4. To include additional project information, or specify how the output should be formatted, select any of the following checkboxes:

- **Preferences** – prints a list of the currently selected preference settings in the **Options** dialog box
- **Project Notes** – prints any notes recorded for the project
- **Log** – prints the entire command log
- **Page Break after each Category** – inserts a page break after each project item category, and after preferences, project notes, and log entries. If the checkbox is not selected, each category is listed immediately after the previous category.
- **Page Break after each Item** – inserts a page break after each item within a category. For example, if you have selected three scripts, a page break will be inserted after each of the script definitions. If the checkbox is not selected, each item is listed immediately after the previous item in the category.

5. Click **Print**.
6. In the **Print** dialog box, configure any necessary printer settings and click **Print**. You can use the **Print** dialog box to modify settings, such as the printer to send the print job to and printer-specific options such as the page size and page orientation.

Recovering Analytics projects that close unexpectedly

If Analytics closes unexpectedly while an unsaved project is open, the next time you open the project you have the choice of opening an autosaved **Working** copy of the project (.ac file), or opening the **Last-saved** version of the project (.acl file).

- **Open the Working copy** – If you open the **Working** copy and then save your changes, the **Last-saved** version of the project file is overwritten and the original **Working** copy is deleted.
- **Open the Last-saved version** – If you open the **Last-saved** version and then save your changes, the **Last-saved** version of the project file is updated and the **Working** copy is deleted.
- **If you are uncertain which option to choose** – You can click **Cancel** and back up both the **Working** copy and the **Last-saved** version, so you retain the option of using either project file.

Guidelines

When you attempt to open a project that closed unexpectedly, an **ACL Analytics** dialog box is displayed presenting you with three options for recovering the project file. Select the appropriate option from the following:

- Click **Working** if you made modifications to project items or performed analysis steps after you last saved the project and you do not want to lose the log entries for these operations.

  > **Note**
  > The **Working** copy has the most complete information, but it may be corrupted if Analytics closed while a command was being processed.

- Click **Last-saved** if any unsaved changes in the project are not important to you.
- Click **Cancel** if you want to retain the option of using either version of the project file. After you close the dialog box, navigate to the Windows folder where the project files are stored and create a backup of both the **Working** copy and the **Last-saved** version using different file names.

This page intentionally left blank

Common data preparation and analysis tasks

When you use Analytics to prepare or analyze data, some general tasks frequently recur:

- saving output results
- specifying the location where output results are saved
- extracting data
- appending data
- exporting data

This section explains these tasks in greater detail. It also provides information about key fields, concatenating fields, and generating random numbers.

Saving results and specifying output folders

When you perform an operation on an Analytics table and save the results to a new Analytics table or a text file, you have several options regarding the manner in which you save the results, and the location of the output folder.

> **Note**
>
> Analytics tables include a table layout, visible in the **Navigator**, and an associated source data file with a .fil extension, not visible in the **Navigator**, stored in a Windows folder.
>
> Understanding the difference between the table layout and the source data file can be important when saving results and specifying output folders.
>
> For more information, see "The structure of Analytics tables" on page 115.

Saving results

When saving results to an Analytics table or a text file, you have the following options:

- **Save** – save the results to a new Analytics table or a text file
- **Append** – append the results to an existing Analytics table or a text file
- **Overwrite** – overwrite an existing Analytics table or a text file

Appending updates the source data file but does not alter the table layout. Overwriting replaces both the source data file and the table layout.

> **Note**
>
> Some Analytics operations support saving result to either an Analytics table or a text file, but not both.

Appending or overwriting source data in another project

Typically, you append to or overwrite a table in the open Analytics project. By navigating to the appropriate Windows folder when saving results, you can append to or overwrite the source data file for an Analytics table in another project.

When you do so, the updated or overwritten table remains in the other project and is also added to the open project, with both table layouts in the two projects sharing the same source data file.

> **Caution**
> Before saving results in this manner, you should be certain that overwriting source data in another project is your intended outcome.

Specifying output folders

When you save results, there are two types of 'output folder' to consider:

- **An Analytics project folder** – contains the resulting Analytics table layout (does not apply when saving results to a text file)
- **A Windows folder** – contains the resulting source data file (.fil) associated with the Analytics table layout, or contains the resulting text file

Analytics project folders are not Windows folders

Analytics project folders are not Windows folders, and creating an Analytics project folder does not create a corresponding Windows folder. Analytics project folders are virtual folders inside the Analytics project file (.acl). You can move a table layout between Analytics project folders and it has no effect on the location of the table's source data file in a Windows folder.

Table layout and source data file placement options

When saving results to an Analytics table, you have several options regarding placement of the table layout and the source data file. You can independently control placement of the table layout and the source data file by using the SET FOLDER command for table layout placement, and standard Windows navigation for source data file placement.

Item	Placement options (output folder)
Table layout	◦ the Analytics project folder containing the active table (the default) ◦ an Analytics project folder other than the active table folder, specified using the SET FOLDER command
Source data file (.fil)	◦ the Windows folder containing the Analytics project (the default) ◦ a Windows folder other than the folder containing the Analytics project ◦ the Prefix folder on AX Server (server tables only; the default) ◦ a folder on AX Server other than the Prefix folder (server tables only)

Using SET FOLDER to specify the Analytics project output folder

Before outputting results, you can use the SET FOLDER command to set which Analytics project folder is used for the table layout. You enter the SET FOLDER command in the Analytics command line, or include it in a script. Several examples are provided below.

The output folder remains as whatever you set it – until you reset it, or close the project. When you reopen the project, the output folder reverts to the default of the active table folder.

> **Note**
> File paths specified in the SET FOLDER command must use a forward slash.

Command syntax	Description
`SET FOLDER /Results`	The table layout is placed in the **Results** Analytics project folder rather than the active table folder.
`SET FOLDER /Results/Duplicates`	The table layout is placed in the **Duplicates** Analytics project subfolder rather than the active table folder.
`SET FOLDER /`	The table layout is placed in the Analytics project root directory rather than the active table folder.
`SET FOLDER`	Resets the output folder to the default of the active table folder.
`DISPLAY OUTPUTFOLDER`	Displays the current Analytics project output folder.

Interaction between table layouts and Analytics project folders

The interaction between table layouts resulting from operations and Analytics project folders is summarized in the table below.

In all cases, the Windows folder where you choose to save the source data file has no bearing on the location of the table layout or the project folder within Analytics.

Action	Output folder in Analytics is active table folder (default)	Output folder in Analytics is specified by SET FOLDER command
Save results to new Analytics table	Table layout added to same Analytics project folder as active table	Table layout added to Analytics project folder specified by SET FOLDER command
Append results to existing Analytics table in project	Existing table layout not moved	Existing table layout not moved
Save results and overwrite existing Analytics table in project	Table layout moved to same Analytics project folder as active table, unless already in same folder	Table layout moved to Analytics project folder specified by SET FOLDER command, unless already in same folder
Append results to existing Analytics table in another project	Table layout added to same Analytics project folder as active table	Table layout added to Analytics project folder specified by SET FOLDER command

Action	Output folder in Analytics is active table folder (default)	Output folder in Analytics is specified by SET FOLDER command
	Table layout in other project unaltered	Table layout in other project unaltered
Save results and overwrite existing Analytics table in another project	Both table layouts share the same source data file	Both table layouts share the same source data file

Specifying the Windows output folder

By default, the source data file (.fil) associated with a table layout is output to the Windows folder containing the Analytics project. To output the data file to a different Windows folder:

- **in the user interface** – navigate to the folder when using a command dialog box
- **in a script** – specify a file path in any command that outputs a table. For example:

```
CLASSIFY ON Vendor_Number SUBTOTAL Invoice_Amount TO "C:\Data
analysis\January\Classified_transactions_Jan.FIL"
```

Harmonizing Analytics project folders and Windows folders

It is possible to harmonize the structure of Analytics project folders and Windows folders if a direct parallel between the two sets of folders is important or helpful for your audit workflow. Analytics project folders are not Windows folders, and creating an Analytics project folder does not create a corresponding Windows folder. However, you can manually create and manually maintain a parallel folder structure if required. No programmatic link ever exists between the two sets of folders.

To harmonize Analytics project folders and Windows folders:

1. In an Analytics project, create the folders you require. For example, “Original Data”, “Working Files”, “Results”, and so on.
2. In the Windows folder containing the project, or another Windows folder, create subfolders that exactly replicate the structure of the Analytics project folders.
3. Organize your initial audit content appropriately. For example, put original data files such as Excel or Access files in the appropriate Windows folder. When using the **Data Definition Wizard** to import one of these files to Analytics, save the new source data file (.fil) in the same Windows folder as the original data file, or in another appropriate Windows folder. Finally, save the Analytics table layout in the Analytics project folder that corresponds with the Windows folder containing the new source data file.

   > **Tip:**
   > To ensure the table layout is saved in the appropriate Analytics project folder, begin the import process by right-clicking the folder.

4. Prior to performing an operation that saves results to an Analytics table, if necessary, use the SET FOLDER command to specify the appropriate Analytics project folder for the resulting table layout.

   For more information, see "Saving results and specifying output folders" on page 189.

5. In the dialog box associated with the operation, specify the appropriate Windows folder for the source data file using an absolute or relative file path, or by navigating to the folder.

   For example: `C:\Results\Classify.fil`, or `Results\Classify.fil`.

Extracting data

Extracting allows you to copy some or all of the records or fields from an Analytics table to a new Analytics table.

The new table can be:

- an identical copy containing all the source table records and fields
- a subset of the records in the source table
- a subset of the fields in the source table
- a subset of both the records and the fields in the source table

The existing sort order in the source table is maintained in the new table.

> **Note**
>
> Extracting data and appending it to the end of an existing Analytics table is a data combining technique. It is explained in the section on combining data. For more information, see "Extracting and appending data" on page 870.

The usefulness of extracting data

The following are some of the reasons for extracting data to a new table:

- produce a subset of only the data relevant to a particular analysis, and reduce file size and processing time
- use filters to isolate particular items in a separate table for further analysis
- preserve the integrity of an original data file by extracting its content to a working copy of the file
- convert computed fields to physical fields populated with the actual computed values
- extract data from a server table to a new, local table
- extract data from two or more related tables to a new Analytics table

The difference between extracting data and copying a table

The difference between extracting all data, and copying a table in the **Navigator** (**Edit > Copy**), is that extracting creates a new source data file (.fil) as well as a new table layout, whereas copying creates only a new table layout that remains associated with the original source data file.

Extracting by record, by view, or by fields

When you extract data, you have the following options:

- **Record** – extract entire records
- **View** – extract all the fields in a view
- **Fields** – extract a selection of individual fields

When you extract entire records, the record is copied exactly, including any data stored in undefined gaps in the table layout.

When you extract all the fields in a view, or individual fields, any undefined portion of a record is ignored, even if you extract all the fields in the source table.

Extracting computed fields

Computed fields remain as computed fields when you extract by record. They are converted to physical fields of the appropriate data type, and populated with the actual computed values, when you extract by view or by fields.

Extracting time data in a computed field

If a computed field contains local times with a UTC offset (for example, 23:59:59-05:00), the local times and the UTC offset are preserved when you extract by record.

When you extract by view or by fields, the local times and the UTC offset are converted to UTC without an offset. For example, 23:59:59-05:00 becomes 04:59:59.

Additional details about extracting by view

Selecting the **View** option in the **Extract** dialog box allows you to extract exactly the data that is currently displayed in the active view.

The following details apply when extracting by view:

Which fields are extracted?	Only fields that are currently displayed in the view are extracted. Any additional fields that are part of the table layout but not displayed in the view are not extracted. All fields in the view are extracted. If you want to extract a subset of fields, remove the unwanted fields from the view, create a new view with just the required fields, or use extract by fields instead of extract by view.
Field order	The fields are extracted in the order they appear in the view. If you want to extract the fields in a different order, rearrange them in the view, or create a new view with the fields in the desired order, prior to extracting.
Filtering	If a filter is currently applied to the view, only the data that meets the filter criteria is extracted.

Record notes	Records notes are extracted only if the **RecordNote** column has previously been added to the view.
Alternate column titles	If any alternate column titles are specified at the view level, extract by view preserves the view-level titles. If you use the syntax in the command log to rerun the extract command, alternate column titles specified in the table layout are used, and view-level titles are ignored.
Scripts Command line	Specifying extract by view is not supported in scripts or from the command line. When rendered in ACLScript, extract by view is actually an extract by fields (`EXTRACT FIELDS`) using all the fields in the active view, in the order in which they appear in the view.

Extracting logical fields

Extracting logical fields requires that **Include Filters in Field Lists** is selected (**Tools > Options > Interface**).

Setting a control total

If you are extracting all the records in a table, or all the data in a view or a selection of fields, you can set a control total on a numeric field to verify that all the data is in fact extracted.

You set a control total for a field in the **Table Layout** dialog box. Once you have extracted the data, in the new table select **Tools > Table History** to compare the input and output control totals. For more information, see "Define a physical field" on page 717.

Extracting data from server tables and local tables

You can extract data from both server tables and local tables. Data extracted from a server table can be saved to a table on the server, or on your local computer. Data extracted from a local table can be saved only to a table on your local computer.

Steps

You can extract some or all of the records or fields from an Analytics table and output them to a new Analytics table.

> **Note**
>
> Extracting data and appending it to the end of an existing Analytics table is a data combining technique. It is explained in the section on combining data. For more information, see "Extract and append data" on page 875.

Show me how

1. Open the table from which you want to extract records or fields.
2. Select **Data > Extract**.
3. On the **Main** tab, select one of the following:
   - **Record** – extract entire records

     The fields in the record are extracted in the order they appear in the table layout.
   - **View** – extract all the fields in the current view

     The fields are extracted in the order they appear in the view.
   - **Fields** – extract a selection of individual fields

     The fields are extracted in the order you select them.

   If you want to extract data from a child table in a table relation, select **Fields**, or select **View** if the child table fields have previously been added to the view. You cannot extract child table data using the **Record** option.

   > **Note**
   >
   > If you are extracting one or more computed fields, selecting **Record** preserves the extracted fields as computed expressions.
   >
   > Selecting **View** or **Fields** converts the extracted fields to physical fields of the appropriate data type and populates them with the actual computed values.

4. If you selected **Fields**, do one of the following:
   - Select the appropriate fields from the **Extract Fields** list.
   - Click **Extract Fields** to select the appropriate fields, or to create an expression, then click **OK**.

   Click **Extract Fields** if you want to select fields from a child table in a table relation. The **From Table** drop-down list in the **Selected Fields** dialog box allows you to select the appropriate child table.
5. If there are records in the current view that you want to exclude from processing, enter a condition in the **If** text box, or click **If** to create an IF statement using the **Expression Builder**.

   > **Note**
   >
   > The **If** condition is evaluated against only the records remaining in a table after any scope options have been applied (**First**, **Next**, **While**).

6. Do one of the following:
   - In the **To** text box, specify the name of the new Analytics table.

- Click **To** and specify the name of the new Analytics table, or select an existing table in the **Save** or **Save File As** dialog box to overwrite the table.

If Analytics prefills a table name, you can accept the prefilled name, or change it.

You can also specify an absolute or relative file path, or navigate to a different folder, to save the new table or overwrite an existing table in a location other than the project location. For example: `C:\Results\GL_2011.fil` or `Results\GL_2011.fil`. Regardless of where you save or overwrite the table, it is added to the open project if it is not already in the project.

7. If you are connected to a server, do one of the following:
   - Select **Local** to save the output table to the project, or to specify a path or navigate to a different local folder.
   - Leave **Local** deselected to save the output table to the Prefix folder on a server.

> **Note**
>
> For output results produced from analysis or processing of Analytics Exchange server tables, select **Local**. You cannot deselect the **Local** setting to import results tables to Analytics Exchange.

Select **Use Output Table** if you want the output table to open automatically upon completion of the operation.

8.
9. Click the **More** tab.
10. Select the appropriate option in the **Scope** panel:
    - **All**
    - **First**
    - **Next**
    - **While**

Show me more

All	This option is selected by default. Leave it selected to specify that all records in the view are processed.
First	Select this option and enter a number in the text box to start processing at the first record in the view and include only the specified number of records.
Next	Select this option and enter a number in the text box to start processing at the currently selected record in the view and include only the specified number of records. The actual record number in the leftmost column must be selected, not data in the row.
While	Select this option to use a WHILE statement to limit the processing of records in the view based on a particular criterion or set of criteria. You can enter a condition in the **While** text box, or click **While** to create a WHILE statement using the **Expression Builder**. A WHILE statement allows records in the view to be processed only while the specified condition evaluates to true. As soon as the condition evaluates to false, the processing terminates, and no further records are considered. You can use the **While** option in conjunction with the **All**, **First**, or **Next** options. Record processing stops as soon as one limit is reached.

> **Note**
> The number of records specified in the **First** or **Next** options references either the physical or the indexed order of records in a table, and disregards any filtering or quick sorting applied to the view. However, results of analytical operations respect any filtering.
> If a view is quick sorted, **Next** behaves like **First**.

11. If required, select **EOF (End of file processing)** to force the extract operation to execute one more time when the end of a table is reached.

    The EOF parameter is usually used if you are extracting records as part of a larger analytic process and the Extract command occurs inside a group in a script. If you are extracting records based on a comparison between sequential records, you may need to use EOF to ensure the final record in a table is extracted.

12. Click **OK**.
13. If the overwrite prompt appears, select the appropriate option.

Appending output results to an existing table

Any Analytics operation that allows you to output results to a new Analytics table or a text file also allows you to append the results to an existing Analytics table or a text file. Appending attaches the records in the output results as a group to the end of the existing table or file – that is, after the last record in the existing table or file.

How sorting works

Any existing sort orders in the output results and the target table or file are separately maintained in the respective record sets in the resulting combined data. If required, you can subsequently sort the combined table to create a single sort order throughout all records.

Identical data structure required

For appending to an Analytics table to be successful, the records in the output results and the target table **must be exactly identical in structure**. The following structural characteristics must all be identical:

- the selection of data elements
- the number and order of fields
- the data type and length of corresponding fields
- the format of corresponding date and datetime fields
- the length of records

If even one characteristic of the record structures is not identical, jumbled data can result. Identical structure is not a requirement when appending to text files.

For more information about record structure, see "Data structure and data format requirements" on page 848.

For information about using the DISPLAY command to compare the data structures of two tables, see "Comparing data structures" on page 202.

The Append To Existing File option

Selecting the **Append To Existing File** option prior to performing an Analytics operation forces the append of the output results to the target table. The append takes place regardless of whether the data structures are identical. Select this option only if you are certain the data structures are identical.

The Append button

A safer approach when appending to an Analytics table is to leave **Append To Existing File** deselected. If the option is deselected, upon processing of the active table Analytics compares the record lengths of the output results and the target table. If the record lengths are identical, the **Append** button appears as an option in the overwrite prompt.

Limits of the Append button

Even if the **Append** button appears, the two data structures may not be identical. For example, the output results and the target table could each have a record length of 100 characters, but the fields could be in a different order, there could be a different number of fields, or the data type of aligned fields (same start position and field length) could be different.

When the Append button does not appear

If the **Append** button does not appear, the record lengths are not identical, which means one or more aspects of the data structures are not identical, and may require manual harmonizing before you proceed. This automated check occurs only if the target table is in the open Analytics project.

Warn Before Overwriting Files option

In order for the overwrite prompt to appear, **Warn Before Overwriting Files** must be selected in the **Options** dialog box (the default setting). If **Warn Before Overwriting Files** is deselected, the overwrite prompt does not appear, the automated check of record lengths is not performed, and output results always overwrite target tables rather than being appended to them – unless you select **Append To Existing File**.

Comparing data structures

You can use the DISPLAY command to display the table layout of an Analytics table. The table layout specifies the data structure of the table.

Before appending a source table or output results to a target table, or merging two tables, you can display the table layouts of both tables and visually compare the structures to check if they are identical. The data structures of the two tables must be identical in order for the append or merge operation to work correctly.

The following data structure elements are displayed by the DISPLAY command:

- record length
- field name
- number of fields
- field order
- field start position
- field length
- field data type
- number of decimal places for numeric fields
- field format details
- computed field expression

Compare data structures

1. Open the source table.
2. Enter `DISPLAY` in the command line and click **Run**.

   Data structure information for the source table is displayed in the Results tab.
3. Click the pin icon to pin the Results tab and preserve the data structure information.
4. Open the target table.
5. Enter `DISPLAY` in the command line and click **Run**.

   Data structure information for the target table is displayed in a second Results tab.
6. Click the pin icon to pin the Results tab and preserve the data structure information.
7. Click back and forth between the two Results tabs to visually compare the two data structures.

   Any difference in data structures will prevent an append or merge operation from working correctly. You may need to first manually harmonize the data structures. For more information, see "Harmonizing fields" on page 851.

   If you are combining two tables, another option is combining the data outside Analytics. For more information, see "Alternative methods for combining data" on page 847.

Exporting data

You can convert and export Analytics data to use in other applications. Destination applications and formats include:

- Microsoft Excel (.xlsx, .xls)
- Text (.txt)
- Delimited text (.del)
- Comma-separated values (.csv)
- Microsoft Access (.mdb)
- Windows clipboard for pasting into other documents or applications
- XML (.xml)
- JSON (.json)
- dBASE III PLUS (.dbf)
- HighBond (export exceptions to Results)

  For more information, see "Exporting exceptions to HighBond Results" on page 208.

Opening an exported file

In the Analytics results tab, the screen display of the export command log entry contains links to:

- the exported file
- the folder containing the file

The links allow you to conveniently open the file, or the folder containing the file, directly from Analytics.

The exported file opens in the application associated with the file extension if the application is installed on your computer.

Exporting to Excel

You can export Analytics tables as individual Excel worksheets to newly created or existing Excel files. Exporting to an existing Excel file is supported for `*.xlsx` only.

Character and size limits

The following limits apply when exporting data to an Excel file:

Number of records	◦ **Excel 2007 and later (*.xlsx)** - a maximum of 1,048,576 records ◦ **Excel 97 and 2003** - a maximum of 65,536 records

	Analytics tables that exceed these maximums export successfully, but the excess records are ignored and not exported.
Length of fields	◦ no specific field length limit ◦ combined field lengths cannot exceed the overall record length limit of 32 KB (32,765 characters in non-Unicode Analytics, 16,382 characters in Unicode Analytics) ◦ for Excel 2.1, a maximum of 247 characters
Length of field names	◦ a maximum of 64 characters ◦ for Excel 2.1, a maximum of 248 characters

Datetime and time data exported to Excel

Datetime and time data exported to Excel may not display correctly when you initially open the Excel file. Datetimes may display only the date portion, and times may display “00/01/1900”. The complete datetime and time data is present in the Excel file, however you need to modify the way the cells are formatted in Excel to allow the data to display correctly.

Exporting data from server tables

You cannot save data exported from a server table to the server. You can export data from both server tables and local tables to your local computer.

Steps

You can export some or all of the records or fields in an Analytics table to use in other applications.

Show me how

Specify the fields to export

1. Select **Data > Export.**
2. On the **Main** tab, select one of the following:
   - **Fields** – specify which fields you want to export

     When you select this option, the fields are exported using the physical field names in the table layout.

     For information about renaming fields, see "Rename a field in a table layout" on page 751.

- **View** – export all fields in the current view

  When you select this option, the fields are exported using the column display names. The fields are exported in the same order as they appear in the view.

  For information about renaming columns, see "Rename columns in a view" on page 780.

3. If you chose **Fields**, do one of the following:
   - Select the field(s) to export from the **Export Fields** list.
   - Click **Export Fields** to select the field(s), or to create an expression.

Select the export format

Select the export format from the **Export As** drop-down list and follow the guidelines below.

Delimited or **Text** (or comma-separated values)	Do one of the following: ◦ **Delimited** – optionally select **Export with field names** to include the field names or the column names as headings in the export. Select the **Column Separator** and **Text Qualifier** characters that you want to use in the delimited file. ◦ **Text** – optionally select **Export with field names** to include the field names or the column names as headings in the export. Field values in the exported text file are separated with blank spaces and values are not qualified. **Tip** To export to a comma-separated values file (`*.csv`), select **Delimited** and make sure to select a comma `,` in the **Column Separator** drop-down list. When specifying the export file name in the **To** field, include the .csv file extension. For example: `vendors.csv`
Excel (.xlsx)	Do one of the following: ◦ **To create a new Excel file, or export to an existing Excel file** Keep the default name in the **Add worksheet** text box, or change it if required. When you export to a newly created or existing `*.xlsx` Excel file a worksheet is automatically created in the Excel file. The worksheet has the same name as the Analytics table you are exporting from unless you change the name. **Note** If you specify a worksheet name, it can contain only alphanumeric characters or the underscore character (_). The name cannot contain special characters, spaces, or start with a number. You can overwrite a worksheet in an existing Excel file, but only if the worksheet was originally created by exporting from Analytics to Excel. You cannot overwrite worksheets that were created directly in Excel, or any worksheet that has been renamed. ◦ **To overwrite an existing Excel file**

	Delete the name in the **Add worksheet** text box, and leave the text box empty. When you overwrite an existing Excel file, a worksheet with the same name as the Analytics table you are exporting from is automatically created in the resulting Excel file.
XML	◦ Optionally select **Export with XML Schema** to include the XML Schema in the exported XML file. The XML Schema contains metadata that describes the structure of the XML file, including the data type of the fields. You can validate the file against the Schema once the file has been exported.
Exports from the Unicode edition of Analytics	Do one of the following: ◦ **Select Unicode** – if the data you are exporting contains characters that are not supported by extended ASCII (ANSI) The exported data is encoded as Unicode UTF-16 LE. ◦ **Do not select Unicode** – if all the characters in the data you are exporting are supported by extended ASCII (ANSI) The exported data is encoded as extended ASCII (ANSI). Any unsupported characters are omitted from the exported file. **Note** The **Unicode** option is available only when you export to **Clipboard**, **Delimited**, **Text**, or **XML**. For more information, see "Galvanize Unicode products" on page 2576.
HighBond (HighBond users only)	◦ See "Exporting exceptions to HighBond Results" on page 208.

Finalize the export

1. If there are records in the current view that you want to exclude from processing, enter a condition in the **If** text box, or click **If** to create an IF statement using the **Expression Builder**.

   Note
   The **If** condition is evaluated against only the records remaining in a table after any scope options have been applied (**First**, **Next**, **While**).

   The IF statement considers all records in the view and filters out those that do not meet the specified condition.
2. Do one of the following:
   - In the **To** text box, specify the name of the file that will contain the exported data.
   - Click **To** and specify the file name, or select an existing file in the **Save** or **Save File As** dialog box.

If Analytics prefills a table name, you can accept the prefilled name, or change it.

> **Note**
> If you are exporting data to the clipboard, the **To** text box is disabled because you are not saving the data in a file.

3. Click the **More** tab.
4. Select the appropriate option in the **Scope** panel:
   - **All**
   - **First**
   - **Next**
   - **While**

   Show me more

All	This option is selected by default. Leave it selected to specify that all records in the view are processed.
First	Select this option and enter a number in the text box to start processing at the first record in the view and include only the specified number of records.
Next	Select this option and enter a number in the text box to start processing at the currently selected record in the view and include only the specified number of records. The actual record number in the leftmost column must be selected, not data in the row.
While	Select this option to use a WHILE statement to limit the processing of records in the view based on a particular criterion or set of criteria. You can enter a condition in the **While** text box, or click **While** to create a WHILE statement using the **Expression Builder**. A WHILE statement allows records in the view to be processed only while the specified condition evaluates to true. As soon as the condition evaluates to false, the processing terminates, and no further records are considered. You can use the **While** option in conjunction with the **All**, **First**, or **Next** options. Record processing stops as soon as one limit is reached.

> **Note**
> The number of records specified in the **First** or **Next** options references either the physical or the indexed order of records in a table, and disregards any filtering or quick sorting applied to the view. However, results of analytical operations respect any filtering.
> If a view is quick sorted, **Next** behaves like **First**.

5. If you are exporting to a delimited file or a text file, optionally select **Append To Existing File** if you want to append the exported data to the end of an existing file.
6. Click **OK**.
7. If the overwrite prompt appears, select the appropriate option.

Exporting exceptions to HighBond Results

If you use HighBond, you can export exception data in an Analytics table to a table in Results. To export exceptions, you use the standard procedure for exporting data from Analytics, with some minor differences.

Security requirements

The ability to export exception data to a control test requires a specific HighBond role assignment, or administrative privileges:

- Users with a Professional User or Professional Manager role for a Results collection can export results to any control test in the collection.

  > **Note**
  > Only users with the Professional Manager role can overwrite existing data in a control test.

- HighBond account admins and Results admins automatically get a Professional Manager role in all collections in the HighBond instances they administer.

Password requirement

Password not required

You do not need to specify a password to export to Results if you used online activation to activate your copy of Analytics. The password is automatically created and sent to Results based on activation information stored on your computer.

Password required

You do need to specify a password to export to Results if you used offline activation to activate your copy of Analytics. The required password value is a HighBond access token.

> **Note**
> A password is also required if you use a script to export to Results, and you run the script in Robots, Analytics Exchange, or the Analysis App window.

Acquire a HighBond access token

1. On the Analytics main menu, select **Tools > HighBond Access Token**.

   The **Manage API tokens** page opens in your browser. You may be required to first sign in to Launchpad.

2. Do one of the following:
   - **Use an existing token** – In the **Token** column, click the partially masked token that you want to use and enter your HighBond account password. The unmasked token is displayed.

     > **Tip**
     > Use an existing token unless you have a reason for creating a new one. If the existing token does not work, create a new one.
     >
     > Using an existing token cuts down on the number of tokens you need to manage.

   - **Create a new token** – Click **Create token > Analytics** and enter your HighBond account password.

     A new Analytics token is created.

     > **Note**
     > If you are a Launchpad System Admin, you also have the option of creating an API token. You should reserve API tokens for their intended purpose, which is programmatic access to the HighBond platform.

3. Click **Copy** to copy the token.

   > **Tip**
   > Do not close the dialog box containing the token until you have successfully pasted the token.

4. In Analytics, paste the token into the password prompt.
5. In Launchpad, close the dialog box containing the token.

   If you created a new token, a partially masked version of the token is added to the top of your list of tokens.

   For more information, see Creating and managing access tokens.

> **Caution**
> Safeguard your access tokens like any account password. They contain information unique to your HighBond account. You should not share access tokens.

Export limits

The limits that apply when exporting to a control test in Results are shown below.

Within these limits, you can export multiple times to the same control test. If data already exists in the control test, you have the option of overwriting it, or appending the new data.

> **Note**
> Although you can export up to 100,000 records to a control test, a better approach is to create smaller, more focused exception sets.

Item	Maximum
Records per export	100,000
Records per control test	100,000
Fields per record	500
Characters per field	256

Keeping fields aligned between Analytics and Results

If you are round-tripping data between Results and Analytics, you need to ensure that all field names in the Results table meet the more stringent Analytics field name requirements. If you do not, you risk misaligning your Analytics and Results data.

For example, any special characters in Results field names are automatically converted to underscores when they are imported into Analytics, which means the field names no longer match the original names in Results. If you then export the Analytics data back to the original table in Results, fields are no longer correctly matched.

To avoid this problem with data that you intend to round-trip, make sure that before you upload the data to Results from CSV or Excel files it meets these Analytics field name requirements:

- no special characters or spaces
- does not start with a number
- contains only alphanumeric characters, or the underscore character (_)

> **Note**
> When you append data to questionnaire fields, the display name of the column in Results remains the name that is specified in the questionnaire configuration, even if you changed the display name in Analytics.

Overwrite option and Results primary key

When you export exception data from Analytics to an existing Results table you have the option of appending the exported data to the table, or completely overwriting the table.

If the Results table has a field specified as a primary key, and the data you are exporting contains a corresponding field, the export behavior differs somewhat. (For more information about specifying a primary key in Results, see Specifying a primary key.)

The different possibilities are summarized below.

	No primary key in Results	Primary key in Results
Overwrite table option not selected	exported data is appended to the existing Results table	◦ **matching value** - if a matching value exists in the primary key field in Results and the corresponding field exported from Analytics, the record in Results is updated with the values present in the exported record ◦ **no matching value** - if a matching value does not exist in the primary key field in Results and the corresponding field exported from Analytics, the record in Results is not updated and the exported record is appended to the table
Overwrite table option selected	exported data replaces (overwrites) the existing Results table	exported data replaces (overwrites) the existing Results table

Export exceptions to Results

Specify the fields to export

1. Open the table with the exception data that you want to export.
2. Select **Data > Export**.
3. On the **Main** tab, select one of the following:
   - **Fields** - specify which fields you want to export

     When you select this option, the fields are exported using the physical field names in the table layout.

     For information about renaming fields, see "Rename a field in a table layout" on page 751.

- **View** – export all fields in the current view

  When you select this option, the fields are exported using the column display names. The fields are exported in the same order as they appear in the view.

  For information about renaming columns, see "Rename columns in a view" on page 780.

4. If you chose **Fields**, do one of the following:
   - Select the field(s) to export from the **Export Fields** list.

     > **Tip**
     > You can **Ctrl+click** to select multiple non-adjacent fields, and **Shift+click** to select multiple adjacent fields.

   - Click **Export Fields** to select the field(s), or to create an expression.

Select the export options

1. In the **Export As** drop-down list, select **HighBond**.
2. Do one of the following:
   - **Append to the Results table**

     If you want to append the exported data to the existing table in Results leave **Overwrite table** deselected.

     > **Note**
     > Analytics fields can only be appended to existing Results fields if they have matching physical field names, regardless of their display name in either application. In Analytics, the physical field name is the name in the table layout.
     >
     > The order of fields within the two applications does not affect field name matching.
     >
     > Exported fields with physical names that do not match the physical name of a field in the Results table create new columns in the table.

   - **Replace (overwrite) the Results table**

     If you want to replace the existing table in Results select **Overwrite table**.

   For more information, see "Overwrite option and Results primary key" on the previous page.
3. (Optional) If you want to export column display names to Results, select **Include field display name**.

   Selecting this option makes the column display name and the physical name in Results identical to the names in Analytics.

   If you do not select **Include field display name**, the result depends on whether you are exporting by fields or by view:

	Exporting Fields	Exporting View
Include field display name selected	Field name in Results is the field name from Analytics. Display name in Results is the display name from Analytics.	
Include field display name not selected	Field name and display name in Results are the field name from Analytics.	Field name and display name in Results are the display name from Analytics.

> **Note**
> Do not select **Include field display name** if you are appending a view to a Results table that was initially created by exporting a view from an Analytics version older than 14.1. Doing so may export columns with field names that are not the same as the names in Results, which will create new columns in Results and misalign the data between applications.

Finalize the export

1. If there are records in the current view that you want to exclude from processing, enter a condition in the **If** text box, or click **If** to create an IF statement using the **Expression Builder**.

   > **Note**
   > The **If** condition is evaluated against only the records remaining in a table after any scope options have been applied (**First**, **Next**, **While**).

   The IF statement considers all records in the view and filters out those that do not meet the specified condition.

2. Do one of the following:

   If you know the ID number of the table you are exporting to:

   Enter the number in the **To** text box.

   - Enter the number without any quotation marks – for example, `99`
   - Enter only the number. Do not enter a file name.
   - If you are exporting to a data center other than North America (US), you must also specify the data center code. The control test ID number and the data center code must be separated by the at sign (@) – for example, `99@eu`. The data center code specifies which regional HighBond server you are exporting the data to.
     - `af` – Africa (South Africa)
     - `ap` – Asia Pacific (Singapore)
     - `au` – Asia Pacific (Australia)
     - `ca` – North America (Canada)
     - `eu` – Europe (Germany)
     - `sa` – South America (Brazil)

- `us` - North America (US)

  You can use only the data center code or codes authorized for your organization's instance of HighBond. The North America (US) data center is the default, so specifying `@us` is optional.

If you do not know the ID number of the table you are exporting to, or if you want to create a new table:

a. Click **To**, and in the **Select Destination Test** dialog box navigate to the appropriate analysis folder.
b. Do one of the following:
   - Select an existing table and click **OK**.
   - Enter a name in the **New data analytic** field and click **Create**.

   You are returned to the **Export** dialog box and the control test ID number and data center code are prefilled in the **To** text box.

3. Click the **More** tab.
4. Select the appropriate option in the **Scope** panel:

   - **All**
   - **First**
   - **Next**
   - **While**

Show me more

All	This option is selected by default. Leave it selected to specify that all records in the view are processed.
First	Select this option and enter a number in the text box to start processing at the first record in the view and include only the specified number of records.
Next	Select this option and enter a number in the text box to start processing at the currently selected record in the view and include only the specified number of records. The actual record number in the leftmost column must be selected, not data in the row.
While	Select this option to use a WHILE statement to limit the processing of records in the view based on a particular criterion or set of criteria. You can enter a condition in the **While** text box, or click **While** to create a WHILE statement using the **Expression Builder**. A WHILE statement allows records in the view to be processed only while the specified condition evaluates to true. As soon as the condition evaluates to false, the processing terminates, and no further records are considered. You can use the **While** option in conjunction with the **All**, **First**, or **Next** options. Record processing stops as soon as one limit is reached.

> **Note**
> The number of records specified in the **First** or **Next** options references either the physical or the indexed order of records in a table, and disregards any filtering or quick sorting applied to the view. However, results of analytical operations respect any filtering.
> If a view is quick sorted, **Next** behaves like **First**.

5. Click **OK**.

   A progress indicator appears while the exception data is exported to Results. When the export operation is complete, an entry is created in the log.

About key fields

Several operations in Analytics make use of key fields:

- joining
- relating
- merging
- sorting
- indexing

Depending on which operation you are performing, the term 'key field' can have a different meaning, and key fields can have a different function. Key fields in Analytics also differ somewhat from the typical definition of a key field in relational database terminology.

Key fields when joining, relating, or merging

Joining, relating, and merging in Analytics are all data combining operations involving two or more tables. The term 'key field', in this context, means the common field in two tables being combined from which values are compared and matched – or in the case of merging, compared and interfiled.

Appending, another method for combining data in Analytics, does not make use of key fields.

Primary and secondary tables and key fields

The first table you open when joining or merging becomes the primary table, and the key field you choose becomes the primary key field. The second table you open becomes the secondary table, and the key field you choose becomes the secondary key field.

When you relate tables, primary is referred to as 'parent', and secondary is referred to as 'child'.

You are free to choose whatever primary and secondary tables and key fields you want when combining data. Analytics does not enforce any particular choice of field, although it does require that key field pairs have an identical data structure.

Unique key and foreign key designation not retained from source data

Data imported into an Analytics table, either locally or on a server, is stored in a non-relational flat file (a .fil file). In a .fil file, fields that may previously have functioned as primary keys, unique keys, foreign keys, or secondary keys in a relational database are not treated any differently from non-key fields.

A primary key from a relational database, such as employee ID, only becomes a primary or parent key in an Analytics table when you designate it as such in an Analytics command. As a user making a decision about how to construct a join or a relation in Analytics, you may need to know which fields

were primary or unique keys in a source database. However, Analytics does not contain this information.

The same situation is true when you directly access database tables using an Analytics database profile. Analytics retains no information about which fields are key fields in the database, and you may need to know this information yourself when constructing a database query.

Uniqueness of Analytics key fields not enforced

Analytics does not enforce uniqueness in the key fields you designate in Analytics commands. Identical values can exist in both the primary and the secondary key fields.

Key fields when sorting or indexing

Sorting and indexing in Analytics are single-table operations that impose a sequential order on a table. In this context, the term 'key field' means the field upon which the sorting or indexing is based, containing the values that are sorted or indexed.

Equivalent to a 'sort key' or an 'index key'

The sorting or indexing key field in Analytics is equivalent to the 'sort key' or 'index key' in general computing or database terminology. Uniqueness is not enforced.

You are free to choose whatever key field you want when sorting or indexing data. Analytics does not contain any information about fields that may have been sort or index keys in the original source data, although the values in those fields may still be in sequential order.

Keys and nested sorting or indexing

In the case of nested sorting or indexing, a 'primary' sort or index key takes precedence over a 'secondary' sort or index key. Primary and secondary keys are established simply by the order in which you select them.

Concatenating fields

If your analysis requires testing or processing two or more fields in a table as a single data element, you can create a computed field that concatenates (adds together) the fields. You can then test or process the combined data in the computed field.

For example, you could concatenate first, middle, and last name fields into a single field containing full names, or concatenate vendor ID and location code fields to produce unique identifiers for each outlet of every retail chain in a table.

> **Note**
>
> You can concatenate only character fields. If necessary, use Analytics functions to convert non-character data prior to concatenating.

1. Open a table and select **Edit > Table Layout**.
2. Click **Add a New Expression**  .
3. Enter a **Name** for the concatenated field.
4. Click **f(x)**  to open the **Expression Builder**.
5. Build an expression using two or more fields and the Add operator (**+**).

   If required, you can include separator characters, such as blanks, in the expression, and use the TRIM( ) function to remove trailing blanks from fields. For example:

   ```
   TRIM(first_name) + " " + TRIM(middle_name) + " " + last_name
   ```

6. Click **OK**.

   If you get an “Expression type mismatch” error, one or more of the fields are probably not character fields.

7. Click **Accept Entry**  and click **Close**  to exit the **Table Layout** dialog box.

   For information about how to add the computed field to the view, see "Add columns to a view" on page 778.

Generating random numbers

You can use Analytics to generate a set of random numbers. You can specify certain parameters, such as the size of the set, and the range.

Typically, the set of generated values is used for applications outside Analytics, such as drawing a random selection of hard-copy files.

> **Note**
> If you require a random selection to be statistical valid or representative of the entire population, you need to follow a more formal sampling process. For more information, see "Sampling data" on page 949.

To generate random numbers:

1. Select **Tools > Generate Random Numbers**.
2. In the **Main** tab enter the following information:
   - **Number** – The size of the set of random numbers to be generated.

     A maximum of 32767 numbers can be generated.
   - **Seed** – Optional. The value used to initialize the random number generator.

     You can specify a seed value, or you can enter a seed value of '0', or leave the seed value blank, if you want Analytics to randomly select a seed value.

     If you specify a seed value, it can be any number. Each unique seed value results in a different set of random numbers. If you respecify the same seed value, the same set of random numbers is generated. Explicitly specify a seed value, and save it, if you want to replicate a particular set of random numbers.
   - **Minimum** – The smallest possible number in the set of random numbers. Any valid numeric value or expression is allowed.
   - **Maximum** – The largest possible number in the set of random numbers. Any valid numeric value or expression is allowed.
   - **Columns** – The number of columns used to display the set of random numbers. The default number of columns is 6.
   - **Unique** – Specifies that only unique numbers are included in the set of random numbers.

     The default behavior is to allow duplicates in the set of random numbers.

> **Note**
> You should not select **Unique** when the specified size of the set of random numbers exceeds 75 percent of the range between **Minimum** and **Maximum**. Doing so can result in too many random number selections being discarded.

- **Sorted** – Specifies that the set of random numbers is displayed in ascending order.

  By default, the numbers are displayed in the order in which they are randomly selected.
- **Append To Existing File** – Specifies that the output results should be appended to the end of an existing file instead of overwriting the existing file.

3. Click the **Output** tab.
4. Select the appropriate output option in the **To** panel:
   - **Screen** – displays the set of random numbers in the Results tab in the Analytics display area.
   - **File** – saves the set of random numbers to a text file.
5. If you selected **File** as the output type, specify the file name in the **Name** text box in the **As** panel, or click **Name** and browse to select an existing file.

   If the **Append To Existing File** checkbox is selected the output is appended to a file with the same name, if found, otherwise you are prompted to either overwrite the file or append the output.

   You can also specify an absolute or relative file path, or navigate to a different folder, to save or append the output to a file in a location other than the project location. For example: `C:\Output\random.txt` or `Output\random.txt`.
6. Click **OK**.

Generate a random selection of records

You can use Analytics to generate a random selection of records.

The method outlined below selects records at random, **however the resulting output cannot be considered representative of the entire population of records**.

If you intend to do analysis on the selected records and project the results back to the entire population, the selection needs to be statistically valid, or representative. For more information, see "Sampling data" on page 949.

Steps

1. Open the table from which you want to randomly select records.
2. From the main menu, select **Sampling > Record/Monetary Unit Sampling > Sample**.
3. Under **Sample Type**, select **Record**.
4. Under **Sample Parameters**, select **Random**.
5. Specify the following values:
   - **Size** – the number of records that you want to randomly select
   - **Seed** – (optional) a seed value to initialize the Analytics random number generator

     The seed value can be any number. You can recreate the same random selection of records by respecifying the same seed value.

     Enter a seed value of '0', or leave the seed blank, if you want Analytics to randomly select a seed value.

   - **Population** – the total number of records in the table
   - **To** – the name of the output table
6. Click **OK**.

This page intentionally left blank

Defining and importing data

Before you can analyze data in Analytics you must create an Analytics table to contain the data. You create an Analytics table by defining and importing data. Regardless of the type of source data, defining and importing follows the same basic process:

1	Navigate or connect	Navigate to a source data file, or connect to a file or a database containing the source data.
2	Define	Define the source data, which means: specify information about the structure and characteristics of the source data so that Analytics can read it. **Note** Analytics automatically defines certain source data formats so that user definition of the data is not required.
3	Import or read directly	Import the source data into a native Analytics data file, or read the data directly from the source without creating an Analytics data file.
4	Name and save the Analytics table	Name and save the automatically created Analytics table.

> **Note**
> When connecting to any data source, or importing from any data source, Analytics is strictly read-only. For more information, see "Data access by Analytics is read-only" on page 228.

Components for defining and importing data

Analytics provides two components for defining data, importing data or reading data directly from the source, and creating an Analytics table:

- the Data Definition Wizard
- the Data Access window

The Data Definition Wizard

The Data Definition Wizard is a page-based wizard that provides a standard way to access a variety of data sources, mostly file-based.

The basic process for defining and importing data using the wizard is consistent, but the selection and sequence of pages presented depends on the type of data source you are using.

New projects and tables

By default, the Data Definition Wizard is automatically displayed when you create a new Analytics project, and when you add a new Analytics table to a project (**Import > File**, or **File > New > Table**).

The Data Access window

The Data Access window is a visual interface that contains a number of data connectors you can use to access source data in either databases or files. The data connectors use either native Analytics ODBC drivers, or whatever Windows ODBC drivers you have installed.

Once you connect to a data source using the Data Access window you have a standard set of options for defining and importing the data:

- Table search and selection
- Field selection
- Table joins
- Data filtering
- Data import preview
- Data import size estimate
- Field length specification
- SQL mode for directly editing the SQL import statement

Which component should I use for defining and importing data?

In many cases, the type of data source you want to access dictates which component you must use. For a list of all the data sources you can access with Analytics, and which component you must use to access them, see "Data sources you can access with Analytics" on page 227.

For some data sources you can use either the Data Definition Wizard or the Data Access window – for example, when importing Microsoft Excel or Access files.

In general, the Data Access window is a modern, visual interface with greater ease of use than the Data Definition Wizard.

The table below compares the different options available with the two different components:

Option	Data Definition Wizard	Data Access window
Select tables	Yes	Yes
Search tables	No	Yes

Option	Data Definition Wizard	Data Access window
Select fields	Depends on the data source	Yes
Import multiple tables	No (Yes for Excel)	Yes (up to 5)
Join tables	No	Yes
Filter data	No	Yes
Preview data import	Yes (basic)	Yes (modern interface, easily refreshable)
Estimate data import size	No	Yes
Specify field length	Yes	Yes
Rename fields	Depends on the data source	Yes (in SQL Mode)
Change field data type	Depends on the data source	No (data type can be changed after import)

Data sources you can access with Analytics

You can access a wide variety of file types, databases, and cloud data sources with Analytics.

The information provided below about data sources includes:

- **Access method** – whether you use the Data Definition Wizard or the Data Access window to access the data, or if you can use either
- **Data read method** – whether the resulting Analytics table reads data from an Analytics data file (.fil) or directly from the data source

  Data in a .fil file is static and must be manually updated, whereas data sources that are accessed directly are updated with the most current information each time the Analytics table is opened.

> **Note**
> The maximum record length supported by an Analytics data file (.fil) is 32 KB. Any record that exceeds 32 KB causes the import process to fail.

Using the Data Access window to access any ODBC data source

You can use the Data Access window to create a connection to any ODBC-compliant data source.

The following ODBC options are available in the Data Access window:

- **ACL connectors** – Analytics contains a number of native data connectors to data sources such as Oracle, Microsoft SQL Server, and Salesforce. See "Databases and cloud data services" on page 229 for the entire list of native data connectors.
- **ACL DSN connectors (Bundled)** – contains the data connectors that are provided by our data partner, CData.
- **Windows DSN connectors** – Use any Windows ODBC driver or DSN already installed or configured on your computer.
- **Other connectors** – Install any ODBC driver you require, and use it with the Data Access window.

> **Note**
> Certain requirements or prerequisites exist when using the Data Access window to connect to a database or a cloud data service. For more information, see "Before you connect to a database or a cloud data service" on page 357.

Data access by Analytics is read-only

When connecting to any data source, or importing from any data source, Analytics is strictly read-only. Analytics cannot add, update, or delete data in a data source, or modify a data source in any way. This restriction applies to all data sources accessible by Analytics: file-based data sources, databases, and cloud data services.

Analytics data files (.fil) created from imported data are also treated as read-only by Analytics. Analytics cannot alter .fil files, with the exception of refreshing the file from the data source.

.fil files are completely separate from the data source used to create them. Deleting a .fil file has no effect on the data source.

File-based data sources

Data source	Use Data Definition Wizard	Use Data Access window	Analytics table reads from
Adobe Acrobat (.pdf)	Yes	No	Analytics data file (.fil)
ACCPAC master file	Yes	No	data source
dBASE-compatible file (.dbf)	Yes	No	data source
Delimited text (.csv or .txt)	Yes	Yes	Analytics data file (.fil)
Microsoft Access (.mdb or .accdb)	Yes	Yes	Analytics data file (.fil)
Microsoft Excel (.xls or .xlsx)	Yes	Yes	Analytics data file (.fil)
Print Image (Report) (.txt)	Yes	No	Analytics data file (.fil)
SAP private file format/DART (.dat)	Yes	No	Analytics data file (.fil)
XBRL (.xml or .xbrl)	Yes	No	Analytics data file (.fil)

Data source	Use Data Definition Wizard	Use Data Access window	Analytics table reads from
XML (.xml)	Yes	No	Analytics data file (.fil)
SAP (via Direct Link, an optional utility)	No	No	Analytics data file (.fil)
HighBond			
The Projects app	Yes	No	Analytics data file (.fil)
The Results app	Yes	No	Analytics data file (.fil)
External Definition			
AS/400 FDF (.fdf)	Yes	No	data source
Cobol (.cob)	Yes	No	data source
PL/1 (.txt)	Yes	No	data source

Databases and cloud data services

You must use the Data Access window to access the databases and cloud data services listed below. The imported data is saved to an Analytics data file (.fil). For more information, see "Importing data using the Data Access window " on page 353.

> **Note**
> You can use the Data Access window to access any ODBC-compliant data source, not just the native data connectors listed below. For more information, see "Using the Data Access window to access any ODBC data source" on page 227.

- Active Directory
- Amazon Athena
- Amazon DynamoDB
- Amazon Redshift
- Amazon S3
- Apache Cassandra
- Apache Drill
- Apache HBase

- Apache Hive
- Apache Spark
- AWS Data Management
- Azure Data Management
- Azure Table Storage
- Box
- Cloudera Impala
- Concur
- Couchbase
- DocuSign
- Dynamics CRM
- Dynamics GP
- Dynamics NAV
- Dynamics 365 Business Central
- Dynamics 365 Finance and Operations
- Dynamics 365 Sales
- Edgar Online
- Email
- Epicor ERP
- Exact Online
- Exchange
- Google BigQuery
- Jira
- JSON
- LDAP
- LinkedIn
- Marketo
- Microsoft SQL Server
- MongoDB
- MySQL
- NetSuite
- OData
- Open Exchange Rates
- Oracle
- Oracle Eloqua
- Oracle Sales Cloud
- Presto
- Qualys
- QuickBooks
- QuickBooks Online
- QuickBooks POS
- REST Services
- Rsam
- RSS/ATOM
- Sage 50 UK
- Sage Cloud Accounting
- Sage Intacct
- Salesforce

- SAP

  (requires a separate subscription entitlement)
- SAP ByDesign
- SAP Hybris Cloud for Customer
- SAP SuccessFactors
- ServiceNow
- SFTP
- SharePoint
- Slack
- Snowflake
- Splunk
- Square
- Stripe
- SugarCRM
- SurveyMonkey
- Sybase
- Sybase IQ
- Tenable SecurityCenter
- Teradata
- Twitter
- UPS
- USPS
- xBase
- Zendesk

Defining and importing data using the Data Definition Wizard

The Data Definition Wizard is a component of Analytics that you can use to perform the following tasks in a single, wizard-based process:

- define data
- import data
- create a new Analytics table

The wizard provides a standard way to access a variety of different data sources. The basic process for creating an Analytics table from a data source is consistent, but depending on the data source, the wizard presents different pages and options. For example, the wizard behaves quite differently if you are importing an Excel file, a PDF file, or an XML file.

By default, the Data Definition Wizard is automatically displayed when you create a new Analytics project, and when you add a new Analytics table to a project (**Import > File**, or **File > New > Table**).

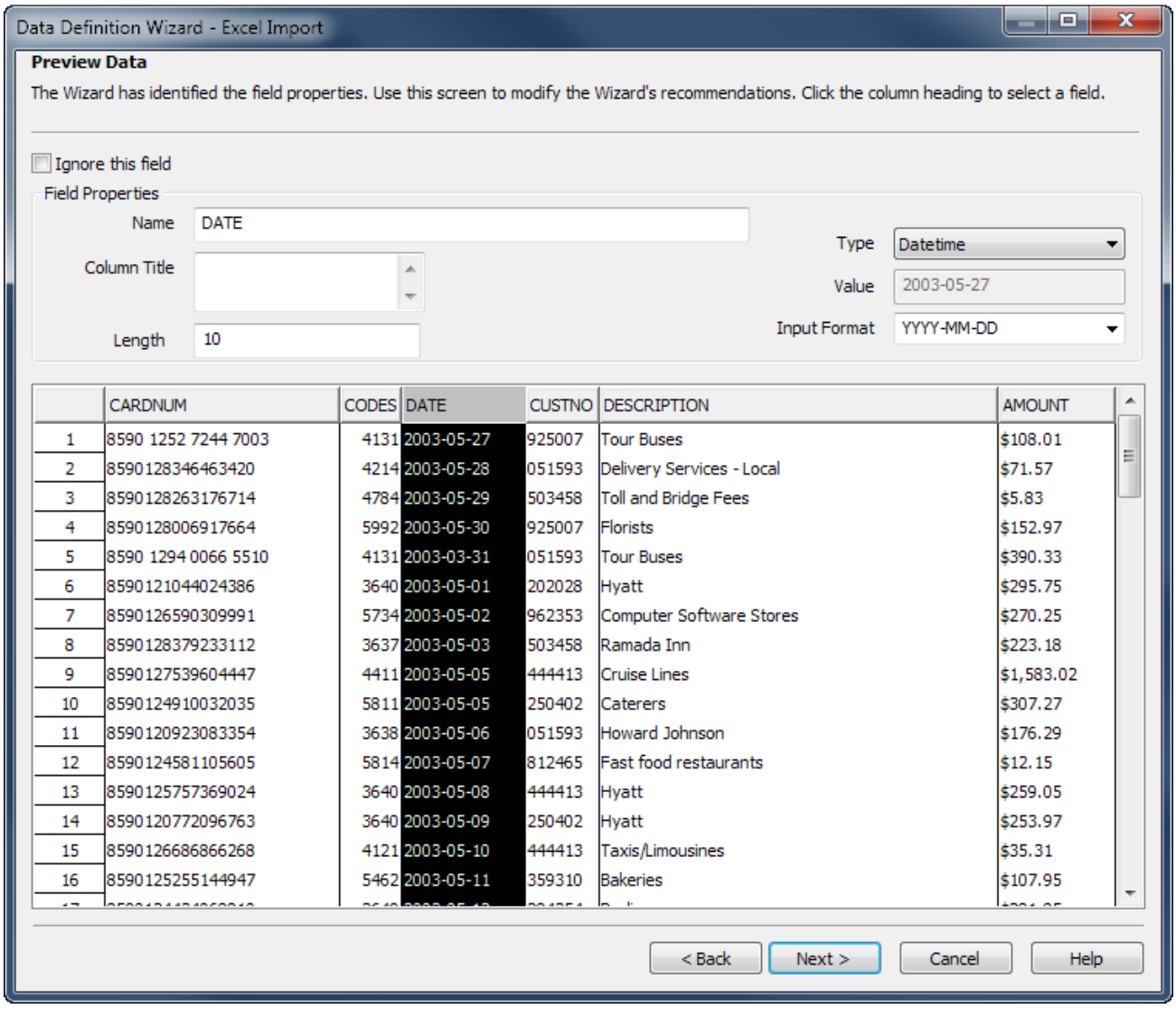
> **Note**
>
> You can also import data using the Data Access window. For more information, see "Importing data using the Data Access window " on page 353.
>
> When connecting to any data source, or importing from any data source, Analytics is strictly read-only. For more information, see "Data access by Analytics is read-only" on page 228.

Defining data

You may be required to **define** the data as you import it, which means to specify metadata such as:

- field names
- field lengths
- field data types
- format of numeric and datetime values

The image below shows the definition of the **DATE** field in an Excel worksheet being imported using the Data Definition Wizard.



	CARDNUM	CODES	DATE	CUSTNO	DESCRIPTION	AMOUNT
1	8590 1252 7244 7003	4131	2003-05-27	925007	Tour Buses	$108.01
2	8590128346463420	4214	2003-05-28	051593	Delivery Services - Local	$71.57
3	8590128263176714	4784	2003-05-29	503458	Toll and Bridge Fees	$5.83
4	8590128006917664	5992	2003-05-30	925007	Florists	$152.97
5	8590 1294 0066 5510	4131	2003-03-31	051593	Tour Buses	$390.33
6	8590121044024386	3640	2003-05-01	202028	Hyatt	$295.75
7	8590126590309991	5734	2003-05-02	962353	Computer Software Stores	$270.25
8	8590128379233112	3637	2003-05-03	503458	Ramada Inn	$223.18
9	8590127539604447	4411	2003-05-05	444413	Cruise Lines	$1,583.02
10	8590124910032035	5811	2003-05-05	250402	Caterers	$307.27
11	8590120923083354	3638	2003-05-06	051593	Howard Johnson	$176.29
12	8590124581105605	5814	2003-05-07	812465	Fast food restaurants	$12.15
13	8590125757369024	3640	2003-05-08	444413	Hyatt	$259.05
14	8590120772096763	3640	2003-05-09	250402	Hyatt	$253.97
15	8590126686866268	4121	2003-05-10	444413	Taxis/Limousines	$35.31
16	8590125255144947	5462	2003-05-11	359310	Bakeries	$107.95

Varying degrees of automation

Whenever possible, the Data Definition Wizard uses one of the following methods to automatically define source data:

- Reads layout information contained in the source file
- Parses the source file and identifies patterns in the data
- Queries the database for layout information

The data definition process is more involved for files that do not contain any layout information. In these cases, the Data Definition Wizard prompts you to provide the required information.

Import Microsoft Excel data

Import Microsoft Excel data to Analytics for analysis using a variety of different tools.

How it works

You use the Data Definition Wizard to select one or more Excel files, specify one or more worksheets to import, and import the Excel data to Analytics. The imported data creates one or more new Analytics tables and associated data files (.fil). Each imported worksheet creates a separate Analytics table.

The Analytics data file contains a copy of the Excel data that is completely separate from the original Excel file.

You can import data from an Excel file even if you do not have Microsoft Excel installed on your computer.

Import a single worksheet or multiple worksheets

You have the option of importing a single Excel worksheet or multiple Excel worksheets in a single operation. The import process differs somewhat, depending on which option you use:

- **single worksheet** - you have the option of manually defining the source Excel data during the import process
- **multiple worksheets** - Analytics automatically defines the source Excel data and no manual definition is possible during the import process

  For example, during the import of multiple worksheets, you **cannot**:

  - specify the data type or length of fields
  - selectively exclude fields from the import

  Once the data is in Analytics, you can make any necessary adjustments to the data definition in the **Table Layout** dialog box.

Import a named range

Instead of importing an entire worksheet, you can import a named range, which is a defined portion of a worksheet. See the Excel Help for information about creating a named range.

Combine multiple worksheets

After you import multiple Excel worksheets into individual Analytics tables you might want to combine them into a single Analytics table. For example, you could combine the data from twelve monthly tables into a single annual table containing all the data. You can combine the worksheets only after you have imported them into individual Analytics tables.

For information about combining multiple Analytics tables, see "Appending tables" on page 858.

> **Tip**
> To reduce labor, try combining the multiple tables first before making any required adjustments to the data definition in the new combined table.

Guidelines

Review the guidelines below to assist you with importing Excel data.

Data types and missing data

To get the best results when importing Excel data, ensure that in each worksheet you intend to import:

- each column contains the same type of data
- there are no blank rows or blank columns

Maximum numbers of columns and characters

Excel 2007 and later

The maximum number of Excel columns, and the maximum number of characters in a field, that you can import from `.xlsx` or `.xlsm` files is not limited to a specific number.

Importing from these Excel file types is governed by the record length limit in Analytics data files (.fil) of 32 KB. If any record in the source Excel data would create an Analytics record longer than 32 KB, the import fails.

> **Note**
> When the new table opens in Analytics, a maximum of 256 columns are displayed in the default view. If the table contains additional columns, you can manually add them to the view, if required.

Excel 97 – 2003

The import of `.xls` files (Excel 97 - 2003) uses an older type of processing, and is subject to the following maximums:

- 255 columns
- 255 characters per field
- 32 KB per record
- 65,000 rows

Supported versions of Excel

You can import data from any version of Excel from Excel 3.0 to Excel 2016.

If you want to import data from an earlier version of Excel, you need to save the Excel file to another file format that Analytics can import, such as `.csv`.

Excel Protected View

Analytics cannot import from an Excel workbook if Protected View is active for the workbook. You must first enable editing in the workbook, save and close the workbook, and then perform the import.

Web-based Excel files not supported

Analytics does not support directly importing Excel files created from web applications such as Google Sheets. You must first open the file in Excel, save it under a different file name, and then import the new file using Analytics.

How overwriting works

If you import Excel data and create a new Analytics table that has the same name as an existing table in the Analytics project you have the option of overwriting the existing table.

Show me more

Both parts of a table overwritten

Overwriting of Analytics tables is complicated by the fact that both parts of a table can be overwritten:

- the table layout, displayed in the **Navigator**
- the associated source data file, stored in a Windows folder

(For information about table layouts and source data files, see "The structure of Analytics tables" on page 115.)

The two parts of the table are overwritten independently of each other. If both parts have the same name as the new table, both are overwritten. This scenario is the most common.

But if the table layout and the source data file have different names, only the one with the same name as the new table is overwritten.

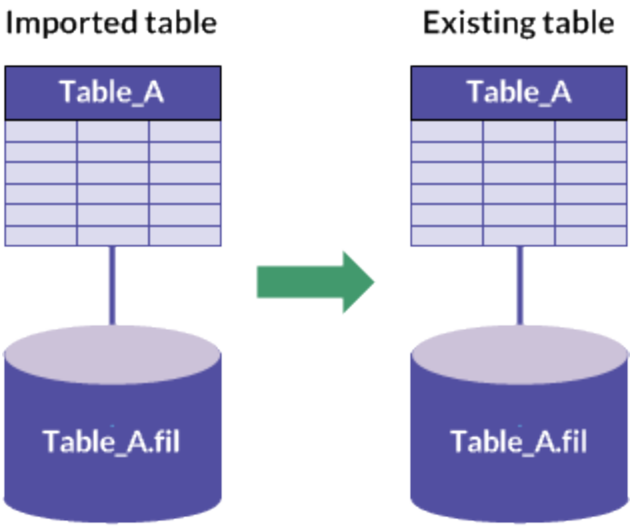
This overwrite behavior applies whether you are importing a single Excel worksheet, or multiple worksheets.

Overwriting when importing multiple worksheets

When you import multiple Excel worksheets, overwriting also depends on both the **Overwrite existing tables** and **Output Path** settings.

The sections below summarize the different possible outcomes of overwriting when importing multiple worksheets, starting with the most common scenario.

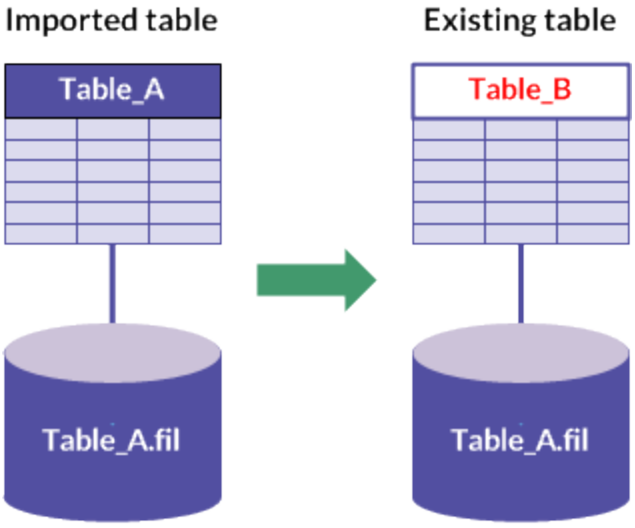
Same name: new table, existing table layout, existing source data file



	Same "Output Path" as existing source data file	Different "Output Path" from existing source data file
"Overwrite existing tables" selected	◦ existing table layout overwritten ◦ existing source data file overwritten	◦ existing table layout overwritten, linked to new source data file ◦ new source data file created ◦ existing source data file preserved, unlinked
"Overwrite existing tables" not selected	◦ new table layout and source data file created, with numeric suffix	◦ new table layout and source data file created, with numeric suffix

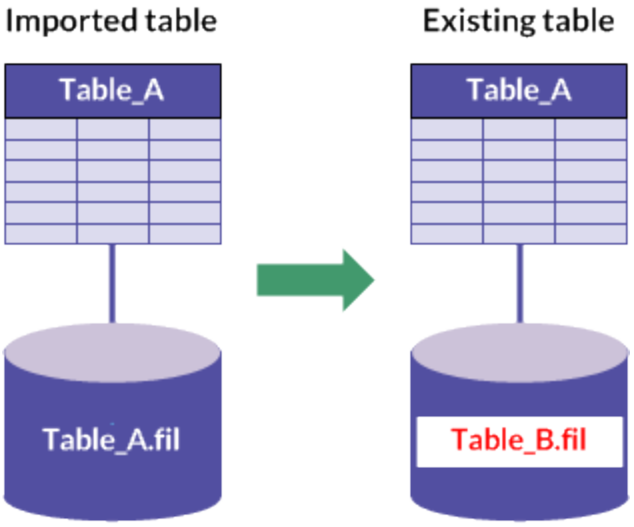
	Same "Output Path" as existing source data file	Different "Output Path" from existing source data file
	For example: • **table layout** -`Table_A2` • **source data file** -`Table_A2.fil` ◦ existing table layout and source data file preserved	For example: • **table layout** -`Table_A2` • **source data file** -`Table_A2.fil` ◦ existing table layout and source data file preserved

Different name: existing table layout



	Same "Output Path" as existing source data file	Different "Output Path" from existing source data file
"Overwrite existing tables" selected	◦ new table layout created ◦ existing source data file overwritten ◦ new and existing table layouts both linked to source data file	◦ new table layout and source data file created ◦ existing table layout and source data file preserved
"Overwrite existing tables" not selected	◦ new table layout and source data file created, with numeric suffix For example: • **table layout** -`Table_A2` • **source data file** -`Table_A2.fil` ◦ existing table layout and source data file preserved	◦ new table layout and source data file created ◦ existing table layout and source data file preserved

Different name: existing source data file



	Same "Output Path" as existing source data file	Different "Output Path" from existing source data file
"Overwrite existing tables" selected	◦ existing table layout overwritten, linked to new source data file ◦ new source data file created ◦ existing source data file preserved, unlinked	◦ existing table layout overwritten, linked to new source data file ◦ new source data file created ◦ existing source data file preserved, unlinked
"Overwrite existing tables" not selected	◦ new table layout and source data file created, with numeric suffix For example: • **table layout** -`Table_A2` • **source data file** -`Table_A2.fil` ◦ existing table layout and source data file preserved	◦ new table layout and source data file created, with numeric suffix For example: • **table layout** -`Table_A2` • **source data file** -`Table_A2.fil` ◦ existing table layout and source data file preserved

Import a single Excel worksheet

Import a single Excel worksheet, or named range, to create a new Analytics table. You have the option of manually defining the source Excel data during the import process.

Show me how

> **Note**
> Make sure the Excel file is closed before you begin the import process.

Locate and select the Excel file

1. Select **Import > File**.
2. In the **Select File to Define** dialog box, locate and select the Excel file and click **Open**.

   Microsoft Excel files have an `.xlsx` or an `.xls` file extension.
3. In the **File Format** page, verify that the **Excel file** option has been selected and click **Next**.

Specify the worksheet to import

1. In the **Data Source** page, select the worksheet or the named range to import.

   > **Note**
   > To see any named ranges, deselect **System Table Only**.

   Worksheets are identified with a dollar sign ($) appended to the worksheet name. The dollar sign is added temporarily, and does not appear in the Analytics table name.
2. Review the default settings on the page, make any required updates, and click **Next**.

Setting	Description
Use first row as Field Names	Values in the first row in the worksheet or the named range are used as field names in the Analytics table. **Note** If you use this setting, the row used as field names is whatever line number is specified in **Start On Line**.
Start On Line	The line number on which to start reading the worksheet. This setting allows you to skip lines at the beginning of a worksheet that contain information you do not want to import. For example, if the first three lines of a worksheet contain header information, enter 4 to start reading data on the fourth line. **Note** The start line for a named range is always the first line in the named range, regardless of the **Start On Line** setting.
Import all fields as character type	Assigns the Character data type to all the imported fields.

Setting	Description
	Tip Assigning the Character data type to all the imported fields simplifies the process of importing Excel files. Once the data is in Analytics, you can assign different data types, such as Numeric or Datetime, to the fields, and specify format details. **Import all fields as character type** is useful if you are importing a table with identifier fields automatically assigned the Numeric data type by Analytics when in fact they should use the Character data type.
First 100 records	Analytics uses only the first 100 records in the worksheet or the named range to determine the data type of fields, and the length of fields, in the Analytics table. With large Excel files, using **First 100 records** significantly speeds up the import process. **Caution** Select this option only if you are confident that values in the first 100 records accurately reflect the data type and length of all subsequent values. If any values after the first 100 records are of a different data type, or are longer, the resulting Analytics table will contain inaccurate or truncated data. Inaccurate or truncated data in critical fields will very likely invalidate the results of subsequent data analysis.
Entire Excel Worksheet or Named Range	Analytics uses all the records in the worksheet or the named range to determine the data type of fields, and the length of fields, in the Analytics table. With large Excel files, using all the records to determine data type and field length significantly slows down the import process. **Note** Select this option if you are uncertain about the consistency of the data types or the length of values in the Excel data.

Edit the Analytics field properties

Analytics makes a best guess about the properties associated with each field in the Excel data. You can accept the default settings, or follow the steps below to manually define the fields.

1. In the **Excel Import** page, select each column heading in the preview table to see the properties associated with the field.
2. For each field, review the settings assigned by Analytics to the properties listed below, and make any required updates.
3. When you have finished reviewing and editing properties, click **Next**.

Property	Description
Ignore this field	The data in the field is not imported.
Name	The name for the field in the table layout. You can keep the name assigned by Analytics, or enter a different name.
Column Title	The column title for the field in the default Analytics view. If you do not specify a column title, the **Name** value is used.
Length	The length of the field in the table layout. Specify the length in characters. If a datetime field has no time data and displays 00:00:00 after the date, you can shorten the length of the field to omit the empty time data. **Note** Maximum field length is 32,767 characters (non-Unicode edition) or 16,383 characters (Unicode edition). The entire field length, up to the maximum, is imported into Analytics, but only the first 256 characters are displayed in the table view. The remainder of the data is present, and can be analyzed, but it is not visible in the view. To see all the data, open the **Table Layout** dialog box. **Tip** Increase the length of a field if you selected **First 100 records** in the previous screen, but you are uncertain about the length of subsequent values in the field.
Note If you selected **Import all fields as character type** in the **Data Source** page, the options below do not apply and are disabled.	
Type	The data type assigned to the field in Analytics. You can keep the data type assigned by Analytics, or select an appropriate data type from the drop-down list.
Value	A read-only property that displays the first value in the field. The value dynamically updates based on any edits you make.
Decimal	Numeric fields only. The number of decimal places in the source data. **Note** The **Decimal** text box appears automatically when you select a Numeric data type.
Input Format	Datetime fields only. The format of datetime values in the source data. Select a format that matches the data, or if necessary create a format to match the data. The format you specify must exactly match the format in the source data. For more information about date and time formats, see "Formats of date and time source data" on page 347.

Property	Description
	Note The **Input Format** text box appears automatically when you select a Datetime data type.

Save the Analytics data file

In the **Save Data File As** dialog box, enter a name for the Analytics data file and click **Save**.

If Analytics prefills a data file name, you can accept the prefilled name, or change it.

You can also navigate to a different folder to save the data file if you do not want to use the default location opened by Analytics.

Finalize the import

1. In the **Final** page, verify the settings for the new Analytics table and click **Finish**.

   If you want to make any changes, click **Back** to get to the appropriate page in the wizard.
2. Enter a name for the table layout that you are adding to the project, or keep the default name, and click **OK**.

   The new Analytics table is created with data from the imported file.

Import multiple Excel worksheets

In a single operation, import multiple Excel worksheets, or named ranges, from an Excel file, or from multiple Excel files. Once imported to an Analytics project, each worksheet or named range becomes a separate Analytics table.

Analytics automatically defines the source Excel data and no manual definition is possible during the import.

Once the data is in Analytics, you can make any necessary adjustments to the data definition in the **Table Layout** dialog box.

Show me how

> **Note**
>
> Make sure all Excel files are closed before you begin the import process.
>
> All first rows in the worksheets and named ranges that you import should use a consistent approach. First rows should be either field names, or data, across all data sets. Avoid mixing the two approaches in a single import operation.
>
> If the data sets have an inconsistent approach to first rows, use two separate import operations.

Locate and select the Excel file or files

1. Select **Import > File**.
2. In the **Select File to Define** dialog box, locate and select the Excel file or files and click **Open**.

   Microsoft Excel files have an `.xlsx` or an `.xls` file extension.

   You can **Shift+click** to select multiple adjacent files, or **Ctrl+click** to select multiple non-adjacent files.
3. In the **File Format** page, verify that the **Excel file** option has been selected and click **Next**.

Specify the worksheets to import

1. In the **Data Source** page, select the worksheets or the named ranges to import.

   > **Note**
   >
   > To see any named ranges, deselect **System Table Only**.

   Select individual worksheets or named ranges, or select the first checkbox if you want to select all the worksheets and named ranges in the Excel file or files.

   Worksheets are identified with a dollar sign ($) appended to the worksheet name. The dollar sign is added temporarily, and does not appear in the resulting Analytics table name.
2. Review the settings assigned by Analytics, make any required updates, and click **Next**.

Setting	Description
Table Name	The name for the table in the Analytics project. Keep the name assigned by Analytics, or double-click a table name, type a different name, and press Enter. **Note** The table name applies to both the new table layout and the new source data file created when importing the data.
Use first row as Field Names	Values in the first row in each worksheet or named range are used as field names in the resulting table layouts.

Setting	Description
	Note If you use this setting, the row used as field names is whatever line number is specified in **Start On Line**. This setting applies globally to all worksheets and named ranges that you import.
Overwrite existing tables	Existing tables with identical names in the Analytics project are overwritten. For detailed information, see "How overwriting works" on page 236.
Start On Line	The line number on which to start reading the worksheets. This setting allows you to skip lines at the beginning of worksheets that contain information you do not want to import. For example, if the first three lines of each worksheet contain header information, enter 4 to start reading data on the fourth line. **Note** The start line for a named range is always the first line in the named range, regardless of the **Start On Line** setting.
Include File Name	Prepend the Excel file name to the name of the Analytics table or tables. **Tip** If worksheets in different files have the same name, prepending the Excel file name allows you to avoid table name conflicts.
Import all fields as character type	Assigns the Character data type to all the imported fields. **Tip** Assigning the Character data type to all the imported fields simplifies the process of importing Excel files. Once the data is in Analytics, you can assign different data types, such as Numeric or Datetime, to the fields, and specify format details. **Import all fields as character type** is useful if you are importing a table with identifier fields automatically assigned the Numeric data type by Analytics when in fact they should use the Character data type.
First 100 records	Analytics uses only the first 100 records in the worksheet or named range to determine the data type of fields, and the length of fields, in the resulting Analytics tables. With large Excel files, using **First 100 records** significantly speeds up the import process.

Setting	Description
	Caution Use this option only if you are confident that values in the first 100 records accurately reflect the data type and length of all subsequent values. If any values after the first 100 records are of a different data type, or are longer, the resulting Analytics table will contain inaccurate or truncated data. Inaccurate or truncated data in critical fields will very likely invalidate the results of subsequent data analysis.
Entire Excel Worksheet or Named Range	Analytics uses all the records in the worksheet or named range to determine the data type of fields, and the length of fields, in the resulting Analytics tables. With large Excel files, using all the records to determine data type and field length significantly slows down the import process. **Note** Use this option if you are uncertain about the consistency of the data types or the length of values in worksheet columns.
Output Path	Specifies the folder where the new Analytics data files (.fil) are saved. If you leave **Output Path** blank, the Analytics data files are saved in the folder containing the Analytics project.

Finalize the import

In the **Final** page, verify the settings for the new Analytics tables and click **Finish**.

If you want to make any changes, click **Back** to get to the appropriate page in the wizard.

The new Analytics tables are created with data from the imported worksheets or named ranges.

> **Note**
> If a numeric suffix has been added to a **Table Name** in the **Final** page, a table with the same name already exists in the Analytics project and you have chosen not to overwrite existing tables.
> For detailed information, see "How overwriting works" on page 236.

Import a Microsoft Access database file

You can create an Analytics table by defining and importing a Microsoft Access database file.

The Access file can be any version from Access 2000 to Access 2010. To import a file from an earlier version of Access you can save the file in another file format that Analytics can define and import.

You can import an Access file even if you do not have Microsoft Access installed on your computer.

1. Select **File > New > Table**.
2. If the **Select Platform for Data Source** page is displayed, select **Local** and click **Next**.
3. Select **File** and click **Next**.
4. In **Select File to Define**, locate and select the file you want to create the Analytics table from and click **Open**. Microsoft Access database files have a **.mdb** or **.accdb** file extension.
5. In the **File Format** page, verify that the **Access database** option has been selected and click **Next**.
6. Complete the following steps in the **Data Source** page:
   a. Select the table or view to import. The available options are listed in the **Select the Access Table/View** list.
   b. If you want to increase or decrease the maximum number of characters imported from text fields, enter a new value in the **Maximum Character Field Length** text box.

      You can enter from 1 to 255 characters.

   c. If you want to increase or decrease the maximum number of characters imported from memo or long text fields, enter a new value in the **Maximum Memo Field Length** text box.

      You can enter from 1 to 32767 characters (non-Unicode Analytics), or 16383 characters (Unicode Analytics).

   d. Click **Next**.
7. In the **Save Data File As** dialog box, modify the filename and location for the Analytics data file, as necessary, and click **Save**.
8. In the **Final** page, verify the settings for the new Analytics table and click **Finish**.
9. Enter a name for the Analytics table you are adding to your project and click **OK**.

Import a delimited text file

Import a delimited text file to Analytics for analysis using a variety of different tools.

How it works

You use the Data Definition Wizard to select one or more delimited text files and import the data to Analytics. The imported data creates one or more new Analytics tables and associated data files (.fil). Each imported delimited text file creates a separate Analytics table.

The Analytics data file contains a copy of the delimited data that is completely separate from the original delimited text file.

You can import delimited text files located on your local computer or on a network drive. Users of Analytics Exchange can also access delimited text files located on an Analytics Server.

Import a single file or multiple files

You have the option of importing a single delimited text file or multiple files in a single operation. The import process differs somewhat, depending on which option you use:

- **single file** – you have the option of manually defining both file-level and field-level properties during the import process
- **multiple files** – you can manually define only file-level properties during the import process. Analytics automatically defines field-level properties and no manual definition is possible during the import

  For example, during the import of multiple files, you **cannot**:

  - specify the data type of fields
  - selectively exclude fields from the import

  Once the data is in Analytics, you can make any necessary adjustments to the data definition in the **Table Layout** dialog box.

Combine multiple files

After you import multiple delimited text files into individual Analytics tables you might want to combine them into a single Analytics table. For example, you could combine the data from twelve monthly tables into a single annual table containing all the data. You can combine the files only after you have imported them into individual Analytics tables.

For information about combining multiple Analytics tables, see "Appending tables" on page 858.

> **Tip**
> To reduce labor, try combining the multiple tables first before making any required adjustments to the data definition in the new combined table.

The structure of delimited text files

Delimited text files typically have a .txt or .csv file extension, although other file extensions are possible. Delimited text files are often used to import data from spreadsheet or database applications into Analytics. Each spreadsheet or database row becomes a row in the delimited text file, with each row or record separated by a line separator. The valid line separators are:

- **CR** – carriage-return
- **LF** – line-feed
- **CRLF** – carriage-return line-feed (the standard DOS/Windows character sequence)

Field separator character

Fields in each record in the delimited text file are separated by a field separator character. There are three main types of delimited text file, based on the field separator character they use:

- **Comma separated values (.csv)** – Commas are used to delimit the fields in each record.
- **Tab separated values** – Tabs are used to delimit the fields in each record.
- **Text files (.txt)** – Commas, tabs, or another field separator character are used to delimit the fields in each record. Other common field separator characters are spaces, pipes (|), and semicolons (;).

Text qualifier character

If a field separator character is used, a text qualifier character is also used to enclose character field values and isolate them from field separators. Common text qualifier characters are double (" ") or single (' ') quotation marks.

For example, if a comma is the field separator character, enclosing the value $1,000 in text qualifiers ("$1,000") ensures that the value is interpreted as a single value and not as two values ($1 and 000).

Example of a delimited text file

The example below shows the first four rows in a delimited text file.

- The first row contains the field names.
- The field separator is a comma. Each row includes seven fields separated by commas.
- The text qualifiers are double quotation marks. The last field includes a text qualifier, so that the comma in the dollar value is not interpreted as a field separator.

```
First_Name,Last_Name,CardNum,EmpNo,HireDate,Salary,Bonus_2011
Lila,Remlawi,8590122497663807,000008,12/28/2007,52750,"$1,405.40"
Vladimir,Alexov,8590122281964011,000060,10/5/2007,41250,"$4,557.43"
Alex,Williams,8590124253621744,000104,8/12/2010,40175,"$7,460.02"
```

How overwriting works

If you import delimited data and create a new Analytics table that has the same name as an existing table in the Analytics project you overwrite the existing table.

Show me more

Both parts of a table overwritten

Overwriting of Analytics tables is complicated by the fact that both parts of a table can be overwritten:

- the table layout, displayed in the **Navigator**
- the associated source data file, stored in a Windows folder

(For information about table layouts and source data files, see "The structure of Analytics tables" on page 115.)

The two parts of the table are overwritten independently of each other. If both parts have the same name as the new table, both are overwritten. This scenario is the most common.

But if the table layout and the source data file have different names, only the one with the same name as the new table is overwritten.

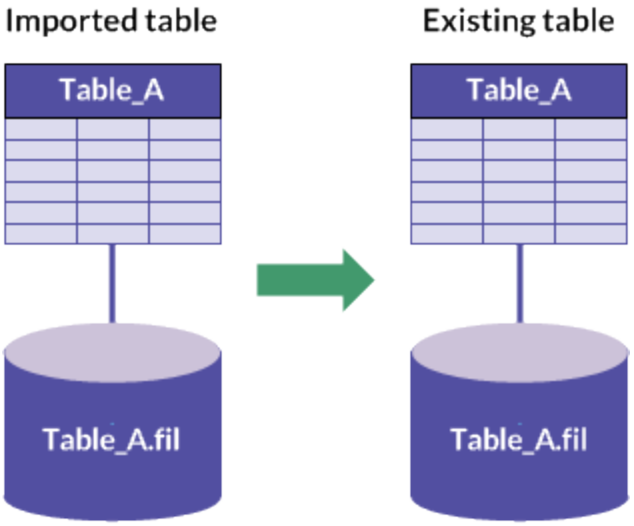
This overwrite behavior applies whether you are importing a single delimited text file, or multiple files.

Overwriting when importing multiple delimited text files

When you import multiple delimited text files, overwriting also depends on both the **Overwrite existing tables** and **Output Path** settings.

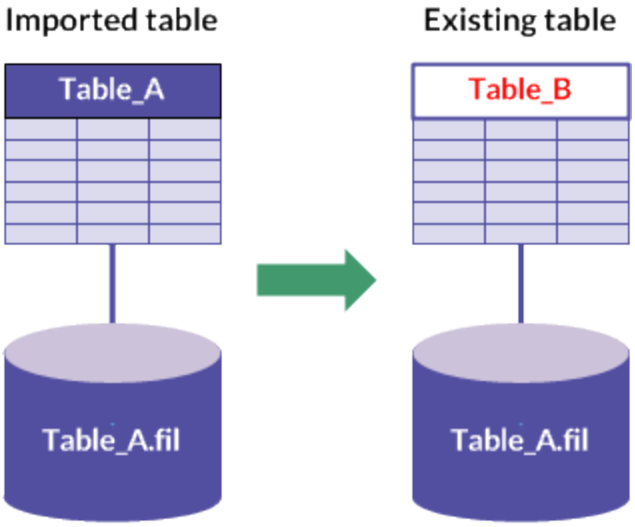
The sections below summarize the different possible outcomes of overwriting when importing multiple delimited text files, starting with the most common scenario.

Same name: new table, existing table layout, existing source data file



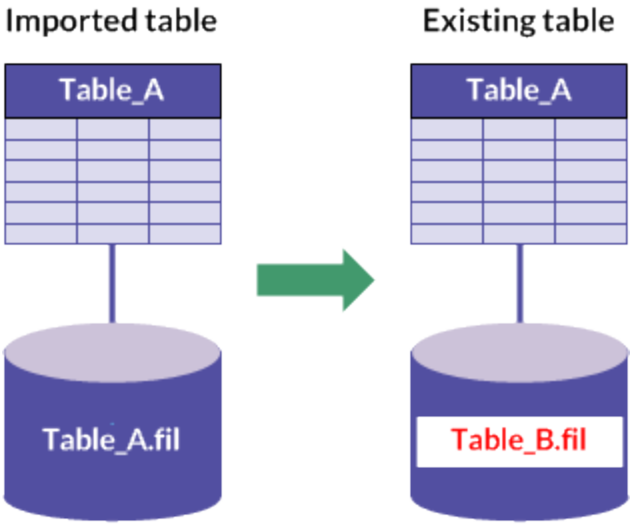
	Same "Output Path" as existing source data file	Different "Output Path" from existing source data file
"Overwrite existing tables" selected	◦ existing table layout overwritten ◦ existing source data file overwritten	◦ existing table layout overwritten, linked to new source data file ◦ new source data file created ◦ existing source data file preserved, unlinked
"Overwrite existing tables" not selected	◦ error message "Existing file or table names detected" appears	◦ error message "Existing file or table names detected" appears

Different name: existing table layout



	Same "Output Path" as existing source data file	Different "Output Path" from existing source data file
"Overwrite existing tables" selected	◦ new table layout created ◦ existing source data file overwritten ◦ new and existing table layouts both linked to source data file	◦ new table layout and source data file created ◦ existing table layout and source data file preserved
"Overwrite existing tables" not selected	◦ error message "Existing file or table names detected" appears	◦ new table layout and source data file created ◦ existing table layout and source data file preserved

Different name: existing source data file



	Same "Output Path" as existing source data file	Different "Output Path" from existing source data file
"Overwrite existing tables" selected	◦ existing table layout overwritten, linked to new source data file ◦ new source data file created ◦ existing source data file preserved, unlinked	◦ existing table layout overwritten, linked to new source data file ◦ new source data file created ◦ existing source data file preserved, unlinked
"Overwrite existing tables" not selected	◦ error message "Existing file or table names detected" appears	◦ error message "Existing file or table names detected" appears

Import a single delimited text file

Import a single delimited text file to create a new Analytics table. You have the option of manually defining both file-level and field-level properties during the import process.

Show me how

Locate and select the delimited file

1. Select **Import > File**.
2. In the **Select File to Define** dialog box, locate and select the delimited text file and click **Open**.

   Delimited text files can have several different file extensions, including `.txt` and `.csv`.

Specify the delimited file properties

1. In the **Character Set** page, verify that the correct character set option is selected and click **Next**.
2. In the **File Format** page, verify that **Delimited text file** is selected and click **Next**.
3. In the **Delimited File Properties** page, review the settings assigned by Analytics to the properties listed below, make any required updates, and click **Next**.

Property	Description
Start on Line	The line number on which to start reading the file. This setting allows you to skip lines at the beginning of a file that contain information you do not want to import. For example, if the first three lines of a file contain header information, enter 4 to start reading data on the fourth line.
Field Width	For the selected column heading in the preview table, specifies the length of the field in the resulting table layout. Specify the length in characters. You can keep the length assigned by Analytics, or enter a different length. **Note** Maximum field length is 32,767 characters (non-Unicode edition) or 16,383 characters (Unicode edition). The entire field length, up to the maximum, is imported into Analytics, but only the first 256 characters are displayed in the table view. The remainder of the data is present, and can be analyzed, but it is not visible in the view. To see all the data, open the **Table Layout** dialog box. **Tip** If you intended to periodically refresh the resulting Analytics table from updated source data, or re-use the import command, enter a longer field length than the one assigned by Analytics. A longer field length provides extra space if updated values in the source data are longer than any of the current values. Values that exceed the available field length are truncated.
Use first row as field names	Values in the first line in the file are used as field names in the resulting table layout. **Note** If you use this setting, the row used as field names is whatever line number is specified in **Start on Line**. If the field names are not correct, you can update them in a subsequent page in the **Data Definition Wizard**.
Treat Consecutive qualifiers as one	Duplicate qualifier characters are ignored. For example, “ACL Services Ltd. dba Galvanize”” (terminating with two double quotation marks) is equivalent to “ACL Services Ltd. dba Galvanize” if you select this option.
Field Separator	The character that separates fields in the file:

Property	Description
	◦ **Comma** ◦ **TAB** ◦ **Semicolon** ◦ **Other** - allows you to specify the character that is used as the field separator
Text Qualifier	The text symbol that identifies values contained in fields: ◦ **Double Quote** ◦ **Single Quote** ◦ **None** - indicates that no text qualifier is used ◦ **Other** - allows you to specify the character that is used as the text qualifier
Clear CR and **Clear LF**	Cleanses the imported data of misplaced carriage return (CR) and/or line feed (LF) characters. Misplaced CR/LF characters can cause incorrect line breaks within records. When enabled, the option replaces any CR/LF characters with a space. Only CR/LF characters that occur inside a pair of text qualifiers are replaced. For Windows files, select both **Clear CR** and **Clear LF**. The two options are disabled if **Text Qualifier** is **None**.
All Character	Assigns the Character data type to all the imported fields. **Tip** Assigning the Character data type to all the imported fields simplifies the process of importing delimited text files. Once the data is in Analytics, you can assign different data types, such as Numeric or Datetime, to the fields, and specify format details. The **All Character** option is useful if you are importing a table with identifier fields automatically assigned the Numeric data type by Analytics when in fact they should use the Character data type.
Replace NULLs	Cleanses the imported data of misplaced NUL characters. Misplaced NUL characters can cause gaps and incorrect field divisions within records. When enabled, the option replaces any NUL characters with a space.

Save the Analytics data file

In the **Save Data File As** dialog box, enter a name for the Analytics data file and click **Save**.

If Analytics prefills a data file name, you can accept the prefilled name, or change it.

You can also navigate to a different folder to save the data file if you do not want to use the default location opened by Analytics.

Edit the Analytics field properties

In the **Edit Field Properties** page, review the settings assigned by Analytics to the properties listed below, make any required updates, and click **Next**.

> **Note**
> Select a column heading in the preview table to see the properties associated with the column.

Property	Description
Ignore this field	Excludes the field from the resulting table layout. The data in the field is still imported, but it is undefined, and does not appear in the new Analytics table. It can be defined later, if necessary, and added to the table.
Name	The name for the field in the table layout. You can keep the name assigned by Analytics, or enter a different name.
Column Title	The column title for the field in the default Analytics view. If you do not specify a column title, the **Name** value is used.
> **Note** If you selected **All Character** in the **Delimited File Properties** page, the options below do not apply and are disabled.	
Type	The data type assigned to the field in the resulting Analytics table. You can keep the data type assigned by Analytics, or select an appropriate data type from the drop-down list. For information about the supported data types in Analytics, see "Data types in Analytics" on page 739.
Value	A read-only property that displays the first value in the field. The value dynamically updates based on any edits you make.
Decimal	Numeric fields only. The number of decimal places in the source data. > **Note** The **Decimal** text box appears automatically when you select a Numeric data type.
Input Format	Datetime fields only. The format of datetime values in the source data. The format you specify must exactly match the format in the source data. For more information about date and time formats, see "Formats of date and time source data" on page 347.

Finalize the import

1. In the **Final** page, verify the settings for the new Analytics table and click **Finish**.
   If you want to make any changes, click **Back** to get to the appropriate page in the wizard.

2. Enter a name for the table layout that you are adding to the project, or keep the default name, and click **OK**.

   The new Analytics table is created with data from the imported file.

Import multiple delimited text files

In a single operation, import multiple delimited text files. Once imported to an Analytics project, each delimited file becomes a separate Analytics table.

You can manually define only file-level properties during the import process. Analytics automatically defines field-level properties and no manual definition is possible during the import.

Once the data is in Analytics, you can make any necessary adjustments to the data definition in the **Table Layout** dialog box.

Show me how

> **Note**
>
> All first rows in the files that you import should use a consistent approach. First rows should be either field names, or data, across all files. Avoid mixing the two approaches in a single import operation.
>
> If the files have an inconsistent approach to first rows, use two separate import operations.

Locate and select the delimited files

1. Select **Import > File**.
2. In the **Select File to Define** dialog box, locate and select the delimited text files and click **Open**.

   Delimited text files with the following file extensions are supported: `.txt`, `.csv`, `.del`, `.dat`

   You can **Shift+click** to select multiple adjacent files, or **Ctrl+click** to select multiple non-adjacent files.

Make the initial import preparations

1. In the **Delimited File Properties** page, select the files to import.

   Keep the files selected by default, or deselect any files you do not want to import. Select the first checkbox if you want to deselect or select all the files.

2. Review the settings assigned by Analytics, make any required updates, and click **Next**.

Setting	Description
Table Name	The name for the table in the Analytics project. Keep the name assigned by Analytics, or double-click a table name, type a different

Setting	Description
	name, and press Enter. **Note** The table name applies to both the new table layout and the new source data file created when importing the data.
Overwrite existing tables	Existing tables with identical names in the Analytics project are overwritten. For detailed information, see "How overwriting works" on page 250.
Output Path	Specifies the folder where the new Analytics data files (.fil) are saved. If you leave **Output Path** blank, the Analytics data files are saved in the folder containing the Analytics project.

3. If the error message "Existing file or table names detected" appears, click **OK** and do one or both of the following:
   - Select **Overwrite existing tables** if any existing table layouts or associated data files with identical names can be overwritten.
   - In the **Table Name** setting, rename imported tables as required to avoid overwriting any existing table layouts or associated data files.
4. In the confirmation dialog box, click **Yes** to continue, or **No** to go back and change the selection of files.

Specify the delimited file properties

> **Note**
> The properties you specify apply to all files being imported. If the files are inconsistently structured, the properties will not be accurate for all files and there could be problems with the import.

1. In the **Delimited File Properties** page, review the settings assigned by Analytics to the properties listed below, make any required updates, and click **Next**.

Property	Description
Start on Line	The line number on which to start reading the files. This setting allows you to skip lines at the beginning of files that contain information you do not want to import. For example, if the first three lines of each file contain header information, enter 4 to start reading data on the fourth line.
Field Width	For the selected column heading in the preview table, specifies the length of the field in the resulting table layout. Specify the length in characters. You can keep the length assigned by Analytics, or enter a different length.

Property	Description
	Note Maximum field length is 32,767 characters (non-Unicode edition) or 16,383 characters (Unicode edition). The entire field length, up to the maximum, is imported into Analytics, but only the first 256 characters are displayed in the table view. The remainder of the data is present, and can be analyzed, but it is not visible in the view. To see all the data, open the **Table Layout** dialog box. **Tip** If you intended to periodically refresh a resulting Analytics table from updated source data, or re-use the import command, enter a longer field length than the one assigned by Analytics. A longer field length provides extra space if updated values in the source data are longer than any of the current values. Values that exceed the available field length are truncated.
Use first row as field names	Values in the first line in each file are used as field names in the resulting table layouts. **Note** If you use this setting, the row used as field names is whatever line number is specified in **Start on Line**. This setting applies globally to all files that you import.
Treat Consecutive qualifiers as one	Duplicate qualifier characters are ignored. For example, “ACL Services Ltd. dba Galvanize”” (terminating with two double quotation marks) is equivalent to “ACL Services Ltd. dba Galvanize” if you select this option.
Field Separator	The character that separates fields in the files: ◦ **Comma** ◦ **TAB** ◦ **Semicolon** ◦ **Other** – allows you to specify the character that is used as the field separator
Text Qualifier	The text symbol that identifies values contained in fields: ◦ **Double Quote** ◦ **Single Quote** ◦ **None** – indicates that no text qualifier is used ◦ **Other** – allows you to specify the character that is used as the text qualifier
Clear CR and **Clear LF**	Cleanses the imported data of misplaced carriage return (CR) and/or line feed (LF) characters. Misplaced CR/LF characters can cause incorrect line breaks within records. When enabled, the option replaces any CR/LF characters with a space. Only CR/LF characters that occur inside a pair of text qualifiers are replaced. For Windows files, select both **Clear CR** and **Clear LF**. The two options are disabled if **Text Qualifier** is **None**.

Property	Description
All Character	Assigns the Character data type to all the imported fields. **Tip** Assigning the Character data type to all the imported fields simplifies the process of importing delimited text files. Once the data is in Analytics, you can assign different data types, such as Numeric or Datetime, to the fields, and specify format details. The **All Character** option is useful if you are importing a table with identifier fields automatically assigned the Numeric data type by Analytics when in fact they should use the Character data type.
Replace NULLs	Cleanses the imported data of misplaced NUL characters. Misplaced NUL characters can cause gaps and incorrect field divisions within records. When enabled, the option replaces any NUL characters with a space.

Finalize the import

In the **Final** page, verify the settings for the new Analytics tables and click **Finish**.

If you want to make any changes, click **Back** to get to the appropriate page in the wizard.

The new Analytics tables are created with data from the imported files.

Defining and importing print image (report) files and PDF files

Print image files, also called report files, are electronic copies of printed reports. Adobe PDF files are application files or scanned files that have been saved in the standard PDF format. The way you define and import print image files or PDF files is almost identical. For that reason, most of the topics in this section apply to both file types.

PDFs can be more challenging to define and import than print image files because data columns that appear to be aligned in the source PDF may become misaligned once Analytics has parsed the PDF (part of the file definition process). Analytics includes two PDF parsers: Xpdf and VeryPDF. You can try using both parsers to see if one gives better results.

> **Caution**
> Use control totals to verify that the Analytics table created from an imported print image or PDF file contains all the data from the source file. Unintentionally excluding records is a possibility when defining print image or PDF files. You should always verify that you have a complete data set in Analytics before beginning any analysis.

Key points for successfully defining a print image or PDF file

Defining a print image or PDF file can be tricky. It is more of an art than a science, one that can require you to carefully analyze the arrangement of the data in the source file in order to plan an effective approach. Arrangements of data in print image or PDF files are typically less standardized than arrangements in other file formats, which complicates the definition process. Success can require an iterative process.

There are a number of key points or techniques, outlined below, that can help you avoid frustration. It is recommended that you review these points carefully before, or as, you define a file, or if you encounter problems.

General points

- "The file definition process is iterative" on the next page
- "You will get better with practice" on the next page

Misaligned data

- "Workarounds for misaligned data" on page 263

Fields and records

- "Fields are blue, records are gray, and undefined data is white" on page 264
- "You can define three kinds of data: detail, header, and footer" on page 265

- "Do not select field names in the source file" on page 268

Capturing records

- "Specify a unique value to capture a set of records" on page 268
- "Tips for choosing a unique value" on page 269
- "Precisely capture a set of records" on page 270
- "Use multiple criteria to capture a set of records" on page 273
- "Check record definitions and field definitions throughout the entire file" on page 274
- "You can define multiline records and fields" on page 274

Additional considerations

- "Define and import only data you need" on page 274
- "Control the order of fields in the resulting Analytics table" on page 274
- "Analytics may auto-define a file" on page 275
- "Use control totals to verify the resulting Analytics table" on page 275

General points

The file definition process is iterative

Successfully defining a print image or PDF file is typically an iterative process and may require a certain amount of trial and error. You will need to perform some or all of the following individual tasks:

- define one or more fields
- define a set of detail records based on a unique value
- define one or more header or footer records
- modify or further build criteria to fine tune a captured set of records
- review each field and record definition for accuracy
- edit inaccurate field or record definitions
- make multiple passes through a file as one way of dealing with misaligned data

You will get better with practice

Defining print image or PDF files may initially seem quite difficult – especially defining files with misaligned data. With practice, you will get better at assessing the structure of the data in a source file, and finding appropriate methods for defining it.

You can use two of the sample files included with Analytics to practice:

- `REPORT3.TXT` is easier to define. Analytics auto-defines the detail records in the file, however you need to edit the auto-definition because it contains errors.
- `Inventory.pdf` is more challenging to define because it contains misaligned data (unless you parse the file page by page). Analytics is unable to auto-define any part of the misaligned file, so you must create a manual definition from scratch.

Analytics auto-defines the detail fields and records in another sample file, `Report.txt`, perfectly. You may find it useful to study the auto-definition of `Report.txt` in the **Data Definition Wizard**.

Misaligned data

Workarounds for misaligned data

In the **Data Definition Wizard**, misaligned data columns in a parsed PDF or print image file (see "Aligned and misaligned data in a parsed PDF file" below) can make it difficult or labor-intensive to create an Analytics table that is usable. If misaligned data is a significant problem, consider any of the following approaches.

> **Note**
> The most suitable approach for your situation depends on the nature of the data you are trying to define, and your experience with Analytics. New users of Analytics should consider asking for the data in a different format.

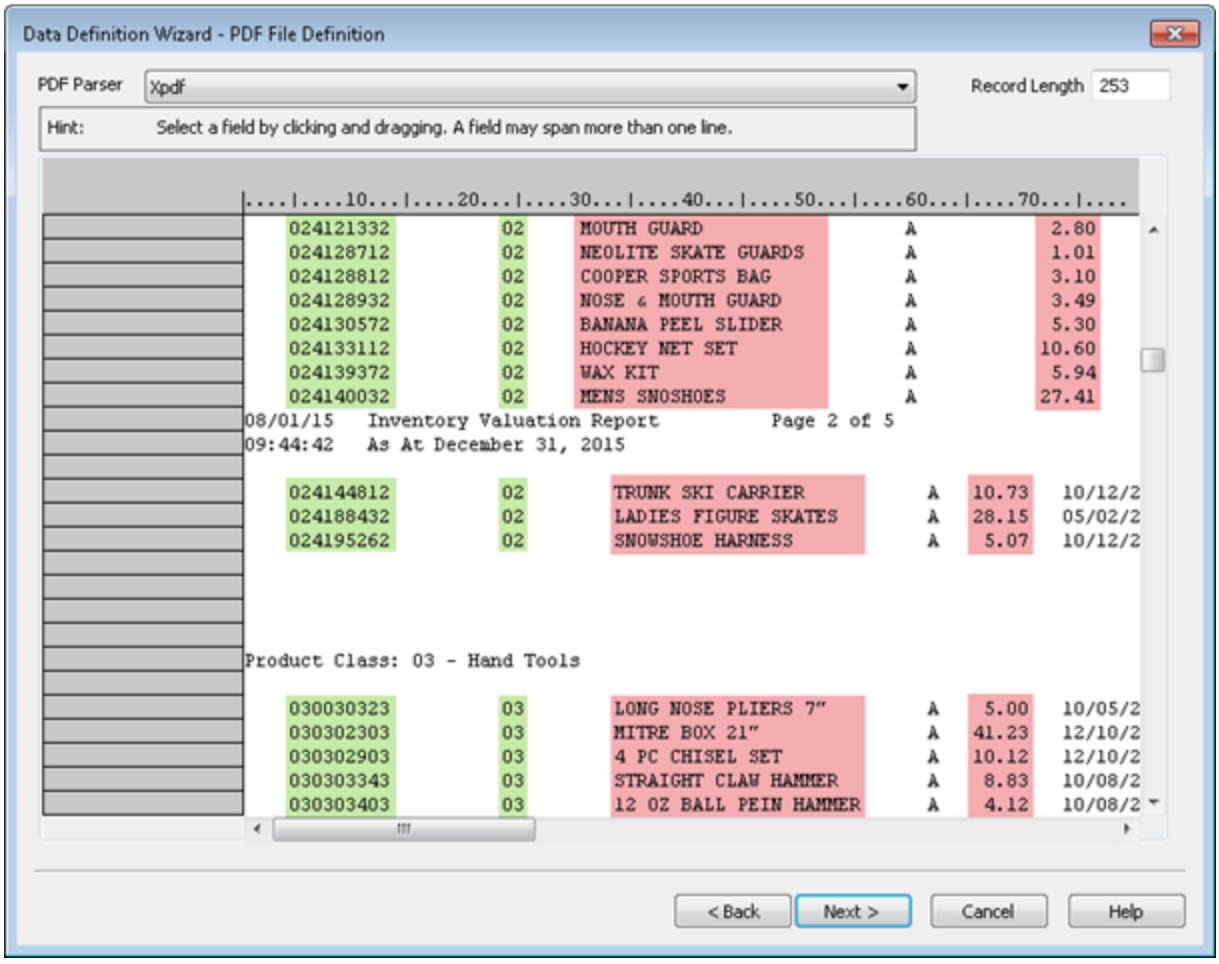
- Go back to the source of the file and ask for the data in a different format.
- Try converting the file using conversion software – for example, software that converts a PDF file to an Excel file or a text file. Import the converted file to Analytics.
- Try copying and pasting PDF data into a text editor. Then import the text file to Analytics.
- Use one or more of the following techniques to define misaligned fields:
  - Create a field definition that is long enough to capture the leftmost and the rightmost characters in a misaligned field.
  - Create overlapping field definitions.
  - Create a single, long field definition that encompasses multiple misaligned fields.

  For more information, see "Defining misaligned fields in a print image or PDF file" on page 308.
- Import the source file more than once. With each import, define a different subset of records. Append the resulting Analytics tables to assemble a complete data set.

  For more information, see "Defining and importing subsets of print image or PDF data" on page 312.

Aligned and misaligned data in a parsed PDF file

The two leftmost data columns in the parsed PDF file shown below are aligned. The remainder of the data columns are misaligned.



Fields and records

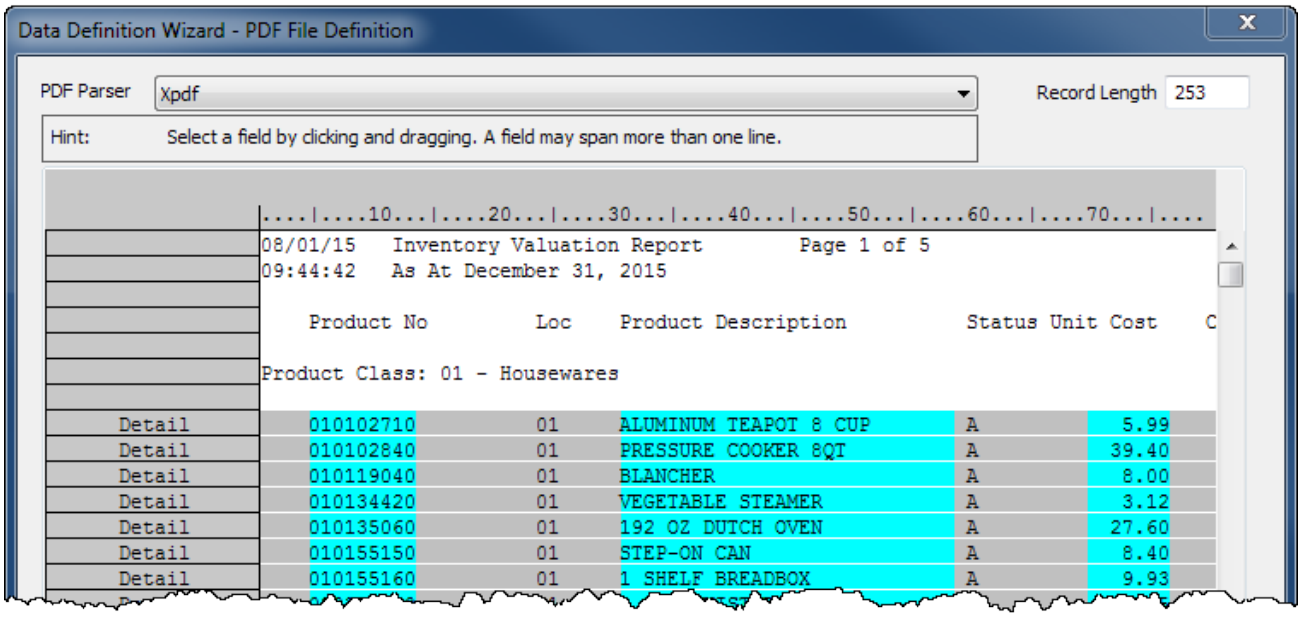
Fields are blue, records are gray, and undefined data is white

As you use the **Data Definition Wizard** to define fields and records in a print image or parsed PDF file, three colors indicate the status of the data:

- **Aqua-blue** highlighting indicates that the data is part of a defined field. All defined fields are also part of a defined record.
- **Gray** highlighting indicates that the data is part of a defined record, but is not part of a defined field.
- **White** background indicates that the data is completely undefined.

> **Note**
>
> Only aqua-blue highlighted fields become part of the resulting Analytics table.
>
> Gray highlighted data in a defined record is ignored unless it is also defined as a field. The gray portions of a record between defined fields are omitted in the resulting Analytics table.
>
> Completely undefined data is ignored. If you want to include any of this data in the resulting Analytics table, you must define additional fields and records.

Defined fields, defined record, and undefined data



You can define three kinds of data: detail, header, and footer

In the **Data Definition Wizard**, you can define three kinds of data in a print image or PDF file.

Kind of data	Description	Example	Location in "The different kinds of data in a PDF file" on the next page
Detail data	The basic content of a file, arranged in records. Defining detail data is mandatory. You cannot define a print image or PDF file without defining detail data.	◦ credit card transactions ◦ inventory records	#2, outlined in blue

Kind of data	Description	Example	Location in "The different kinds of data in a PDF file" on the next page
Header data	The identifying information that appears above blocks or subsets of detail records. Defining header data is optional. If you do not need the header information, you do not need to define it.	◦ store number and location where the credit card transactions took place ◦ “Product Class” information	#1, outlined in red
Footer data	The information that appears below blocks or subsets of detail records. Defining footer data is optional. If you do not need the footer information, you do not need to define it.	◦ subtotaled credit card transactions by store ◦ “Class Totals”	#3, outlined in aqua-blue

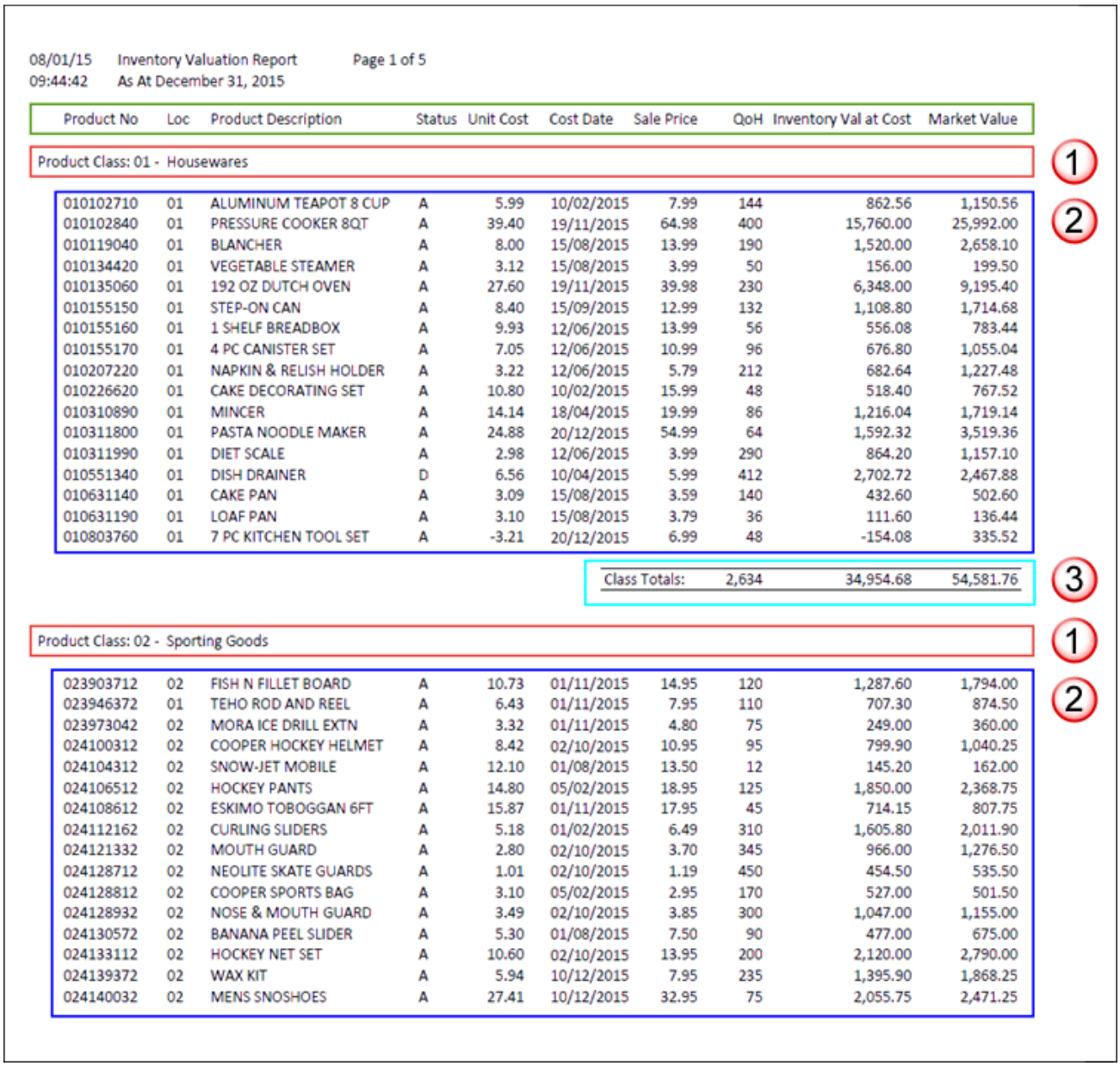
Additional guidelines

- You can define detail, header, or footer data in any order you want. A sequence is not enforced.
- You can also specify field names (outlined in green in "The different kinds of data in a PDF file" below). The method for specifying field names differs from the process for defining detail, header, or footer data.

> **Note**
> Do not use Header data to attempt to define field names that may appear in a print image or PDF file.

The different kinds of data in a PDF file

The example below highlights the different kinds of data in a PDF of an inventory valuation report.

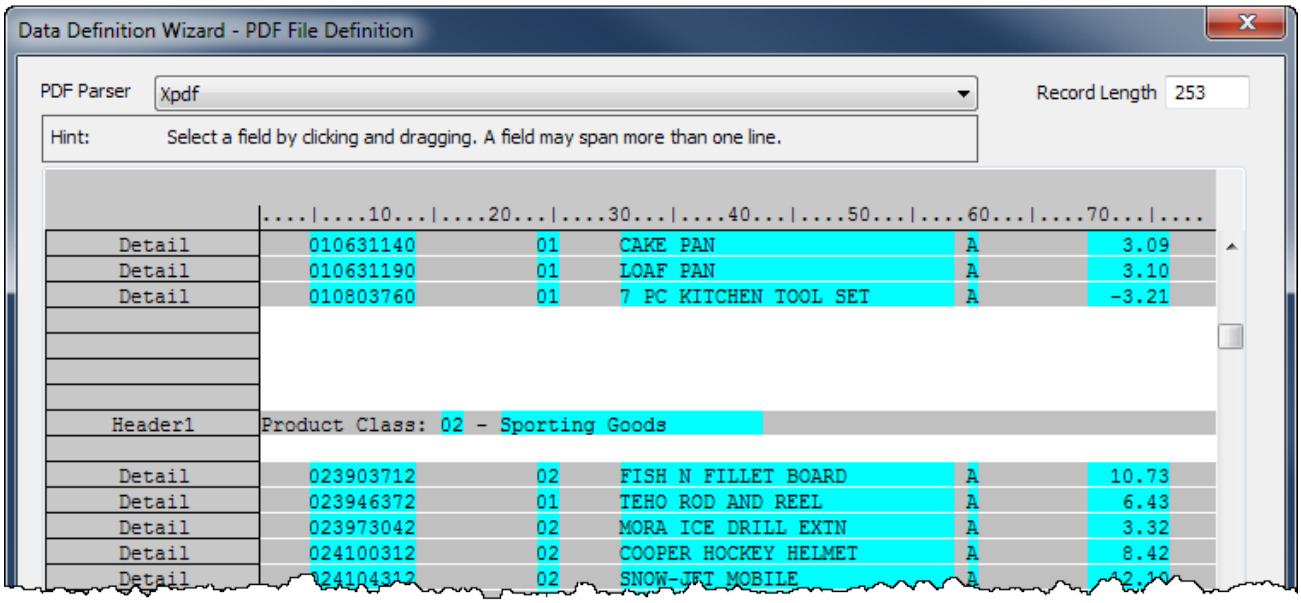


08/01/15 Inventory Valuation Report Page 1 of 5
09:44:42 As At December 31, 2015

Product No	Loc	Product Description	Status	Unit Cost	Cost Date	Sale Price	QoH	Inventory Val at Cost	Market Value
Product Class: 01 - Housewares									
010102710	01	ALUMINUM TEAPOT 8 CUP	A	5.99	10/02/2015	7.99	144	862.56	1,150.56
010102840	01	PRESSURE COOKER 8QT	A	39.40	19/11/2015	64.98	400	15,760.00	25,992.00
010119040	01	BLANCHER	A	8.00	15/08/2015	13.99	190	1,520.00	2,658.10
010134420	01	VEGETABLE STEAMER	A	3.12	15/08/2015	3.99	50	156.00	199.50
010135060	01	192 OZ DUTCH OVEN	A	27.60	19/11/2015	39.98	230	6,348.00	9,195.40
010155150	01	STEP-ON CAN	A	8.40	15/09/2015	12.99	132	1,108.80	1,714.68
010155160	01	1 SHELF BREADBOX	A	9.93	12/06/2015	13.99	56	556.08	783.44
010155170	01	4 PC CANISTER SET	A	7.05	12/06/2015	10.99	96	676.80	1,055.04
010207220	01	NAPKIN & RELISH HOLDER	A	3.22	12/06/2015	5.79	212	682.64	1,227.48
010226620	01	CAKE DECORATING SET	A	10.80	10/02/2015	15.99	48	518.40	767.52
010310890	01	MINCER	A	14.14	18/04/2015	19.99	86	1,216.04	1,719.14
010311800	01	PASTA NOODLE MAKER	A	24.88	20/12/2015	54.99	64	1,592.32	3,519.36
010311990	01	DIET SCALE	A	2.98	12/06/2015	3.99	290	864.20	1,157.10
010551340	01	DISH DRAINER	D	6.56	10/04/2015	5.99	412	2,702.72	2,467.88
010631140	01	CAKE PAN	A	3.09	15/08/2015	3.59	140	432.60	502.60
010631190	01	LOAF PAN	A	3.10	15/08/2015	3.79	36	111.60	136.44
010803760	01	7 PC KITCHEN TOOL SET	A	-3.21	20/12/2015	6.99	48	-154.08	335.52
						Class Totals:	2,634	34,954.68	54,581.76
Product Class: 02 - Sporting Goods									
023903712	02	FISH N FILLET BOARD	A	10.73	01/11/2015	14.95	120	1,287.60	1,794.00
023946372	01	TEHO ROD AND REEL	A	6.43	01/11/2015	7.95	110	707.30	874.50
023973042	02	MORA ICE DRILL EXTN	A	3.32	01/11/2015	4.80	75	249.00	360.00
024100312	02	COOPER HOCKEY HELMET	A	8.42	02/10/2015	10.95	95	799.90	1,040.25
024104312	02	SNOW-JET MOBILE	A	12.10	01/08/2015	13.50	12	145.20	162.00
024106512	02	HOCKEY PANTS	A	14.80	05/02/2015	18.95	125	1,850.00	2,368.75
024108612	02	ESKIMO TOBOGGAN 6FT	A	15.87	01/11/2015	17.95	45	714.15	807.75
024112162	02	CURLING SLIDERS	A	5.18	01/02/2015	6.49	310	1,605.80	2,011.90
024121332	02	MOUTH GUARD	A	2.80	02/10/2015	3.70	345	966.00	1,276.50
024128712	02	NEOLITE SKATE GUARDS	A	1.01	02/10/2015	1.19	450	454.50	535.50
024128812	02	COOPER SPORTS BAG	A	3.10	05/02/2015	2.95	170	527.00	501.50
024128932	02	NOSE & MOUTH GUARD	A	3.49	02/10/2015	3.85	300	1,047.00	1,155.00
024130572	02	BANANA PEEL SLIDER	A	5.30	01/08/2015	7.50	90	477.00	675.00
024133112	02	HOCKEY NET SET	A	10.60	02/10/2015	13.95	200	2,120.00	2,790.00
024139372	02	WAX KIT	A	5.94	10/12/2015	7.95	235	1,395.90	1,868.25
024140032	02	MENS SNOSHOES	A	27.41	10/12/2015	32.95	75	2,055.75	2,471.25

Detail data and header data in a parsed PDF file

The example below shows the inventory valuation report above once it has been parsed in the **Data Definition Wizard**. One detail record with five fields, and one header record with two fields, have been defined.



How header and footer data is treated

Although the **Data Definition Wizard** treats header or footer data like a record with fields, only detail data becomes an actual set of records in the resulting Analytics table. Any header or footer data you define becomes one or more fields added to the detail records.

The added header and footer fields repeat the same value for every record in an individual block or subset of records. For example, “Store 3” for one block of records, “Store 4” for the next block, and so on.

Do not select field names in the source file

Do not attempt to define field names by selecting them in the print image or PDF file. Although it may feel counter-intuitive, leave all field names in the source file unselected. Instead, you create field names by typing their names into the **Field Definition** dialog box. If you select field names in the source file, Analytics treats the field names as data contained in fields.

Specify a unique value to capture a set of records

The key to accurately capturing a set of records is selecting or specifying a value unique to the set of records. In other words, the value appears in a specific byte position (character position) in all the records in the set, and does not appear in that position anywhere else in the source file. The unique value can be one character or multiple characters.

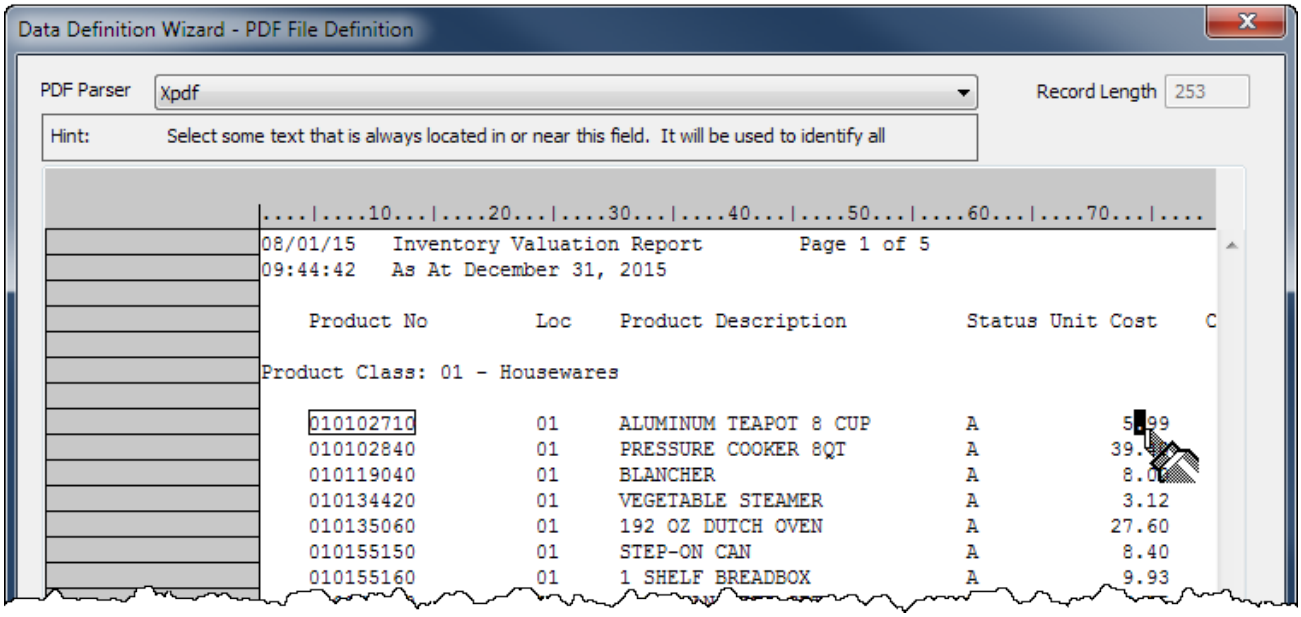
For example, in "Selecting a value unique to the set of records" on the facing page, the decimal point in the “Unit Cost” field is selected as the unique value. It appears in the same position in every amount in the field, and it does not appear in that position above or below the field.

You can select or specify the unique value in either of two places:

- In the initial data value you select to begin defining the initial data field
- In the same row as the initial data value

Selecting a value unique to the set of records

In the example below, the unique value is in the same row as the initial data value. The initial data value, surrounded by a box after being selected, is the first product number in the “Product No” field.



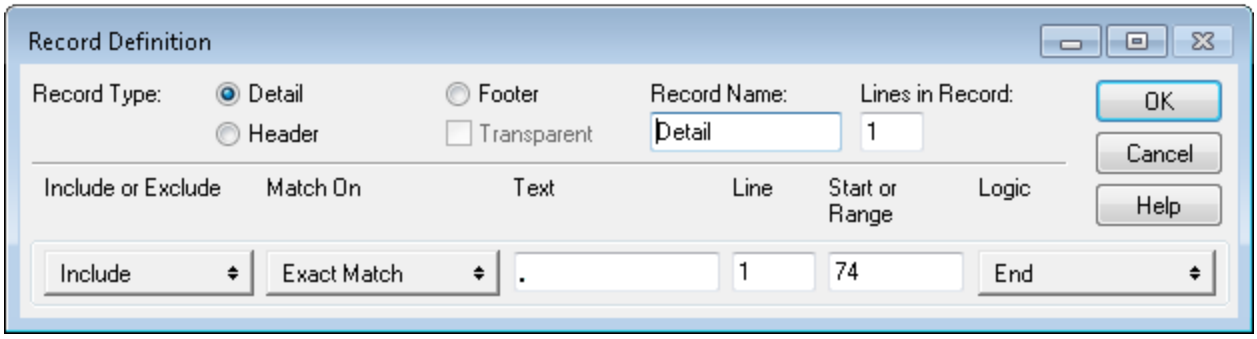
Tips for choosing a unique value

To choose a unique value, look for record data in which one or more consistently positioned characters are unique, or uniquely positioned, when compared to data above or below the set of records.

Any of the following possibilities could be good candidates for the unique value because they normally appear in the same position in every record, and they do not normally appear in that position outside the set of records:

- a decimal point in numbers
- one or both slashes in dates
- one or more hyphens in ID numbers
- a string of characters forming a standard prefix
- in header or footer data, a label that appears consistently, such as “Customer ID:” or “Subtotal:”

The initial selection of the unique value creates an **Exact Match** criterion in the **Record Definition** dialog box. In the example below, the criterion specifies that a decimal point must appear in byte position 74 in order for any record to be included in the set of records.

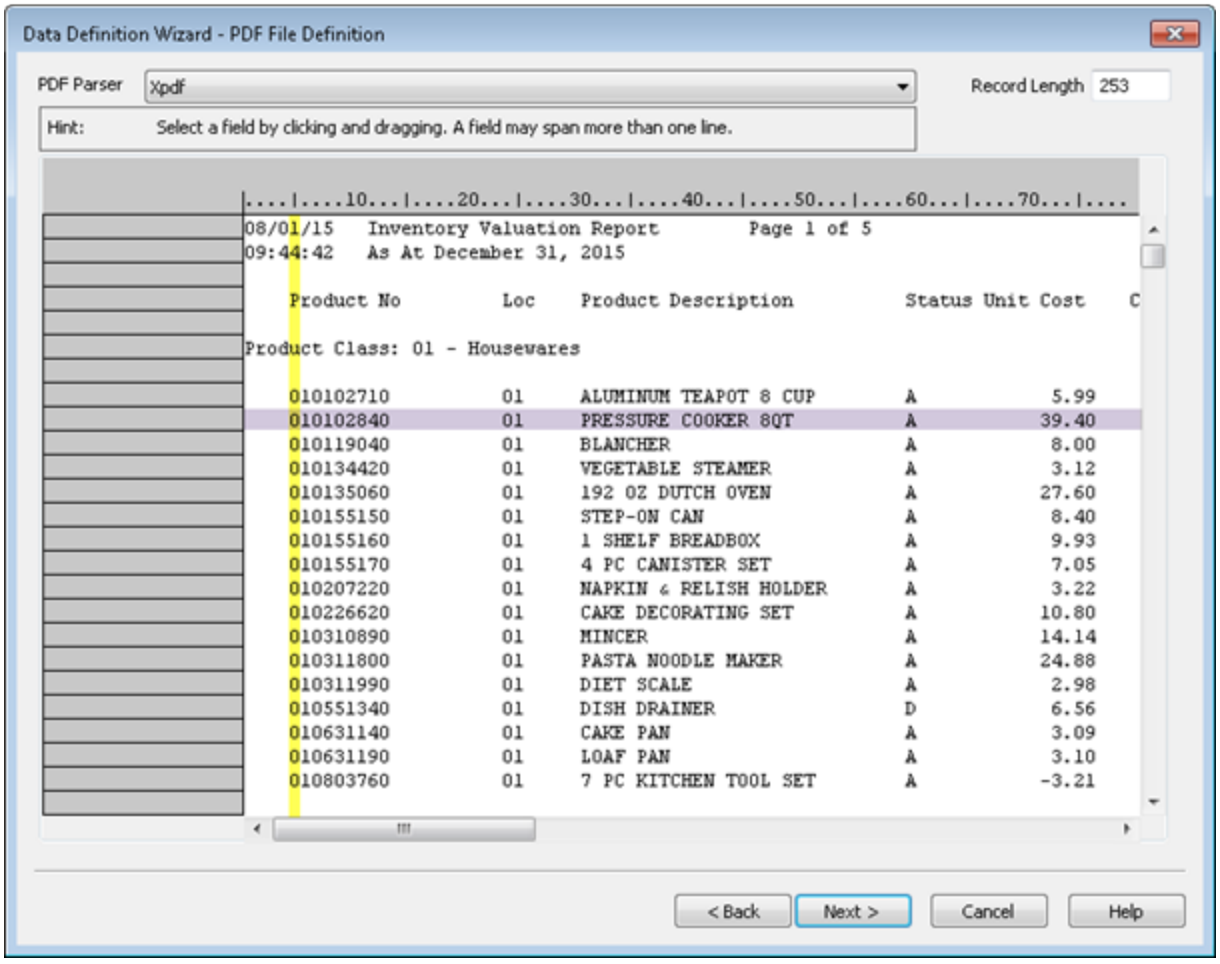


If required, you can change the exact match to a generic match, such as **Numeric**, or **Non-Blank**, which can provide greater flexibility when specifying a unique value. For more information, see "Working with record definitions" on page 304.

Precisely capture a set of records

Precisely capturing a set of records can be challenging. You may choose a value you think is unique to the set of records you want to capture and discover that some of the required records are not captured, or additional non-record data is captured.

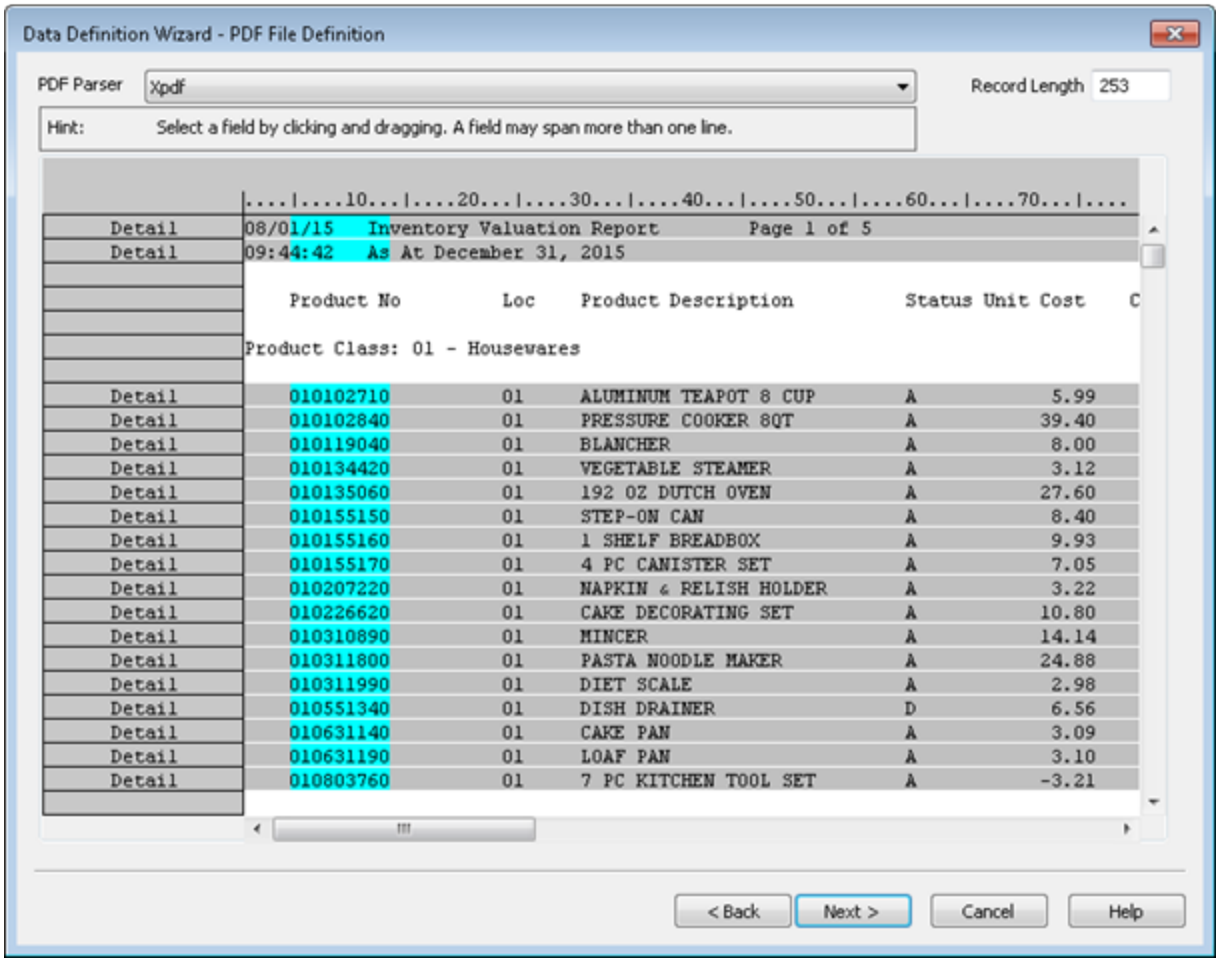
To understand this situation better, it may help to think of a print image or PDF file as a grid formed by columns and rows. Imagine each column is exactly one character or one space wide, and extends from the very top of the file to the very bottom of the file.



When you select or specify a value, in a specific position, to capture a set of records, Analytics considers any character or characters in that position, from the top of the file to the bottom, as it searches for the value. Characters are considered even if they are outside those rows that you consider record data. If the value you specified is not sufficiently precise, additional, non-record data can be captured and included in the set of records.

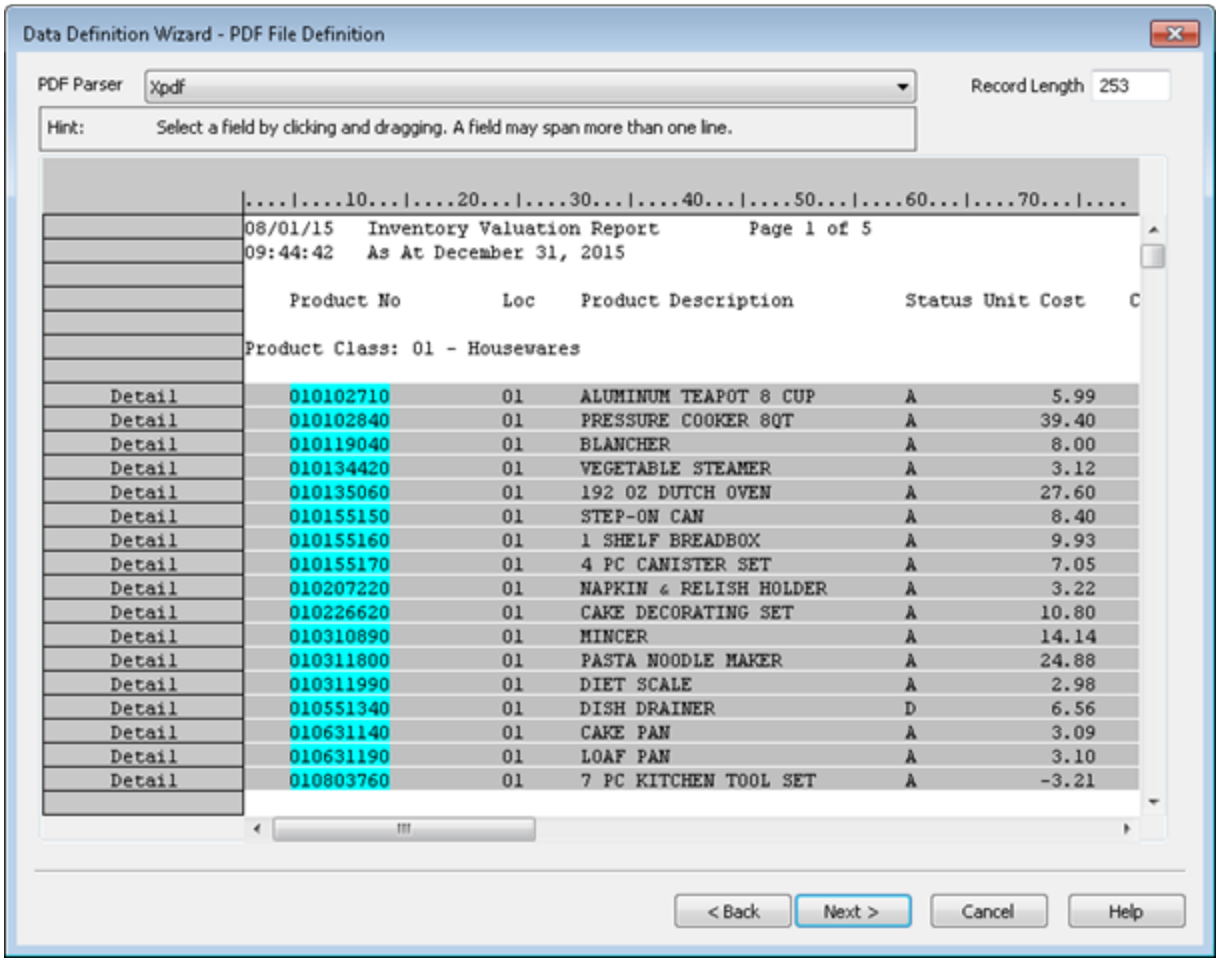
Imprecisely defined data field

In the example above, if you specified a generic numeric value in the first position of the “Product No” field as the unique value for capturing a set of records, any numbers in that position anywhere in the file would be captured in addition to the actual first digit of the product number. See the example below.



Precisely defined data field

If, however, you specified a generic numeric value encompassing all nine digits of the field, you would create a criterion that is sufficiently precise to capture only the intended set of records.



Use multiple criteria to capture a set of records

You may find that a single criterion, such as a slash in a date field, is not sufficient to precisely capture a set of records. Perhaps some of the date values are missing. Or, by chance, a slash appears in the same position in header or footer information, outside the set of records you want to capture. In this situation, you can add additional criteria to fine-tune the captured set of records.

> **Note**
> You can use criteria to either **include** or **exclude** rows in the source file.

Some examples of multiple criteria:

- Include rows that have the first slash in a date field AND the second slash in the date field
- Include rows that have the decimal point in a numeric field AND exclude rows that contain the word “Subtotal”
- Include rows that have alpha characters in the first specified set of byte positions OR include rows that have alpha characters in the second specified set of byte positions

For more information, see "Working with record definitions" on page 304.

Check record definitions and field definitions throughout the entire file

As you define records and fields, make sure to scroll through the file to check the accuracy of the definitions. Blank values, unexpected characters, and misaligned data can cause any of the following problems:

- some of the records in the file are excluded
- non-record data is incorrectly captured as a record
- field data is incompletely contained within a field definition, which truncates values
- data from two different fields appears in a single field definition

If a record definition is incorrect, you need to modify or further build the criteria used to capture the set of records. For more information, see "Working with record definitions" on page 304.

If a field definition is incorrect, you need to edit the definition. For more information, see "Working with field definitions" on page 299.

You can define multiline records and fields

You can define record or field data that extends beyond one line or one row in a source file. For example, the address data in each record could be arranged on multiple lines. For more information, see "Working with multiline records and fields" on page 314.

Define and import only data you need

Do not waste time or complicate the definition and import process by defining data fields you do not need for your analysis. Only define header or footer records if they add valuable information. Each additional data element you include can add complexity and make the definition process more difficult.

Control the order of fields in the resulting Analytics table

The order in which you define fields in a detail record is the order in which they appear in the resulting Analytics table. If you delete a detail field during the definition process, and then re-add it, it loses its original position and is placed last among detail fields. Detail fields remain together, regardless of any internal reshuffling.

> **Tip**
> If you use an initial detail field to capture detail records, but you do not want the field to appear first in the resulting Analytics table, you can use the field to capture records, and then delete it and re-add it.

Header and footer fields appear in the resulting Analytics table in the order in which you define them. They appear before detail fields if you have not defined an initial detail field, and they appear after detail fields once you have defined an initial detail field.

You also have the option of reordering fields once you have finished importing the print image or PDF file into Analytics. You can drag columns to reorder them in a view. You can also extract by view if you want to create a new table in which the fields in the table layout are physically reordered. For more information, see "Extracting data" on page 194. You may find reordering fields in Analytics is easier than trying to maintain a precise field order in the **Data Definition Wizard**.

Analytics may auto-define a file

Analytics may auto-define a print image or PDF file if it can identify recurring patterns in the data. If the initial appearance of the source file in the **Data Definition Wizard** includes aqua-blue field definitions and gray record definitions, then Analytics has either partially or fully auto-defined the file.

If you check the field and record definitions throughout the file and you judge the auto-definition to be complete and accurate, the work of defining the file is largely complete. You can proceed to the next page in the **Data Definition Wizard**.

Frequently, Analytics's auto-definition is not completely accurate and you need to decide which is easier: editing the auto-definition, or deleting the entire auto-definition and starting a manual definition from scratch. You can delete and start over at any point, so you may want to try some editing initially, and then if it becomes apparent that the auto-definition is too far away from what you require, delete it at that point.

> **Note**
> Only detail records are auto-defined. Header or footer data, if you require it, must be manually defined.

Use control totals to verify the resulting Analytics table

Before beginning any data analysis, make sure you use control totals to verify that the Analytics table created from a print image or PDF file contains all the data present in the source file. **An incomplete Analytics table will render any analysis you do invalid.**

To verify an Analytics table using control totals:

1. Do one of the following:
   - If the records are grouped in the source file, classify or summarize the Analytics table to group the records in the same manner.

     When you classify or summarize, select **Subtotal Fields** that match one or more subtotal fields in the source file.

     For more information, see "Classifying data" on page 1261, and "Summarizing data" on page 1267.
   - If the records are not grouped in the source file, total any fields in the Analytics table that are

also totaled in the source file.

For more information, see "Totaling fields" on page 838.

2. Output the results to screen, or to a new Analytics table, and compare the subtotals or totals in Analytics with the numbers in the source file.

   If all numbers are identical, you have a complete data set.

   If one or more numbers are not identical, the data in the Analytics table varies from the data in the source file. If you imported subsets of data, and reassembled a complete data set in Analytics, one possibility is that duplicate records exist in the Analytics table. For information about removing duplicate records, see "Remove duplicates" on page 1211.

   If duplicate records are not the problem, you may have to redo the definition and import of the source file. If you redo the definition, make sure to check the field and record definitions carefully, to ensure you are accurately capturing the data.

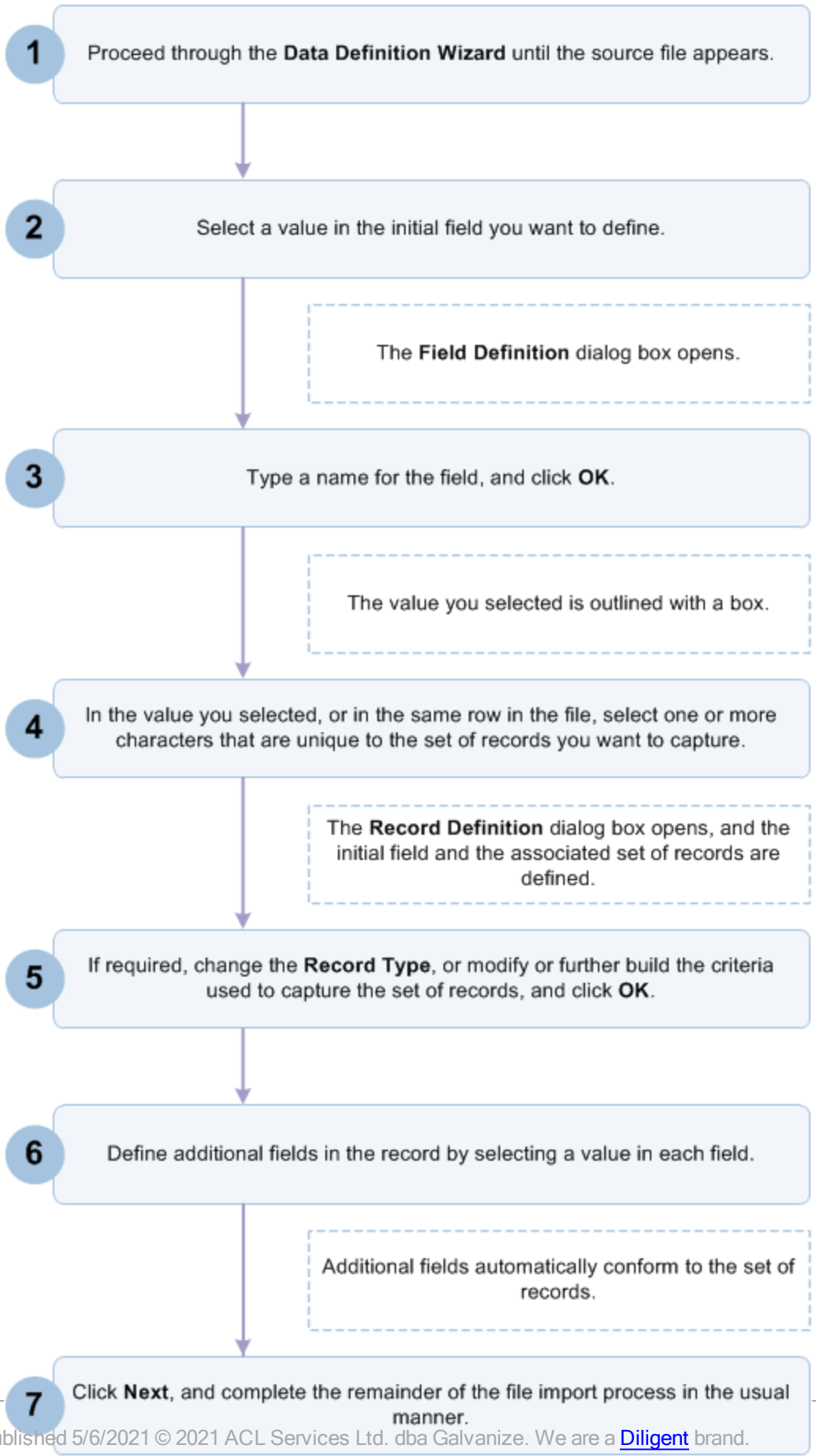
Quick Start: How to define a print image or PDF file

To start manually defining a print image or PDF file you select an initial data value, and then capture an associated set of records.

- "Workflow for defining a print image or PDF file" on the next page – the basic workflow for defining a print image or PDF file prior to importing it into Analytics
- "Quick start steps" on page 280 – step-by-step instructions with screen captures

The best way to learn how to define these types of files is to try defining one of the sample files included with Analytics. “REPORT3.TXT” is easier to define. “Inventory.pdf” is more challenging.

Workflow for defining a print image or PDF file


1
Proceed through the **Data Definition Wizard** until the source file appears.
2
Select a value in the initial field you want to define.
The **Field Definition** dialog box opens.
3
Type a name for the field, and click **OK**.
The value you selected is outlined with a box.
4
In the value you selected, or in the same row in the file, select one or more characters that are unique to the set of records you want to capture.
The **Record Definition** dialog box opens, and the initial field and the associated set of records are defined.
5
If required, change the **Record Type**, or modify or further build the criteria used to capture the set of records, and click **OK**.
6
Define additional fields in the record by selecting a value in each field.
Additional fields automatically conform to the set of records.
7
Click **Next**, and complete the remainder of the file import process in the usual manner.

Quick start steps

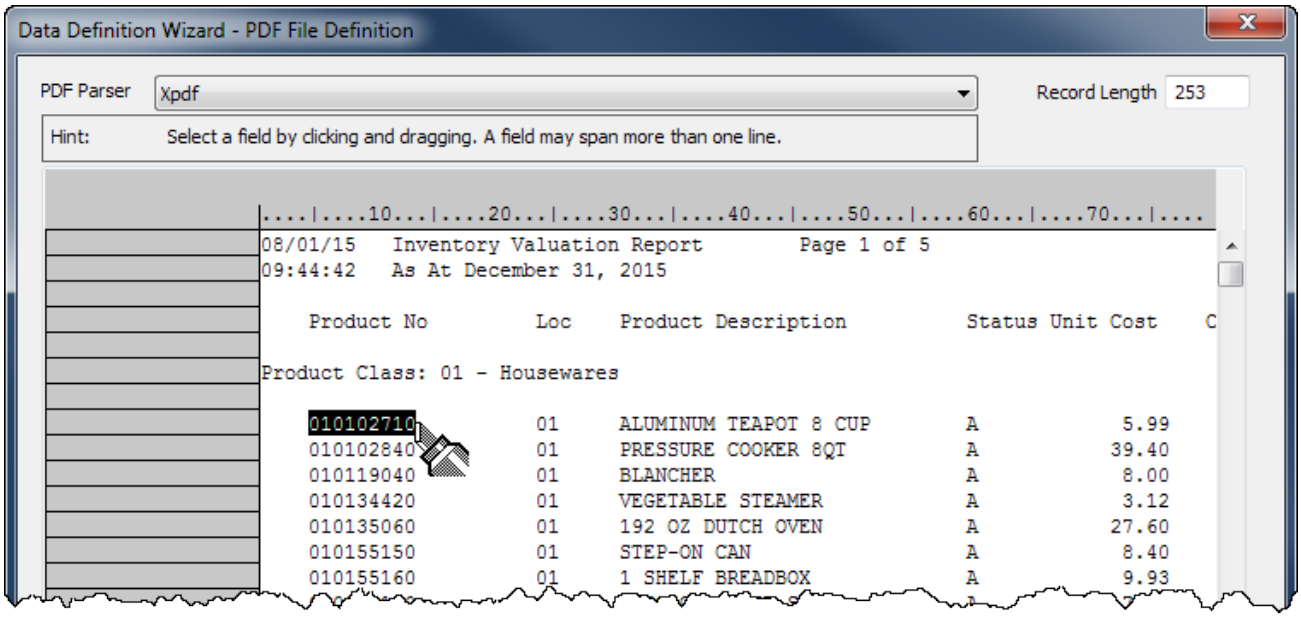
The procedure below outlines the basic steps for defining a print image or PDF file. For detailed instructions, see "Define and import a print image file" on page 283 or "Define and import a PDF file" on page 291.

1. Proceed through the **Data Definition Wizard** until the source file appears.

   For detailed instructions, see "Define and import a print image file" on page 283, or "Define and import a PDF file" on page 291.

2. Select a value in the initial field you want to define.

   In the example below, the first value in the “Product No” field is selected.



   The **Field Definition** dialog box opens.

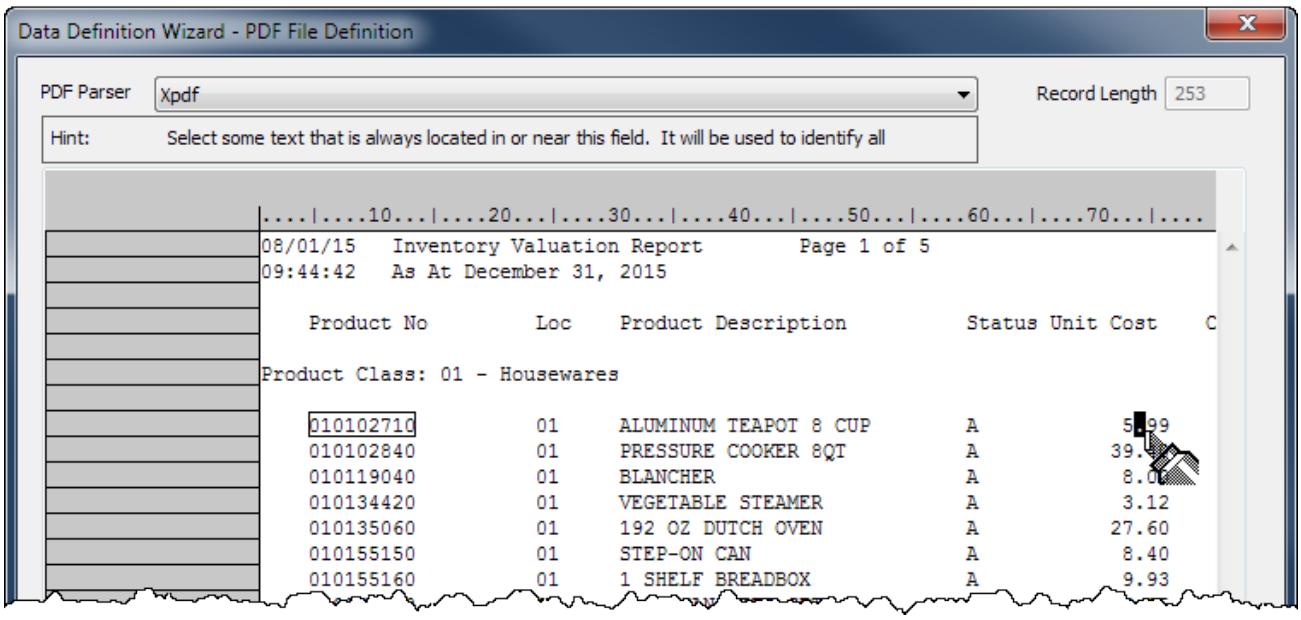
3. Type a name for the field, update the data type if required, and click **OK**.

   The value you selected is outlined with a box.

4. In the value you selected, or in the same row in the file, select one or more characters that are unique to the set of records you want to capture.

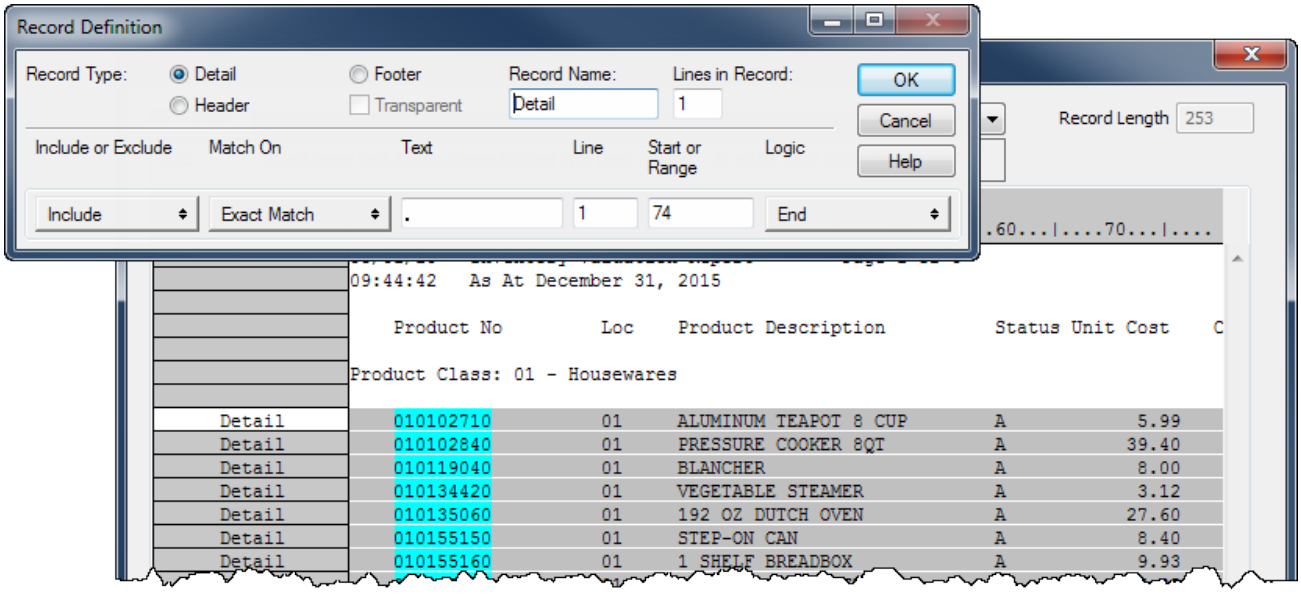
For more information, see "Defining and importing print image (report) files and PDF files" on page 261.

In the example below, the decimal point in the “Unit Cost” field is selected.



The **Record Definition** dialog box opens, and the initial field and the associated set of records are defined.

The field is aqua-blue, and the records are gray. Undefined data continues to have a white background.



5. If required, change the **Record Type**, or modify or further build the criteria used to capture the set of records, and click **OK**.

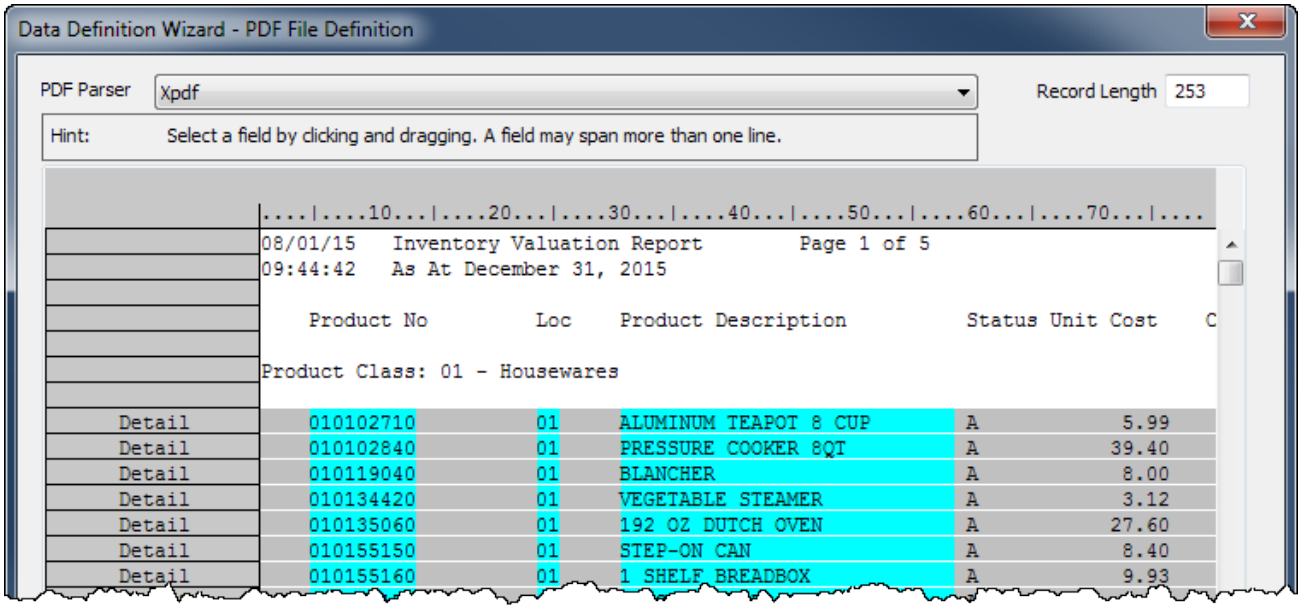
For detailed information, see "Working with record definitions" on page 304.

6. Define additional fields in the record by selecting a value in each field.

   Additional fields automatically conform to the set of records.

   If field values vary in length, select the longest value, or select extra blank spaces to allow for longer values elsewhere in the field.

   In the example below, three fields are defined: “Product No”, “Loc”, and “Product Description”.



7. When you have finished defining the fields you require, click **Next**.

   The remainder of the defining and importing process is similar to the process for defining and importing other data formats such as Excel and delimited text files.

   For complete instructions, see "Define and import a print image file" on the facing page, or "Define and import a PDF file" on page 291.

Define and import a print image file

You can create an Analytics table by defining and importing a print image file.

When you use the **Data Definition Wizard** to process a print image file, Analytics may fully or partially auto-define the file, or you may need to manually define the file.

> **Note**
> Defining print image files can be challenging. If you encounter problems, review "Defining and importing print image (report) files and PDF files" on page 261.

Locate and select the print image file

1. Select **File > New > Table**.
2. If the **Select Platform for Data Source** page is displayed, select **Local** and click **Next**.
3. In the **Select Local Data Source** page, select **File** and click **Next**.
4. In the **Select File to Define** dialog box, locate and select the print image file you want to create the Analytics table from and click **Open**.

   Print image files typically have a **.txt** file extension.
5. In the **Character Set** page, verify that the correct character set option has been selected and click **Next**.
6. In the **File Format** page, verify that **Print Image (Report) file** is selected and click **Next**.

   The print image file is parsed and the **Print Image File Definition** page displays the parsed file.

Define the print image file

1. In the **Print Image File Definition** page, scroll vertically and horizontally to examine the parsed file.

   Highlighting indicates whether Analytics has auto-defined data in the file:

Highlighting	Meaning
Aqua-blue highlighting	Data auto-defined as a field.
Gray highlighting	Data auto-defined as a record. Record definition depends on at least one field being defined in the record.
White background	Undefined data.

Highlighting	Meaning
	Analytics was not able to detect a pattern in the data and could not auto-define it.

2. Do one of the following:

Result of auto-definition	Action to take
If Analytics auto-defined the file and you do not want to make any updates	If Analytics auto-defined the entire file perfectly, and you do **not** want to: ◦ update the generic field names ◦ add any header or footer data to the detail data go to "Finalize the print image file definition" on page 288
If Analytics auto-defined the file and you want to make updates	If Analytics auto-defined the entire file perfectly, and you want to: ◦ update the generic field names ("Field_1", "Field_2", and so on), go to "Edit the auto-definition" below ◦ If you want to add header or footer data to the detail data, go to "Manually define the print image file" on the facing page **Tip** You can also update the generic field names in a subsequent page in the **Data Definition Wizard**, which you may find more convenient.
If the auto-definition contains errors	If the auto-definition: ◦ contains errors ◦ excludes data that you need ◦ includes data that you do not need you must do one of the following: ◦ "Edit the auto-definition" below ◦ delete the entire auto-definition and "Manually define the print image file" on the facing page **Tip** If the auto-definition contains significant errors, deleting the entire auto-definition and manually defining the file can be easier.
If the parsed file is entirely undefined	If the parsed file is entirely undefined, indicated by a completely white background, you must "Manually define the print image file" on the facing page

Edit the auto-definition

If you want to edit the auto-definition (or a manual definition), in the **Print Image File Definition** page, do any of the following:

Edit task	Instructions
Edit a field definition	Right-click an aqua-blue field and select **Edit Field**, or double-click the field. You can make a number of changes, including: ◦ updating the field name ◦ updating the data type ◦ under **Advanced Options**: • changing the field length (**Field Width**) • changing the starting position of the field For detailed information, see "Working with field definitions" on page 299.
Edit a record definition	Right-click a gray record and select **Edit Record**, or double-click the record. You can make two main changes: ◦ update the categorization of the record - detail, header, and footer are the options ◦ modify the criteria that Analytics used to capture the set of records For detailed information, see "Working with record definitions" on page 304.
Delete a field definition or a record definition	Right-click a field or a record and select **Delete Field** or **Delete Record**. You can delete definitions for fields that you do not want in the Analytics table, or that you want to define manually because of errors in their auto-definition. If you delete a record definition, any field definitions contained by the record are also deleted, and all instances of the record definition in the file are deleted. **Note** You are deleting the field definition or the record definition only, not the actual data. If necessary, you can redefine the same field or record data. **Tip** If you want to selectively delete records, select **Edit Record** and fine-tune the criteria that Analytics used to capture the set of records. For detailed information, see "Working with record definitions" on page 304.

Manually define the print image file

Tip

Before you begin, you may find it helpful to first review the basic version of the steps below, with accompanying screen captures: "Quick start steps" on page 280.

> **Note**
>
> You can also define a print image file using saved field and record definitions, if they exist.
>
> For more information, see "Define the print image file using a set of saved field and record definitions" on the facing page.

1. In the **Print Image File Definition** page, select a data value to start defining one of the fields in the table.

   For example, you could select a social security number in an SSN field. When you select the data value, the **Field Definition** dialog box opens.

   Guidelines:

   - You can select a value anywhere in the data. You do not have to use the first field in the table, or select the first value in a field.
   - The value you select can be detail data, header data, or footer data.
   - Do not select field names. Leave all field names in the source file unselected. If you select field names in the source file, Analytics treats them as data contained in fields.
   - If field values vary in length, select the longest value, or select extra blank spaces to allow for longer values that may be lower in the field and not currently displayed.

   If you intend to use the initial data value you selected to uniquely identify a set of records, see "Working with field definitions" on page 299.
2. Enter a name for the field, if necessary update the data type, and click **OK**.
3. In the data value you just selected, or in the same row in the file, select the character, or string of characters, that uniquely identifies the set of records in the source file.

   For example, select:

   - a slash in a date value
   - a decimal point in a numeric value
   - a unique identifying value anywhere in the row containing the data value you selected

   When you select the unique character or characters, the **Record Definition** dialog box opens, and all records containing the character or characters are highlighted gray.

   For detailed information, see "Defining and importing print image (report) files and PDF files" on page 261.

   If you need to define a record that extends beyond one row in the source file, see "Working with multiline records and fields" on page 314.
4. If required, update the **Record Type** to match the type of data you are defining: detail, header, or footer.
5. If required, modify the criteria used to capture the set of records.

   For example, you could add additional criteria to omit some of the records that were initially captured.

   For detailed information, see "Working with record definitions" on page 304.

6. Click **OK**.

   The field you defined is highlighted aqua-blue, and the associated set of captured records is highlighted gray.

7. Scroll vertically to examine the defined field, and the associated set of captured records.
8. If the field is not defined correctly, or if the set of captured records needs adjustment, double-click the field or the record, and make the necessary edits in the **Field Definition** dialog box, or the **Record Definition** dialog box.

   For more information, see "Working with field definitions" on page 299, or "Working with record definitions" on page 304.

9. Define the remaining fields in the record by selecting a representative data value for each field.

   Additional fields automatically conform to the set of records.

   Guidelines:

   - Define only those fields you want in the resulting Analytics table.
   - With each field definition, scroll vertically to examine the defined field. Edit the definitions as required.

     For example, if data values are not fully contained by a field, you need to adjust the length or starting position of the field, or both.

     For more information, see "Edit the auto-definition" on page 284.

   - If you need to define field values that extend beyond one row in the source file, see "Working with multiline records and fields" on page 314.

     > **Tip**
     >
     > The order in which you define detail fields is the order in which they appear in the resulting Analytics table.
     >
     > If you delete a detail field during the definition process, and then re-add it, it loses its original position and is placed last among detail fields.

10. If you want to define another record, repeat steps 1 to 9.

    Guidelines:

    - When you select a data value to start defining a new field and associated set of records, ensure **New Record** is selected in the dialog box that appears, and click **OK**.
    - You can define multiple header or footer records, but only one detail record. The order in which you define the different record types is not enforced.

Define the print image file using a set of saved field and record definitions

You can define a print image file using field and record definitions from a previous file definition session that have been saved in a **print image query file**. The print image query file must already exist, and the saved definitions must match the current data.

> **Note**
> Loading a print image query file deletes any current field and record definitions.

1. In the **Print Image File Definition** page, click **Load**.
2. Navigate to a previously saved print image query file, select it, and click **Open**.

   The definitions are applied to the current data.

   Print image query files have a .txt extension.

   > **Note**
   > Only load a file with definitions that you know match, or closely match, the current data.

3. After loading the file, do one of the following:
   - **If the entire file is now perfectly defined** – go to "Finalize the print image file definition" below
   - **If the file definition needs adjustment** – go to "Edit the auto-definition" on page 284

Finalize the print image file definition

1. Optional. If you want to save the current set of field and record definitions to a print image query file, do the following:
   a. Click **Save**.
   b. Enter a name for the print image query file and click **Save**.

   > **Note**
   > Field and record definitions often represent a lot of work, and it is recommended that you save them.
   >
   > If you subsequently discover that the imported data needs an adjustment, and must be redefined and reimported, saved definitions do not have to be recreated from scratch.

2. When you are satisfied with all field and record definitions, click **Next**.

   > **Note**
   > If required, you can return to this point in the process and make updates to the field and record definitions.

Save the Analytics data file

In the **Save Data File As** dialog box, enter a name for the Analytics data file and click **Save**.

If Analytics prefills a data file name, you can accept the prefilled name, or change it.

You can also navigate to a different folder to save the data file if you do not want to use the default location opened by Analytics.

Edit the Analytics field properties

In the **Edit Field Properties** page, review the settings assigned by Analytics to the properties listed below, make any required updates, and click **Next**.

> **Note**
> Select a column heading in the preview table to see the properties associated with the column.

Property	Description
Ignore this field	Excludes the field from the resulting table layout. The data in the field is still imported, but it is undefined, and does not appear in the new Analytics table. It can be defined later, if necessary, and added to the table.
Name	The name for the field in the table layout. You can keep the name assigned by Analytics, or enter a different name.
Column Title	The column title for the field in the default Analytics view. If you do not specify a column title, the **Name** value is used.
Type	The data type assigned to the field in Analytics. You can keep the data type assigned by Analytics, or select an appropriate data type from the drop-down list. For information about the supported data types in Analytics, see "Data types in Analytics" on page 739.
Value	A read-only property that displays the first value in the field. The value dynamically updates based on any edits you make.
Decimal	Numeric fields only. The number of decimal places in the source data. > **Note** > The **Decimal** text box appears automatically when you select a Numeric data type.
Input Format	Datetime fields only. The format of datetime values in the source data. The format you specify must exactly match the format in the source data. For more information about date and time formats, see "Formats of date and time source data" on page 347.

Finalize the import

1. In the **Final** page, verify the settings for the new Analytics table and click **Finish**.

   If you want to make any changes, click **Back** to get to the appropriate page in the wizard.
2. Enter a name for the table layout that you are adding to the project, or keep the default name, and click **OK**.

   The new Analytics table is created with data from the imported file.

Define and import a PDF file

You can create an Analytics table by defining and importing an Adobe PDF file.

When you use the **Data Definition Wizard** to process a PDF file, Analytics may fully or partially auto-define the file, or you may need to manually define the file.

> **Note**
> Defining PDF files can be challenging. If you encounter problems, review "Defining and importing print image (report) files and PDF files" on page 261.

Locate and select the PDF file

1. Select **File > New > Table**.
2. If the **Select Platform for Data Source** page is displayed, select **Local** and click **Next**.
3. In the **Select Local Data Source** page, select **File** and click **Next**.
4. In the **Select File to Define** dialog box, locate and select the PDF file you want to create the Analytics table from and click **Open**.

   Adobe PDF files have a **.pdf** file extension.
5. In the **File Format** page, verify that **PDF Adobe Acrobat file** is selected and click **Next**.

Define the PDF file

1. In the **PDF File Definition** page, if required, enter the password for the PDF file and click **Next**.
2. If you want to specify a particular page or page range for parsing, rather than **All** pages, select **Pages**, and specify one or more page numbers.

   You can specify individual pages separated by commas (1,3,5), page ranges (2-7), or a combination (1, 3, 5-7, 11).

   > **Tip**
   > In some circumstances, parsing a PDF file on a page-by-page basis can help with data misalignment.
   >
   > If you take this approach, you need to import the file more than once, create more than one Analytics table, and then append the resulting tables in Analytics.
   >
   > For more information, see "Defining and importing subsets of print image or PDF data" on page 312.

3. Leave the **PDF Parser** at the default setting of **Xpdf**, or select **VeryPDF**.

   If you are importing the file for the first time, and you have no reason to do otherwise, leave the setting at **Xpdf**.

   If you have already encountered data alignment issues when using **Xpdf** with the file, select **VeryPDF** to see if the parsing results are better.

4. Click **Next**.

   The PDF file is parsed and the **PDF File Definition** page updates to display the parsed file.

5. Scroll vertically and horizontally to examine the parsed file.

   Highlighting indicates whether Analytics has auto-defined data in the file:

Highlighting	Meaning
Aqua-blue highlighting	Data auto-defined as a field.
Gray highlighting	Data auto-defined as a record. Record definition depends on at least one field being defined in the record.
White background	Undefined data. Analytics was not able to detect a pattern in the data and could not auto-define it.

6. Optional. If the data in the parsed file is misaligned, click **Back**, switch the parser selection in **PDF Parser**, and click **Next**.

   The PDF file is re-parsed using the parser you selected, which may produce better data alignment.

   Any existing field and record definitions are deleted when you re-parse the file.

7. Do one of the following:

Result of auto-definition	Action to take
If Analytics auto-defined the file and you do not want to make any updates	If Analytics auto-defined the entire file perfectly, and you do **not** want to: ◦ update the generic field names ◦ add any header or footer data to the detail data go to "Finalize the PDF file definition" on page 297
If Analytics auto-defined the file and you want to make updates	If Analytics auto-defined the entire file perfectly, and you want to: ◦ update the generic field names (“Field_1”, “Field_2”, and so on), go to "Edit the auto-definition" on the facing page ◦ If you want to add header or footer data to the detail data, go to "Manually define the PDF file" on page 294

Result of auto-definition	Action to take
	Tip You can also update the generic field names in a subsequent page in the **Data Definition Wizard**, which you may find more convenient.
If the auto-definition contains errors	If the auto-definition: ◦ contains errors ◦ excludes data that you need ◦ includes data that you do not need you must do one of the following: ◦ "Edit the auto-definition" below ◦ delete the entire auto-definition and "Manually define the PDF file" on the next page **Tip** If the auto-definition contains significant errors, deleting the entire auto-definition and manually defining the file can be easier.
If the parsed file is entirely undefined	If the parsed file is entirely undefined, indicated by a completely white background, you must "Manually define the PDF file" on the next page

Edit the auto-definition

If you want to edit the auto-definition (or a manual definition), in the **PDF File Definition** page, do any of the following:

Edit task	Instructions
Edit a field definition	Right-click an aqua-blue field and select **Edit Field**, or double-click the field. You can make a number of changes, including: ◦ updating the field name ◦ updating the data type ◦ under **Advanced Options**: • changing the field length (**Field Width**) • changing the starting position of the field For detailed information, see "Working with field definitions" on page 299.
Edit a record definition	Right-click a gray record and select **Edit Record**, or double-click the record. You can make two main changes: ◦ update the categorization of the record - detail, header, and footer are the options ◦ modify the criteria that Analytics used to capture the set of records For detailed information, see "Working with record definitions" on page 304.

Edit task	Instructions
Delete a field definition or a record definition	Right-click a field or a record and select **Delete Field** or **Delete Record**. You can delete definitions for fields that you do not want in the Analytics table, or that you want to define manually because of errors in their auto-definition. If you delete a record definition, any field definitions contained by the record are also deleted, and all instances of the record definition in the file are deleted. **Note** You are deleting the field definition or the record definition only, not the actual data. If necessary, you can redefine the same field or record data. **Tip** If you want to selectively delete records, select **Edit Record** and fine-tune the criteria that Analytics used to capture the set of records. For detailed information, see "Working with record definitions" on page 304.

Manually define the PDF file

Tip

Before you begin, you may find it helpful to first review the basic version of the steps below, with accompanying screen captures: "Quick start steps" on page 280.

Note

You can also define a PDF file using saved field and record definitions, if they exist.

For more information, see "Define the PDF file using a set of saved field and record definitions" on page 296.

1. In the **PDF File Definition** page, select a data value to start defining one of the fields in the table.

   For example, you could select a social security number in an SSN field. When you select the data value, the **Field Definition** dialog box opens.

   Guidelines:

   - You can select a value anywhere in the data. You do not have to use the first field in the table, or select the first value in a field.
   - The value you select can be detail data, header data, or footer data.
   - Do not select field names. Leave all field names in the source file unselected. If you select field names in the source file, Analytics treats them as data contained in fields.
   - If field values vary in length, select the longest value, or select extra blank spaces to allow for longer values that may be lower in the field and not currently displayed.

If you intend to use the initial data value you selected to uniquely identify a set of records, see "Working with field definitions" on page 299.

2. Enter a name for the field, if necessary update the data type, and click **OK**.
3. In the data value you just selected, or in the same row in the file, select the character, or string of characters, that uniquely identifies the set of records in the source file.

   For example, select:

   - a slash in a date value
   - a decimal point in a numeric value
   - a unique identifying value anywhere in the row containing the data value you selected

   When you select the unique character or characters, the **Record Definition** dialog box opens, and all records containing the character or characters are highlighted gray.

   For detailed information, see "Defining and importing print image (report) files and PDF files" on page 261.

   If you need to define a record that extends beyond one row in the source file, see "Working with multiline records and fields" on page 314.
4. If required, update the **Record Type** to match the type of data you are defining: detail, header, or footer.
5. If required, modify the criteria used to capture the set of records.

   For example, you could add additional criteria to omit some of the records that were initially captured.

   For detailed information, see "Working with record definitions" on page 304.
6. Click **OK**.

   The field you defined is highlighted aqua-blue, and the associated set of captured records is highlighted gray.
7. Scroll vertically to examine the defined field, and the associated set of captured records.
8. If the field is not defined correctly, or if the set of captured records needs adjustment, double-click the field or the record, and make the necessary edits in the **Field Definition** dialog box, or the **Record Definition** dialog box.

   For more information, see "Working with field definitions" on page 299, or "Working with record definitions" on page 304.
9. Define the remaining fields in the record by selecting a representative data value for each field.

   Additional fields automatically conform to the set of records.

   Guidelines:

   - Define only those fields you want in the resulting Analytics table.
   - With each field definition, scroll vertically to examine the defined field. Edit the definitions as required.

     For example, if data values are not fully contained by a field, you need to adjust the length or starting position of the field, or both.

     For more information, see "Edit the auto-definition" on page 293.

- If you need to define field values that extend beyond one row in the source file, see "Working with multiline records and fields" on page 314.

> **Tip**
>
> The order in which you define detail fields is the order in which they appear in the resulting Analytics table.
>
> If you delete a detail field during the definition process, and then re-add it, it loses its original position and is placed last among detail fields.

10. If you want to define another record, repeat steps 1 to 9.

    Guidelines:

    - When you select a data value to start defining a new field and associated set of records, ensure **New Record** is selected in the dialog box that appears, and click **OK**.
    - You can define multiple header or footer records, but only one detail record. The order in which you define the different record types is not enforced.

Define the PDF file using a set of saved field and record definitions

You can define a PDF file using field and record definitions from a previous file definition session that have been saved in a **print image query file**. The print image query file must already exist, and the saved definitions must match the current data.

> **Note**
>
> Loading a print image query file deletes any current field and record definitions.

1. In the **PDF File Definition** page, click **Load**.
2. Navigate to a previously saved print image query file, select it, and click **Open**.

   The definitions are applied to the current data.

   Print image query files have a .txt extension.

> **Note**
>
> Only load a file with definitions that you know match, or closely match, the current data.

3. After loading the file, do one of the following:
   - **If the entire file is now perfectly defined** – go to "Finalize the PDF file definition" on the facing page
   - **If the file definition needs adjustment** – go to "Edit the auto-definition" on page 293

Finalize the PDF file definition

1. Optional. If you want to save the current set of field and record definitions to a print image query file, do the following:
   a. Click **Save**.
   b. Enter a name for the print image query file and click **Save**.

> **Note**
> Field and record definitions often represent a lot of work, and it is recommended that you save them.
>
> If you subsequently discover that the imported data needs an adjustment, and must be redefined and reimported, saved definitions do not have to be recreated from scratch.

2. When you are satisfied with all field and record definitions, click **Next**.

> **Note**
> If required, you can return to this point in the process and make updates to the field and record definitions.

Save the Analytics data file

In the **Save Data File As** dialog box, enter a name for the Analytics data file and click **Save**.

If Analytics prefills a data file name, you can accept the prefilled name, or change it.

You can also navigate to a different folder to save the data file if you do not want to use the default location opened by Analytics.

Edit the Analytics field properties

In the **Edit Field Properties** page, review the settings assigned by Analytics to the properties listed below, make any required updates, and click **Next**.

> **Note**
> Select a column heading in the preview table to see the properties associated with the column.

Property	Description
Ignore this field	Excludes the field from the resulting table layout. The data in the field is still imported, but it is undefined, and does not appear in the new Analytics table. It can be defined later, if necessary, and added to the table.

Name	The name for the field in the table layout. You can keep the name assigned by Analytics, or enter a different name.
Column Title	The column title for the field in the default Analytics view. If you do not specify a column title, the **Name** value is used.
Type	The data type assigned to the field in Analytics. You can keep the data type assigned by Analytics, or select an appropriate data type from the drop-down list. For information about the supported data types in Analytics, see "Data types in Analytics" on page 739.
Value	A read-only property that displays the first value in the field. The value dynamically updates based on any edits you make.
Decimal	Numeric fields only. The number of decimal places in the source data. **Note** The **Decimal** text box appears automatically when you select a Numeric data type.
Input Format	Datetime fields only. The format of datetime values in the source data. The format you specify must exactly match the format in the source data. For more information about date and time formats, see "Formats of date and time source data" on page 347.

Finalize the import

1. In the **Final** page, verify the settings for the new Analytics table and click **Finish**.

   If you want to make any changes, click **Back** to get to the appropriate page in the wizard.
2. Enter a name for the table layout that you are adding to the project, or keep the default name, and click **OK**.

   The new Analytics table is created with data from the imported file.

Working with field definitions

A field definition is information that delineates a single field in a print image or PDF file. Because a print image or PDF file is an image, without any metadata identifying fields and records, you need to specify one or more field definitions to identify the fields in the file, and differentiate them from surrounding data or white space.

One or more field definitions may be automatically created by Analytics during the file definition process, or you may have to manually create field definitions.

Using the initial data value to uniquely identify a set of records

To start manually defining a print image or PDF file, you select an initial data value, and then capture an associated set of records. If you decide to use part or all of the initial data value to uniquely identify the set of records, follow the guidelines below when choosing the field containing the initial data value.

- The field can be positioned anywhere in the record. It does not have to be the first field in the record.
- Look for a field in which the data has a consistent structure. For example:
  - a date field with a consistent format such as MM/DD/YYYY
  - an SSN field
  - a credit card number field
  - any ID or numeric field with a consistent structure

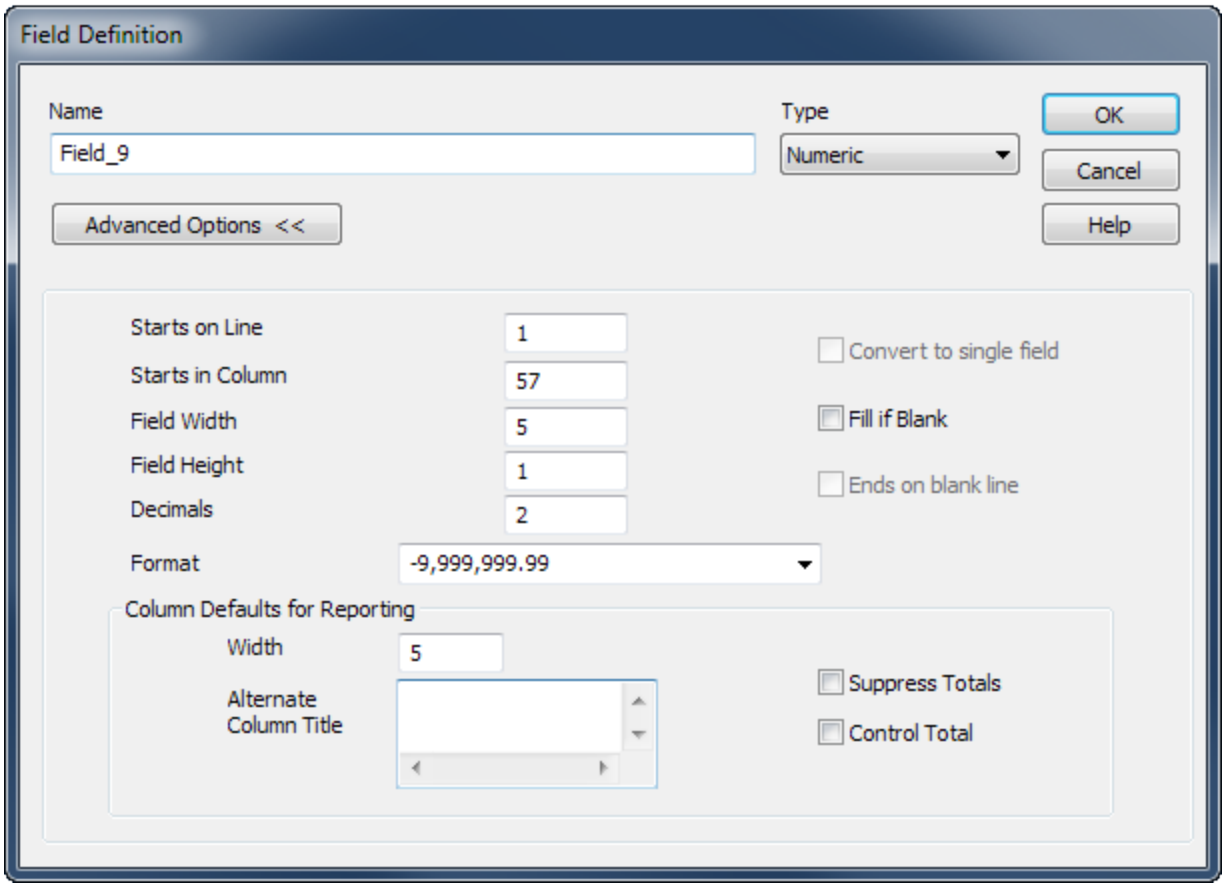
  You will have greater success using a consistently structured field than you will using a field with varying contents.
- One or more consistently positioned characters in the field must be unique, or uniquely positioned, when compared to data above or below the field.
- Avoid a field with missing values. It is possible to use a field with missing values, but it complicates the process of defining the file.

> **Note**
>
> The value you use to uniquely identify a set of records does not have to be contained in the initial data value or the initial data field. It can occur anywhere in the row containing the initial data value. For more information, see "Defining and importing print image (report) files and PDF files" on page 261.

The Field Definition dialog box

The **Field Definition** dialog box is where you specify the information that delineates a field in a print image or PDF file.

The figure below shows the **Field Definition** dialog box with the **Advanced Options** expanded.



The table below explains the purpose of each item in the **Field Definition** dialog box:

Item name	Purpose
Name	Specifies a field name other than the generic field name assigned by Analytics. The name you specify becomes the physical field name in the resulting Analytics table - that is, the field name in the table layout.
Type	Specifies the data type of the field. The options are **Character**, **Numeric**, and **Datetime**. If the values in a numeric or datetime field are inconsistent, you can try defining and importing the fields as character data.
Starts on Line	Specifies which line in a record contains the start of the field. For example: ◦ If each record containing the field appears on a single line, then the value must be '1' ◦ If each record containing the field spans two lines, and the field starts on the second line, then the value must be '2'
Starts in Column	Specifies the starting byte position of the field. For example, if three blank spaces at the beginning of a line precede the first character in a field, then the **Starts in Column** value must be '4' (non-Unicode Analytics), or '7'

Item name	Purpose
	(Unicode data in Unicode Analytics). **Note** The starting position of a field is critical to the success of the defining and importing process. Once a field is defined, scroll through the source file to ensure the starting position accommodates all values in the field. Adjust the starting position if necessary. For Unicode data, typically you should specify an odd-numbered starting byte position. Specifying an even-numbered starting position can cause characters to display incorrectly.
Field Width	Specifies the length in bytes of the field. The length you specify becomes the physical field length in the resulting Analytics table - that is, the field length in the table layout. **Note** Field length is critical to the success of the defining and importing process. Once a field is defined, scroll through the source file to ensure that the field is long enough to accommodate all values in the field. Adjust the length if necessary. For Unicode data, specify an even number of bytes only. Specifying an odd number of bytes can cause characters to display incorrectly.
Field Height	Specifies the number of lines that constitute a single value in the field. For example: ◦ If each value appears on a single line, then the field height must be '1' ◦ If each value spans two lines, then the field height must be '2' ◦ If each value spans a varying number of lines, such as the content of a Note field, set the field height to accommodate the value that spans the greatest number of lines (see **Ends on blank line** below)
Decimals (numeric fields only)	Specifies the number of decimal places in numeric values.
Format (numeric and datetime fields only)	Specifies the format for numeric or datetime data. The format needs to match the format of the numeric or datetime values in the source file. For example: ◦ If numbers such as -1,234.00 appear in the field, you need to select or specify the format **-9,999,999.99**. ◦ If dates such as 31/12/2015 appear in the field, you need to select or specify the format **DD/MM/YYYY**. Use MMM in the format to match months that use abbreviations or that are spelled out.

Item name	Purpose
	Tip If numeric or datetime data in the source file is inconsistently formatted, you can import it as character data and try to clean up the inconsistencies using Analytics functions in the resulting Analytics table.
Convert to single field (character fields only) (multiline fields only)	Specifies that multiline fields defined in the source file are imported to Analytics as a single field containing the data from all lines. For example, if you define address data that spans several lines, selecting **Convert to single field** creates a single field with all the address data on one line. If you leave **Convert to single field** unselected (the default setting), multiline fields are imported to Analytics as multiple fields each containing the data from a single line.
Fill if Blank	Specifies that a field value is copied to subsequent blank values until a new field value occurs. For example, if the value "01" in the Product Class field appears in only the first record of a block of Product Class 01 records, selecting **Fill if Blank** causes the value "01" to appear with every record.
Ends on blank line (multiline fields only)	Specifies that values in a multiline field terminate when they encounter a blank line. This option addresses a situation that occurs when values in a multiline field span a varying number of lines. You must set the **Field Height** to accommodate the value that spans the greatest number of lines. However, doing so can cause a mismatch between values with fewer lines and field or record boundaries. Selecting **Ends on blank line** causes the field height, and the field or record boundaries, to dynamically resize to fit the number of lines occupied by each value. **Note** This feature only works if one or more blank lines separate each value in the multiline field.
Column Defaults for Reporting: ◦ **Width** ◦ **Alternate Column Title** ◦ **Suppress Totals** (numeric fields only) ◦ **Control Total** (numeric fields only)	**Note** The **Column Defaults for Reporting** settings are optional. They do not affect processing of the field in the **Data Definition Wizard**. The same properties can be set later, in Analytics. Specifies properties of the field as it appears in the default view in the resulting Analytics table, and in Analytics reports. ◦ **Width** - Specifies the display width of the field in bytes. This value is used as the column size when displaying the contents of the field in Analytics views and reports. ◦ **Alternate Column Title** - Specifies a column heading to use, instead of the field name, when displaying the field in Analytics views and reports. ◦ **Suppress Totals** - Specifies that values in the field are not automatically totaled in Analytics reports. By default, Analytics automatically totals numeric fields in reports. You can suppress this behavior if the field contains data, such as unit prices, for which totals are not meaningful.

Item name	Purpose
	◦ **Control Total** - Specifies that the field is a control total field. A control total is the sum of values in a numeric field, which can be used to check data integrity. When you extract or sort data to a new table, Analytics includes the input and output totals of a control total field in the table history. Input refers to the original table. Output refers to the new table. If the two totals match, no data was lost in the extract or sort operation. If you specify control totals for more than one field, the table history reports on only the numeric field with the leftmost starting position. **Note** The **Control Total** setting in the **Field Definition** dialog box does not create control totals when you import a print image or PDF file to Analytics. For information about creating control totals for this purpose, see "Defining and importing print image (report) files and PDF files" on page 261.

Working with record definitions

A record definition is information that captures, or delineates, a set of records in a print image or PDF file. Because a print image or PDF file is an image, without any metadata identifying fields and records, you need to specify one or more record definitions to identify the records in the file, and differentiate them from surrounding data or white space.

Defining a set of detail records is required when defining a print image or PDF file. You may also want to define header or footer records, but they are not a requirement.

A detail record definition may be automatically created by Analytics during the file definition process, or you may have to manually create one or more record definitions.

The starting point is selecting an initial data value

The starting point for a record definition is selecting an initial data value in an initial data field. A character or characters in the initial data value, or in the row containing the initial data value, form the basis for the record definition, which identifies the set of records associated with the initial data field. For more information, see "Defining and importing print image (report) files and PDF files" on page 261.

Once the initial data field and the set of associated records are defined, you specify as many additional field definitions as required to break up the remaining portion of the record into its separate data elements.

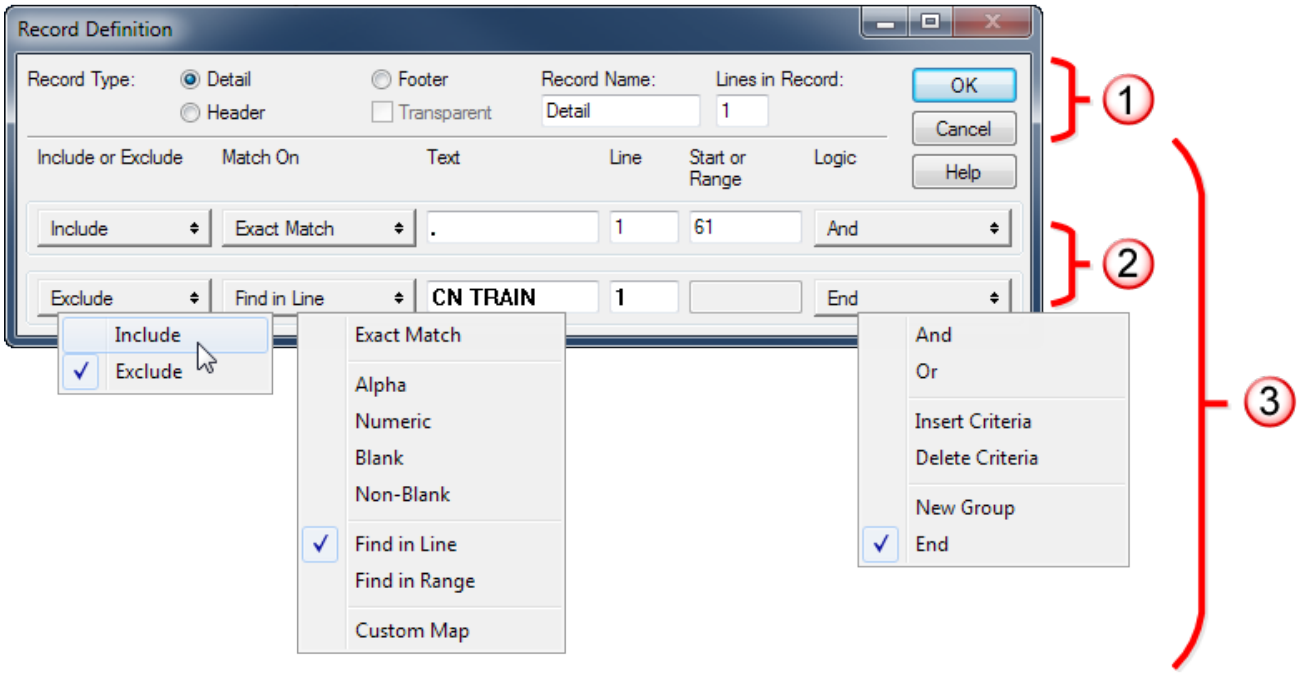
The Record Definition dialog box

The **Record Definition** dialog box is where you specify the information that delineates a set of records in the source file. It allows you to perform two main tasks:

- Specify the type of data represented by the records: detail, header, or footer
- Modify or further build the criteria used to capture the set of records

You do not start completely from scratch. Analytics will already have specified some information, which you must often fine-tune, or add to.

The figure below shows the **Record Definition** dialog box with two criteria, and with its three menus open. The menus and the adjacent text fields are what you use to modify or further build the criteria that capture the set of records.



The **Record Definition** dialog box includes the following elements:

1. Record type specification
2. Individual criteria
3. Criteria builder

The table below explains the purpose of each item in the **Record Definition** dialog box:

Item name	Purpose
Record Type: ◦ **Detail** ◦ **Header** ◦ **Footer**	Specifies the type of data represented by the records: detail, header, or footer. ◦ **Detail records** - the main information in a file For example, in a file listing overdue invoices, the invoice entries are the detail records. You can define only one set of detail records in a file. ◦ **Header records** - the identifying information that appears above blocks or subsets of detail records For example, a file might list account information for each customer (header record), followed by a list of each customer's unpaid invoices (detail record). If necessary, you can define more than one set of header records. ◦ **Footer records** - information that appears below blocks or subsets of detail records For example, a file might list subtotals for each customer's unpaid invoices (footer record). If necessary, you can define more than one set of footer records. **Note** Although header and footer data is initially treated like a separate record in the Data Definition Wizard, in the resulting Analytics table this data becomes one or more additional fields, with repeated values, added to the detail record.

Item name	Purpose
Transparent (applies to header records only)	Specifies that header records do not split multiline detail records. If a header record splits a multiline detail record in the source file, which can happen at a page break, selecting **Transparent** unifies the detail record in the resulting Analytics table.
Record Name	Allows you to customize the default record names that appear in the leftmost column in the **Data Definition Wizard**. You may find customizing a default name useful if you are creating multiple header or footer records. The value appears in the **Data Definition Wizard** only and does not appear in the resulting Analytics table.
Lines in Record	Specifies the number of lines that constitute a single record in the source file. For example, if each detail record in the source file appears on a single line, then the value must be '1'. If each detail record spans three lines, then the value must be '3'.
Include or Exclude (part of the criteria builder)	Specifies whether records that match the criteria should be included in, or excluded from, the set of records. This menu contains the following options: ◦ **Include** – include records that match the criteria ◦ **Exclude** – exclude records that match the criteria
Match On (part of the criteria builder)	Specifies the method to use, or the type of characters to use, to uniquely identify the set of records in the file. This menu contains the following options: ◦ **Exact Match** – matching records must contain the character, or string of characters, in the **Text** field, in the specified **Line** of the record, starting at the specified **Start** position ◦ **Alpha** – matching records must contain one or more alpha characters, in the specified **Line** of the record, at the specified **Start** position, or in all positions of the specified **Range** ◦ **Numeric** – matching records must contain one or more numeric characters, in the specified **Line** of the record, at the specified **Start** position, or in all positions of the specified **Range** ◦ **Blank** – matching records must contain one or more blank spaces, in the specified **Line** of the record, at the specified **Start** position, or in all positions of the specified **Range** ◦ **Non-Blank** – matching records must contain one or more non-blank characters (includes special characters), in the specified **Line** of the record, at the specified **Start** position, or in all positions of the specified **Range** ◦ **Find in Line** – matching records must contain the character, or string of characters, in the **Text** field anywhere in the specified **Line** of the record ◦ **Find in Range** – matching records must contain the character, or string of characters, in the **Text** field, in the specified **Line** of the record, anywhere in the specified **Range** ◦ **Custom Map** – matching records must contain characters that match the character pattern in the **Text** field, in the specified **Line** of the record, starting at the specified **Start** position The **Custom Map** option uses the same syntax as the MAP( ) function.
Text (part of the criteria	For **Exact Match**, **Find in Line**, or **Find in Range**, specifies the character, or string of characters, that uniquely identifies the set of records in the file.

Item name	Purpose
builder)	For **Custom Map**, specifies the character pattern that uniquely identifies the set of records in the file. The field is disabled for the other **Match On** options.
Line (part of the criteria builder)	Specifies which line of the record the criteria applies to. For example, if you create a custom map to match zip codes, and the zip codes appear on the third line of a three-line record, you must specify '3' in **Line**. For single-line records, the value is always '1'.
Start or Range (part of the criteria builder)	Specifies either: ◦ the starting byte position in the record for the comparison against the criteria ◦ the range of bytes in the record for the comparison against the criteria You can highlight a position or range in the source file to automatically populate the **Start or Range** field. You can also manually enter position or range numbers. For ranges, uses the syntax *start byte*:*end byte*.
Logic (part of the criteria builder)	Allows you to add or delete criteria, and specify the logical relations between criteria. You can add a maximum of 8 criteria. This menu contains the following options: ◦ **And** - adds an additional criterion with a logical AND ◦ **Or** - adds an additional criterion with a logical OR ◦ **Insert Criteria** - inserts an empty criterion below the criterion to which it is applied The criterion is initially inserted with a logical AND. You can change to a logical OR, but only after you have specified values for the inserted criterion. ◦ **Delete Criteria** - deletes the criterion to which it is applied ◦ **New Group** - creates a separate criteria group The **New Group** option allows you to build multiple criteria groups, which operate as separate blocks of logic. The groups are related to one another with either a logical OR or a logical AND. ◦ **End** - designates a criterion as the final criterion Selecting **End** for a criterion deletes any subsequent criteria, including criteria in other groups. **Tip** The **Logic** buttons may become unresponsive if you are missing values in a criterion. Supply any missing values to reactivate the **Logic** buttons.

Defining misaligned fields in a print image or PDF file

The procedure below outlines techniques for defining misaligned fields in a print image or PDF file. The techniques require that you only approximately define the misaligned fields in the **Data Definition Wizard**. Once the data is in Analytics, you create one or more computed fields that use Analytics functions to precisely shape the data in the fields, including aligning all values.

For information about creating a computed field, see "Define a conditional computed field" on page 734.

To define misaligned fields in a print image or PDF file:

1. Define any aligned fields in the usual manner.

   At a minimum, you must define at least one field as part of defining a set of records. Try to create a set of records that captures all the record data in the file, even if much of the data is misaligned.

   For more information about defining a set of records, see "Quick Start: How to define a print image or PDF file" on page 277.

2. Use one or more of the following techniques to define misaligned fields:
   - **Create a field definition that is long enough to capture the leftmost and the rightmost characters in the misaligned field.**

     In Analytics, you will use the ALLTRIM( ) function to align the field.
   - **Create overlapping field definitions, if necessary.**

     In some cases, data misalignment results in the values in two different fields overlapping. Define each field separately, so that all the values that belong in each field are captured by their respective field definitions.

     The same byte positions will be contained by the end of one field definition and the beginning of the other field definition. In the shared byte positions, try to capture consistently structured data – for example, a single unbroken string of characters, rather than a single string in some records, and two separate strings, or no characters, in other records.

     In Analytics, you will use the ALLTRIM( ), the REGEXREPLACE( ), and the SPLIT( ) functions to align the fields and get rid of unwanted characters.
   - **Create a single, long field definition that encompasses multiple misaligned fields.**

     If an entire section of a set of records is misaligned, you can use a long field definition to capture the problematic section of the record data. The field must be long enough to capture the leftmost and the rightmost characters in the block of misaligned data.

     If misaligned data occurs in separate sections of a set of records, create additional long field definitions to capture each misaligned section.

> **Note**
> If the values in a field vary in the number of words they contain, try to create a separate field definition for these values, or ensure that these values represent the last field at the end of a long field definition encompassing multiple misaligned fields. The “Product Description” field in the sample “Inventory.pdf” is an example of a field in which values vary in number of words.

In Analytics, you will use the ALLTRIM( ), the REGEXREPLACE( ), and the SPLIT( ) functions to break up the single field into separate, aligned data elements.

3. Check the entire file to ensure that none of the values in the misaligned fields are outside the aqua-blue highlighting of their field definition. Adjust the length of the field definition, if required.
4. Make sure that a data type of **Character** is specified for each field definition in the **Field Definition** dialog box.
5. Complete the import process in the usual manner.

   In the **Edit Field Properties** page, make sure that a data type of ASCII or UNICODE is specified for every field.

   For more information, see "Define and import a print image file" on page 283, or "Define and import a PDF file" on page 291.
6. For a misaligned field with no data from an overlapping field, create a computed field in Analytics that uses the following expression:

   ```
   ALLTRIM(misaligned_field_name)
   ```

   Leading and trailing spaces are removed from the field, which aligns all values in the field.
7. For a misaligned field that contains data from an overlapping field, do the following in Analytics:
   a. Create an initial computed field that uses the following expression to replace one or more spaces between the field value and the unwanted characters with a single space:

      ```
      ALLTRIM(REGEXREPLACE(misaligned_field_name, "\s+", " "))
      ```

      The expression also removes leading and trailing spaces from the misaligned field.
   b. Create a second computed field that uses one of the following expressions to extract the field value and discard the unwanted characters.
      - If the unwanted characters are at the end of the field, use this expression:

        ```
        SPLIT(initial_computed_field_name," ", 1)
        ```

      - If the unwanted characters are at the beginning of the field, use this expression:

```
SPLIT(initial_computed_field_name," ", 2)
```

> **Tip**
>
> If unwanted characters are sometimes at the end of a field, and sometimes at the beginning, or if they are present in only some of the records, you need to create a conditional computed field that applies different versions of the SPLIT( ) expression to different parts of the misaligned field. For example, the condition RECNO( ) > 100 allows you to apply a version of the expression to only those records beyond the first 100 records.
>
> For more information, see "Define a conditional computed field" on page 734.

8. For a long field definition that encompasses multiple misaligned fields, do the following in Analytics:
   a. Create an initial computed field that uses the following expression to replace one or more spaces between data elements with a single space:

      ```
      ALLTRIM(REGEXREPLACE(long_field_name, "\s+", " "))
      ```

      The expression also removes leading and trailing spaces from the long field.

      > **Tip**
      >
      > You may find including the OMIT( ) function in the expression is useful for removing pieces of data that appear inconsistently and complicate subsequent processing. For example, OMIT(ALLTRIM(REGEXREPLACE (*long_field_name*, "\s+", " ")), "-") does the same as the expression above, and also removes hyphens.

   b. Create a second computed field that uses this expression to extract the first data element:

      ```
      SPLIT(initial_computed_field_name," ", 1)
      ```

   c. Create as many additional computed fields as required, using variations of the same expression, to extract all the data elements.

      For example:

      ```
      SPLIT(initial_computed_field_name," ", 2)
      SPLIT(initial_computed_field_name," ", 3)
      ```

      To specify successive data elements, keep increasing the number in the *segment* parameter of the SPLIT( ) function.

> **Note**
> For field values that contain more than one word, such as the values in the "Product Description" field in the sample "Inventory.pdf", this technique isolates each word in a separate field. If required, you can reunite the values by concatenating the separate fields. For more information, see "Concatenating fields" on page 218.

9. Once you have finished extracting all the data elements to separate fields, do the following to convert numeric and datetime data to the appropriate data type:
   a. For numeric fields, create a computed field that uses this expression:

      ```
      VALUE(field_name, number_of_decimal_places)
      ```

      For more information, see "VALUE( ) function" on page 2438.
   b. For date fields, create a computed field that uses this expression:

      ```
      CTOD(field_name, "date_format")
      ```

      For more information, see "CTOD( ) function" on page 2078.

      To converted datetime or time values, use the CTODT( ) or the CTOT( ) functions.

      > **Tip**
      > You can save labor, and create fewer computed fields, by converting the data type at the same time you apply functions to correct misaligned data. For example:
      > `VALUE(ALLTRIM(misaligned_field_name), 2)`

10. Once you have created all the required computed fields, add them to the table view.

    You do not need to add the initial computed field to the view, and you can remove any misaligned fields, or long field or fields, from the view.

    For more information, see "Add columns to a view" on page 778, or "Remove columns from a view" on page 779.

Defining and importing subsets of print image or PDF data

If defining a complete set of records in a print image or PDF file is difficult or even impossible because of misaligned data, you can define and import multiple subsets of data from the file. In Analytics, you can then append the resulting Analytics tables to assemble a complete data set.

This technique works best if the source file in the **Data Definition Wizard** contains blocks of records in which all fields are aligned within each block. If the data is more randomly misaligned, see "Defining misaligned fields in a print image or PDF file" on page 308.

> **Tip:**
> For PDF definition, you have the option of parsing the PDF file on a page-by-page basis. In some cases, data misalignment occurs across page breaks. You may be able to solve an alignment issue by using page-sized subsets of data.

To define and import a subset of print image or PDF data:

1. Perform the definition and import process in the usual manner, with these differences:
   - **Define and import the same file multiple times.**

     With each iteration, define a different subset of records. The fields in each subset must be internally aligned.

     A subset of records does not need to be contiguous. For example, the fields in records at the start and at the end of a file could be aligned with each other, but misaligned with fields in the middle of the file.
   - **Devise a method for keeping track of which records are included in each subset.**

     If you unintentionally capture the same record more than once, you can remove duplicate records from the reassembled data set in Analytics. For more information, see "Remove duplicates" on page 1211.
   - **With each iteration, make sure the data structure remains consistent.**

     Ensure that the name, the length, the data type, and the order of corresponding fields remain consistent. Maintaining this consistency of data structure makes appending the resulting Analytics tables much easier.

> **Tip:**
> After importing the first subset, open the resulting table in Analytics, and enter `DISPLAY` in the command line to display the data structure of the table layout. Use the displayed table layout information as a guide for creating the subsequent subsets of records and fields.
>
> To save labor, use the generic Analytics field names (“Field_1”, “Field_2”, and so on) when defining and importing subsets of records. Once you have reassembled the data set in Analytics, you can rename all the fields in the reassembled table.

2. When you save each Analytics data file, and each Analytics table layout, use an incrementing numeric suffix to prevent overwriting tables you have already created. For example, “Table_1.fil”, “Table_2.fil”, and so on.
3. Once you have defined and imported all the records in the source file, append the multiple Analytics tables.

   For more information, see "Extracting and appending data" on page 870.

Working with multiline records and fields

You can define record or field data that extends beyond one line or one row in a source file. For example:

- Address data or comment data arranged on multiple lines
- Different types of data stacked together in a single field
- Multiline fields with values that vary in the number of lines they contain

The sections that follow explain how to define files with data arranged in this manner.

Multiline records versus multiline fields

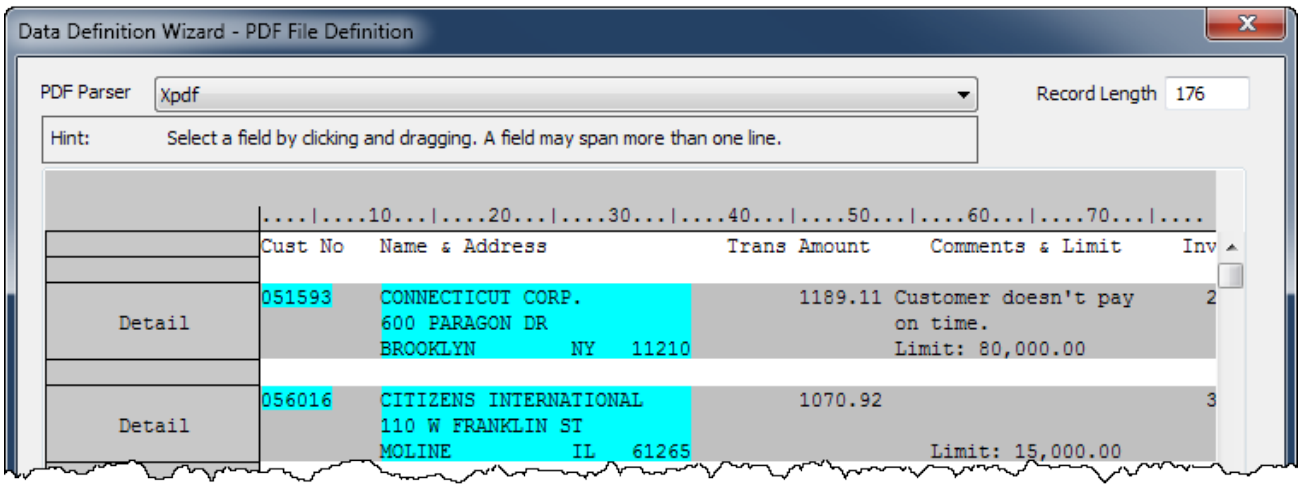
In a source file, multiline records contain data on more than one line or row that all belongs to the same record. (See "Multiline record with a one-line field and a multiline field" below.)

The fields in a multiline record may or may not be multiline fields. For example, a two-line record could be a succession of one-line field values that have been wrapped to a second line by the layout of a print image or PDF report.

Multiline fields contain field values that span two or more lines or rows in the source file. For example: addresses that are arranged on multiple lines, or comment fields with text on multiple lines. If a field is multiline, the record containing the field must also be multiline.

Multiline record with a one-line field and a multiline field

The example below shows a three-line record that contains the one-line “Cust No” field and the three-line “Name & Address” field.



Defining a multiline record

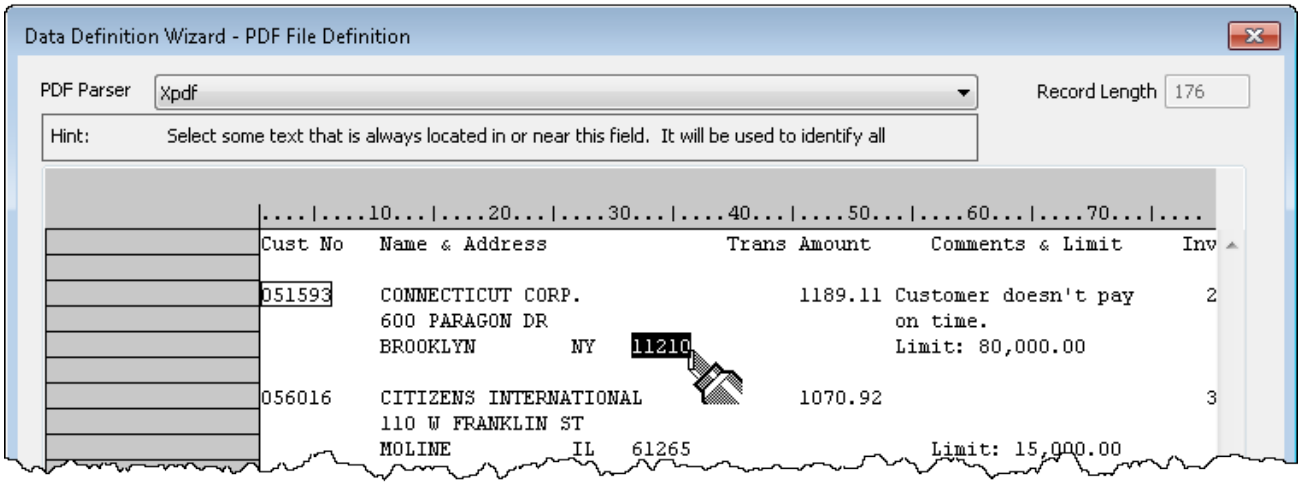
You can define a multiline record using either of these methods:

- Select an initial data value in the first line of a record, and a unique value in the last line of the record.

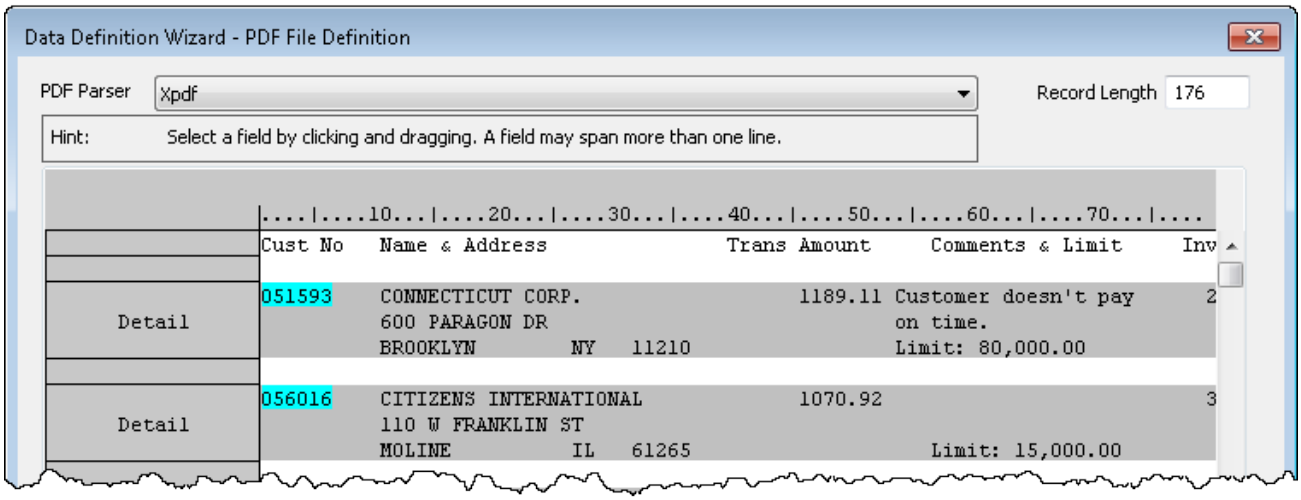
  In "Initial data value in the first line of a record and unique value in the last line" below, a customer number is selected in the first line, and a zip code is selected in the last line. "Defined multiline record" on the next page shows the resulting multiline record.

- Define the first line of a record, and then in the **Record Definition** dialog box, edit the record definition by specifying the appropriate number of lines in the **Lines in Record** setting. You can use this method when it is not possible to specify a unique value in the last line of the record.

Initial data value in the first line of a record and unique value in the last line



Defined multiline record



Defining fields in a multiline record

You have the following options when defining fields in a multiline record:

- **You can define values that occupy a single line as a single-line field**

  In "Single-line and multiline fields" on the facing page, the following values are all defined as single-line fields: customer number, city, state, zip code, transaction amount, and limit.

  To define a single-line field, select a single instance of one of the values in the field.

- **You can define values that occupy multiple lines in one of two ways:**
  - **As multiple fields each containing the data from a single line**

    In "Single-line and multiline fields" on the facing page, the customer name and street address are defined in this manner.

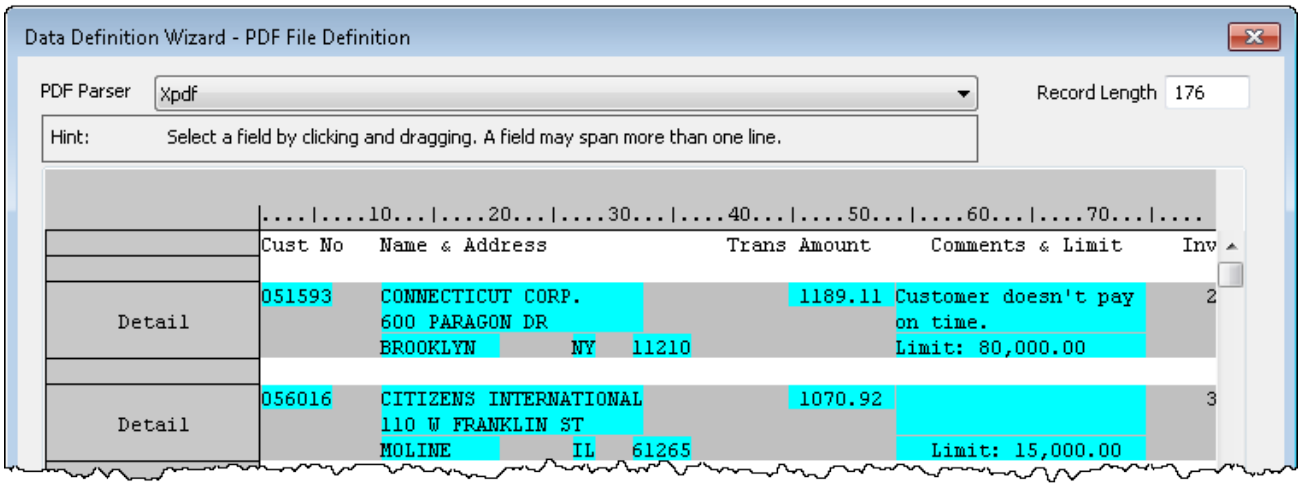
    To define multiple, single-line fields, select a single instance of the multiline data. By default, Analytics creates a separate field for each line. For each field, an incrementing numeric suffix is added to the field name.

  - **As a single field containing the data from all lines**

    In "Single-line and multiline fields" on the facing page, the comment values are defined as a single field.

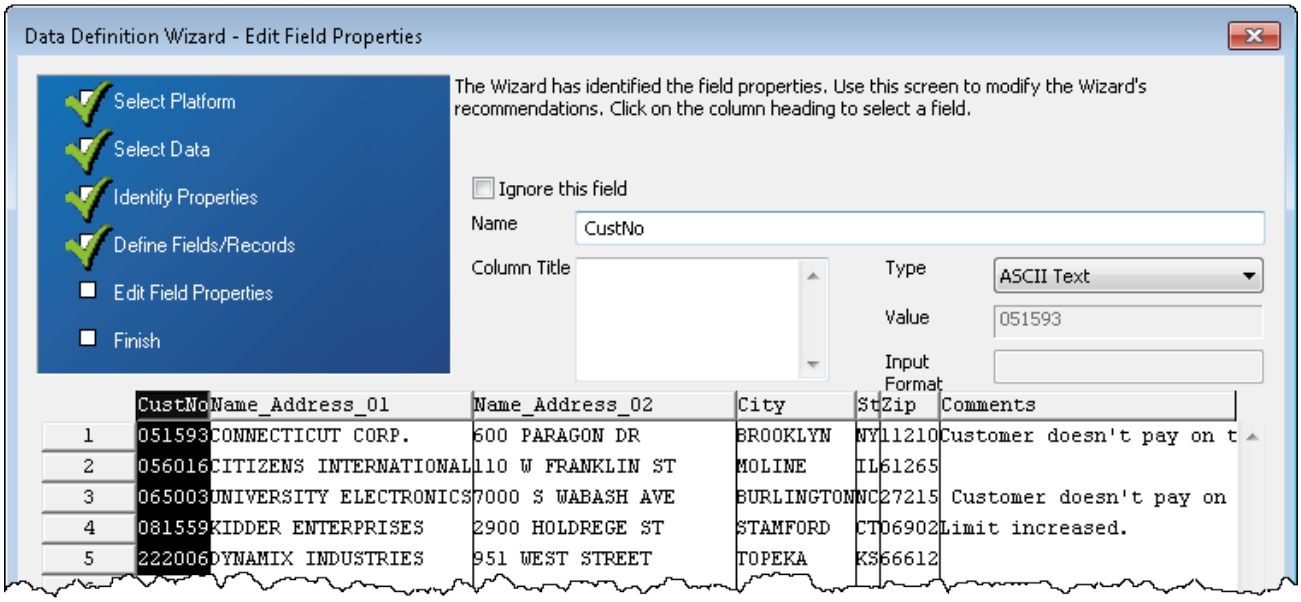
    To define a single field containing the data from all lines, select a single instance of the multiline data. In the **Field Definition** dialog box, under **Advanced Options**, select **Convert to single field**.

Single-line and multiline fields



Preview of the fields in the resulting Analytics table

The example below shows how the defined fields in "Single-line and multiline fields" above appear in the resulting Analytics table.



Defining fields that vary in height

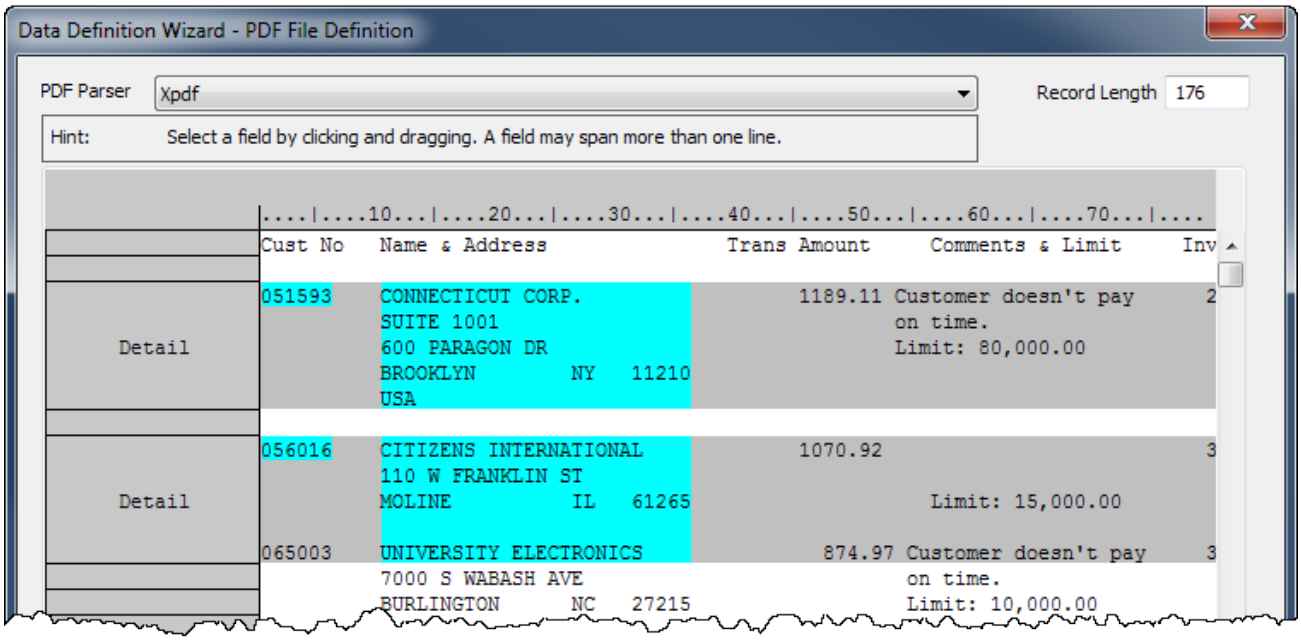
You can use the **Ends on blank line** setting in the **Field Definition** dialog box to define multiline fields in which the values vary in the number of lines they occupy. The setting specifies that values

terminate when they encounter a blank line. This feature only works if one or more blank lines separate each value in the multiline field.

Mismatch between field boundaries and field values

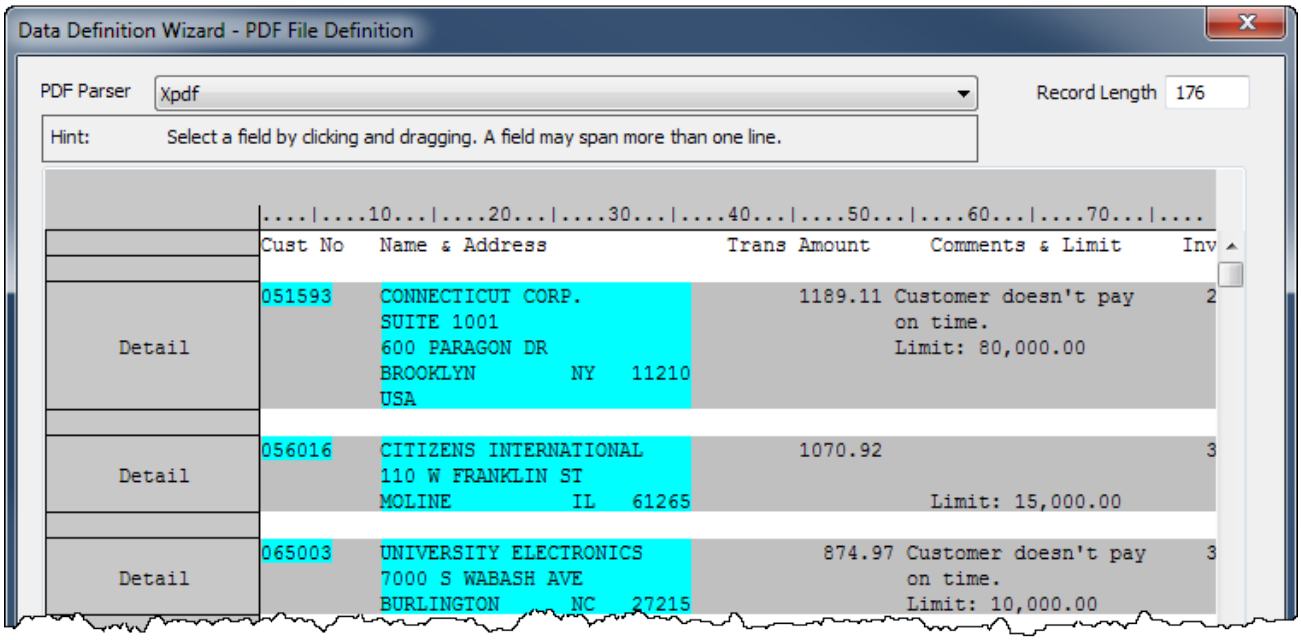
The example below shows the problem that can occur when field values vary in height.

The field height is set to '5' to capture all five lines of the first value in the "Name & Address" field. Because the second value has only three lines, the five-line field height captures too much data, encompassing all of the second value, and the first line of the third value. The result is a mismatch between field boundaries and field values, which also creates mismatched record boundaries.



Field boundaries resized to fit field values

The example below shows how the problem can be solved by selecting **Ends on blank line**. Now, field and record boundaries dynamically resize to fit the number of lines occupied by each field value.


Data Definition Wizard - PDF File Definition
PDF Parser
Xpdf
Record Length
176
Hint:
Select a field by clicking and dragging. A field may span more than one line.
....|....10...|....20...|....30...|....40...|....50...|....60...|....70...|....
Cust No
Name & Address
Trans Amount
Comments & Limit
Inv
Detail
051593
CONNECTICUT CORP.
SUITE 1001
600 PARAGON DR
BROOKLYN NY 11210
USA
1189.11
Customer doesn't pay on time.
Limit: 80,000.00
2
Detail
056016
CITIZENS INTERNATIONAL
110 W FRANKLIN ST
MOLINE IL 61265
1070.92
Limit: 15,000.00
3
Detail
065003
UNIVERSITY ELECTRONICS
7000 S WABASH AVE
BURLINGTON NC 27215
874.97
Customer doesn't pay on time.
Limit: 10,000.00
3

Import an ACCPAC master file

You can create an Analytics table by defining and importing an ACCPAC master file. The file can be on your local computer or a network drive, or in a folder located on an Analytics Server (if installed).

1. Select **File > New > Table**.

   The first page displayed in the **Data Definition Wizard** depends on your configuration. If integration with Analytics Server is enabled the **Select Platform for Data Source** page is displayed, otherwise the **Select Local Data Source** page is displayed.
2. Complete one of the following steps to select the location of the file:
   - If the **Select Platform for Data Source** page is displayed and you want to use Analytics to define the file, select **Local** and click **Next**. In the **Select Local Data Source** page select **File** and click **Next**.
   - If the **Select Platform for Data Source** page is displayed and you want to use an Analytics Server to define the file, select **ACL Server** and select the Windows server profile from the drop-down list, and then click **Next**. In the **Select ACL Server Data Source** page select **Flat Files** and click **Next**.
   - If the **Select Local Data Source** page is displayed select **File** and click **Next**.
3. In the **Select File to Define** page, locate and select the file you want to create the Analytics table from and click **Open**.
4. In the **Character Set** page, verify that the correct character set option has been selected and click **Next**.
5. In the **File Format** page, verify that the **ACCPAC master file** option has been selected and click **Next**.
6. In the **Identify Fields** page, complete any of the following actions to modify the fields identified in the record:
   - Delete an existing field separator by clicking on the field separator line you want to remove.
   - Move an existing field separator by clicking and dragging the field separator line to the new location.
   - Create a new field separator by clicking the grid in the position where you want to add the field separator.
7. When you have identified all of the fields in the record, click **Next**.
8. In the **Edit Field Properties** page, you can modify the name and properties for each field by selecting the column header for the field you want to modify in the preview table and updating any of the following properties:
   - **Ignore this field** – If you do not want the field to be included in the Analytics table layout, select this checkbox.
   - **Name** – Keep the name assigned by Analytics for the field in the table layout, or enter a different name.

- **Column Title** – Enter the column title to display in the default Analytics view. If a column title is not specified the **Name** value is used.
- **Type** – Select the appropriate data type from the drop-down list. For information about the supported data types in Analytics, see "Data types in Analytics" on page 739.

  The **Decimal** and **Input Format** text boxes appear automatically when you select the corresponding data type.
- **Value** – A read-only property that displays the first value in the field. The value is updated based on any edits you make.
- **Decimal** (numeric fields only) – Specify the appropriate number of decimal places.
- **Input Format** (datetime fields only) – Specify the format that matches the data. For more information about date and time formats, see "Formats of date and time source data" on page 347.

9. Click **Next** after you have finished editing the field properties you want to change.
10. In the **Final** page, verify the settings for the new Analytics table and click **Finish**.
11. Enter a name for the Analytics table you are adding to the project, or keep the default name, and click **OK**.

Import a dBASE-compatible file

You can create an Analytics table by importing any file that conforms to the dBASE file format standard. The **Data Definition Wizard** parses the data in the dBASE-compatible file and uses this information to automatically define the Analytics table layout.

The dBASE file format

Exporting data in the dBASE file format is an option found in many databases and business applications, so it can be used to access data from a variety of data sources. If the exported file conforms to the dBASE standard, this is a convenient way to access data with Analytics because the file contains all of the required field definitions. If the exported data does not fully conform to the dBASE standard, Analytics may be unable to define the file.

The Record_Deleted field

dBASE files always include a field called “Record_Deleted” that is used to keep track of deleted records. Analytics defines a corresponding field in the table layout, which is not required and you can delete it from the table layout after removing it from the default view.

dBASE file criteria

Data exported to a dBASE file must meet the following criteria in order to be imported to Analytics:

Column names	Column names can have a maximum length of ten characters. The first ten characters of each field must be unique or the duplicate fields cannot be exported.
Field names	Field names must be specified in the first row, and data must start in the second row.
Data type	Each column should contain values of only one data type. For example, if the first value in a field contains character data, the field will be exported as character data.
Fields with numbers	Fields that contain only numbers will be exported as numeric data. In some cases, this will result in a field with the wrong data type in Analytics. For example, invoice numbers are numeric values, but they are often stored in character fields. When this occurs, you need to change the field data type in the **Table Layout** dialog box.

Steps

1. Select **File > New > Table**.

   The first page displayed in the **Data Definition Wizard** depends on your configuration. If integration with Analytics Server is enabled the **Select Platform for Data Source** page is displayed, otherwise the **Select Local Data Source** page is displayed.

2. Complete one of the following steps to select the location of the file:
   - If the **Select Platform for Data Source** page is displayed and you want to use Analytics to define the file, select **Local** and click **Next**. In the **Select Local Data Source** page select **File** and click **Next**.
   - If the **Select Platform for Data Source** page is displayed and you want to use an Analytics Server to define the file, select **ACL Server** and select the Windows server profile from the drop-down list, and then click **Next**. In the **Select ACL Server Data Source** page select **Flat Files** and click **Next**.
   - If the **Select Local Data Source** page is displayed select **File** and click **Next**.
3. In the **Select File to Define** page, locate and select the file you want to create the Analytics table from and click **Open**.

   dBASE-compatible files have a **.dbf** file extension.

4. In the **File Format** page, verify that the **dBASE compatible file** option has been selected and click **Next**.
5. In the **Final** page, verify the settings for the new Analytics table and click **Finish**.
6. Enter a name for the Analytics table you are adding to your project and click **OK**.

Import an SAP Audit Format file

You can create an Analytics table by defining and importing an SAP Audit Format file (formerly known as the SAP Private File Format). SAP Audit Format files have a .dat file extension.

You can define and import Audit Format files located on your local computer or on a network drive.

> **Note**
>
> Galvanize offers two utilities for directly accessing an SAP system and importing data to Analytics:
>
> - **SAP connector** – one of the Analytics connectors, available with an additional subscription
> - **Direct Link** – an optional add-on that can be purchased from Galvanize

SAP AIS and SAP DART

An SAP Audit Format file is produced by an SAP data extract. SAP AIS (Audit Information System) and SAP DART (Data Retention Tool) are both SAP utilities that can produce data extracts.

When creating extracts using these SAP utilities, the SAP user can specify the file output format. Some Analytics-friendly output formats are spreadsheet, text, dbf, and the SAP Audit Format. These extracts can then be imported into Analytics using the Data Definition Wizard. The ideal format to use is the SAP Audit Format because it requires the least amount of effort to import into Analytics.

Steps

1. Select **File > New >Table** .

   The first page displayed in the **Data Definition Wizard** depends on your configuration. If integration with Analytics Server is enabled the **Select Platform for Data Source** page is displayed, otherwise the **Select Local Data Source** page is displayed.

2. Complete one of the following steps to select the location of the file:
   - If the **Select Platform for Data Source** page is displayed and you want to use Analytics to define the file, select **Local** and click **Next**. In the **Select Local Data Source** page select **File** and click **Next**.
   - If the **Select Platform for Data Source** page is displayed and you want to use an Analytics Server to define the file, select **ACL Server** and select the server profile from the drop-down list, and then click **Next**. In the **Select ACL Server Data Source** page select **Flat Files** and click **Next**.
   - If the **Select Local Data Source** page is displayed select **File** and click **Next**.

3. In **Select File to Define**, locate and select the file you want to create the Analytics table from and click **Open**.
4. In the **Character Set** page, verify that the correct character set option has been selected and click **Next**.
5. In the **File Format** page, verify that the **SAP private file format / DART** option has been selected and click **Next**.
6. In the **SAP Private File Format** page, select the appropriate option for field naming:
   - **Use local language field descriptions as ACL field names** – Select this option to use the localized field descriptions configured for the SAP system, instead of the standard German language field names. This option is recommended if the Analytics table will be used in only one language.
   - **Use standard-delivered SAP German abbreviations as ACL field names** – Select this option if you prefer to use the German field names, or if the Analytics table will be used in multiple languages.
7. Click **Next**.
8. In the **Save Converted SAP File As** dialog box, enter the file name and modify the folder location for the Analytics data file, as necessary, and click **Save**.
9. In the **Final** page, verify the settings for the new Analytics table and click **Finish**.
10. Enter a name for the Analytics table you are adding to your project and click **OK**.

Import an XML file

You can create an Analytics table by defining and importing an XML file. The **Data Definition Wizard** allows you to select the XML elements to import, configure the structure of the resulting Analytics table, and customize column headings and data types for the elements you are importing.

> **Note**
>
> In some cases, you may need to adjust one or more field definitions in the resulting Analytics table so that the data displayed in the view accurately reflects the data in the source XML file. You adjust field definitions in the Analytics table layout.
>
> Analytics imports the exact raw data contained in an XML file, and you can see this source data in the table layout. On occasion, a field definition created during the table definition and import process misinterprets the source data, and the definition requires subsequent adjustment. For example, a numeric field could be misinterpreted as a date field, and dates rather than numbers may initially appear in the view.

1. Select **File > New > Table**.
2. If the **Select Platform for Data Source** page is displayed, select **Local** and click **Next**.
3. In the **Select Local Data Source** page, select **File** and click **Next**.
4. In the **Select File to Define** dialog box, locate and select the XML file you want to create the Analytics table from and click **Open**.

   XML files have a **.xml** file extension.
5. In the **File Format** page, verify that the **XML file** option has been selected and click **Next**.
6. In the **Select XML Data Structures** page, select one or more XML data structures to include in the resulting Analytics table:
   a. To select a data structure, click the name of the associated XML element.
   b. Click **Add** to add the data structure to the **Preview** pane.

      > **Note**
      >
      > Generating a preview of the data in a large XML file can be slow, so the **Auto Preview** option is automatically deselected for XML files larger than 2 GB.

   c. Select and add all the data structures you want to include in the Analytics table.
   d. If necessary, select a data structure in the **Preview** pane and click **Remove** to remove it.
   e. Click **Next**.

   An XML data structure is a collection of XML elements and attributes. For more information, see "Selecting XML data structures" on page 328.
7. In the **Select XML Elements** page, fine-tune the selection of XML elements and attributes and click **Next**.

   For more information, see "Selecting and configuring XML elements" on page 329.

8. In the **Preview Data** page, modify the name or properties for any field, if necessary.

   To modify field properties, select the appropriate column heading in the preview table, in the bottom half of the page, and update any of the following properties:

   - **Name** – Keep the name assigned by Analytics for the field in the table layout, or enter a different name.
   - **Column Title** – Enter the column title to display in the default Analytics view. If a column title is not specified the **Name** value is used.
   - **Type** – Select the appropriate data type from the drop-down list. For information about the supported data types in Analytics, see "Data types in Analytics" on page 739.

     The **Decimal** and **Input** text boxes appear automatically when you select the corresponding data type.
   - **Value** – A read-only property that displays the first value in the field. The value is updated based on any edits you make.
   - **Decimal** (numeric fields only) – Specify the appropriate number of decimal places.
   - **Input** (datetime fields only) – Specify the format that matches the data. For more information about date and time formats, see "Formats of date and time source data" on page 347.

9. Click **Next**.
10. In the **Save Data File As** dialog box, enter a name for the Analytics data file, and if necessary modify the location where the file will be saved, and click **Save**.
11. In the **Final** page, verify the settings for the new Analytics table and click **Finish**.
12. Enter a name for the Analytics table you are adding to the project, or keep the default name, and click **OK**.
13. Review the data in the new Analytics table and update any field definitions, if required.

    If a field has the wrong data type specified, the data may not appear in the view, or the data may be misinterpreted. For example, a numeric value may be interpreted as a date.

14. To update a field definition, do the following:
    a. Select **Edit > Table Layout**.
    b. In the **Edit Fields/Expressions** tab, double-click the field you want to modify.
    c. Make the necessary changes to the field definition and click **Accept Entry**  .

       For example, you may need to change the **Type** from **Datetime** to **Numeric**.

Selecting XML data structures

Use the **Select XML Data Structures** page of the wizard to select one or more XML data structures to include in an Analytics table.

XML data structures consist of elements, nested child elements, and attributes that Analytics identifies when it analyzes an XML file. They are displayed in the **XML Data Structures** treeview, which is a hierarchical representation of the XML file. Each XML data structure is represented by a table icon  , and the name of the XML element and nested elements or attributes that it contains.

Prior to selecting one or more XML data structures you should review the XML file and determine a suitable Analytics table structure for your audit objectives. With that table structure in mind, select only XML data structures with columns that match columns in the intended table structure. You can use the subsequent pages in the wizard to fine-tune the individual elements to include, and modify column properties.

If you need to analyze an XML file with a complex structure, you may need to import the XML file more than once. You can select different data structures for each Analytics table you create, and then join or relate the tables in Analytics.

To select data structures:

1. In the **XML Data Structures** treeview, click the name of an XML element to select the associated data structure for inclusion in the resulting Analytics table.

   The names of the columns in the data structure, with sample data, are displayed in the **Sample Structure** pane. Names preceded by the @ symbol are derived from an XML attribute rather than an XML element.

   > **Note**
   > If more than one instance of a nested element exists within the data structure, the repeated element may not be listed in the data structure in the treeview. You can select the specific instances of the repeated element in a subsequent page of the wizard.

2. Click **Add** to add the data structure to the **Preview** pane.

   All the displayed elements and attributes in the selected data structure are added, with column names that correspond to the element and attribute names.

   The data is displayed if **Auto Preview** is checked.

   > **Note**
   > Generating a preview of the data in a large XML file can be slow, so the **Auto Preview** option is automatically deselected for XML files larger than 2 GB.

3. Repeat steps 1 to 2 to add any additional data structures that you want to include in the Analytics table.
4. Click **Next**.

Selecting and configuring XML elements

Use the **Select XML Elements** page of the wizard to select or fine-tune the individual XML elements and attributes that will appear in the resulting Analytics table. You can also modify column properties, if necessary.

In the **XML Elements** treeview, Analytics provides a hierarchical representation of all elements and attributes contained in the XML file. A green check mark identifies elements or attributes you previously selected. If you previously selected multiple data structures, they are now combined in a single **Preview** table. Data from the selected columns is displayed if **Auto Preview** is checked.

> **Note**
>
> Generating a preview of the data in a large XML file can be slow, so the **Auto Preview** option is automatically deselected for XML files larger than 2 GB.

To select and configure elements:

1. If you want to remove a column, select the column in the **Preview** table and click **Remove**.
2. If you want to add a column, select the element or attribute in the **XML Elements** treeview and click **Create Column**.

   > **Note**
   >
   > If adding an element violates your intended Analytics table structure and causes gaps to appear in the table, data of the same kind probably needs to be merged in a single column (step 4 below).
   >
   > If adding an element creates duplicate or multiple identical records (identical except for the added element), more than one instance of the element probably exists within a parent element and the instances need to be split into separate columns.

3. If you want to move a column, select the column and click the left or right arrow button, or drag the column to a new position.
4. If you want to merge data of the same type in a column, select the column in the **Preview** table, select the element or attribute to add in the **XML Elements** treeview, and click **Add to Column**.
5. If you want to modify column properties, select the column in the **Preview** table and click **Column Properties**.

   Make any of the following changes in the **XML Column Properties** dialog box and click **OK**:

   - Change the name of the column.
   - Change the data type of a column.

- If the column is assigned the Text data type, repeat the name of the column in each row of the column.
- If an element is repeated in the data structure, assign specific instances of the repeated element to the column. For example, if there are multiple `<description>` elements in a data structure, you could assign only the first instance to a Description column.
- Remove a specific column from a merged column.

6. Click **Next**.

Modifying XML column properties

Use the **XML Column Properties** dialog box to modify properties of the selected column. You can specify how the field will be defined in the Analytics table, and the data that will be retrieved from the XML file.

The **Source** column displays the hierarchy of each element selected as the field's data source. For example, a value of `/catalog/cd/title/` describes the following nested structure in the XML file:

```
<catalog>
  <cd>
    <title></title>
  </cd>
</catalog>
```

To modify the properties of a column:

1. If you want to change the name of the column, enter the new name in the **Column Name** text box.

   The default value for the field is the name of the first source element. The **Column Name** value is the name of the field in the Analytics table layout.

2. If you want to change the data type of the column select Text, Numeric, or Datetime in the **Column Type** drop-down list.

   If Analytics cannot determine the datetime format, enter the **Date Format** that matches the data. The **Date Format** field appears when **Datetime** is selected.

   If you change the column data type and data no longer appears in the column, the data type and the data are mismatched.

3. If you want to remove an XML element from a column, select the element in the **Source** column and click **Remove**.
4. If you want to display the name of the XML element in each row, instead of the data stored in the element, select Name in the **Type** drop-down list. This option only applies to columns where the column type is text.
5. If there are multiple instances of the same XML element at the same level in the hierarchy, and you want to choose only one element or a subset, you can use the **Instance** field to specify the instance(s) to select.

   For example, if you entered a value of 2 in the **Instance** field, only the value in second `<description/>` element would be selected for the column from the following example:

   ```
   <organization>
     <company/>
     <description/>
   ```

```
  <department/>
  <description/>
</organization>
```

You can enter a single number, multiple numbers separated by commas, a numeric range, or a combination of these. For example, if you want to include the first instance, and the fifth to tenth instances, in a single column, enter the following: `1, 5-10`. By default, all instances are initially displayed in the column.

> **Tip:**
> If you want to add different instances of a repeated element to different columns in an Analytics table, you need to create a column for each instance and set the **Instance** field to the appropriate value.

6. Click **OK**.

About XML files

XML (eXtensible Markup Language) is a markup language commonly used to transmit data between computer systems or applications.Analytics can import any well-formed XML document. A well-formed document is one that follows XML syntax rules.

XML file structure

XML files are structured in a standard way and support any number of hierarchy levels. Hierarchy levels are represented by nesting XML elements within other elements. A data structure is a group of elements that can be mapped to an Analytics table. Each data structure is identified with a table icon  in the **Data Definition Wizard**. When you define an XML file, Analytics identifies any data structures in the file that can be used to create an Analytics table.

Mapping XML data structures to Analytics tables

When you import a simple XML document with only one data structure, it can be mapped directly to an Analytics table. If you have a more complex XML file with several data structures, you may be able to combine the multiple data structures into a single Analytics table. If combining multiple data structures does not give the desired results, you can import each data structure into a separate Analytics table. You can then define relations between the resulting Analytics tables using the **Relate Tables** command.

File extensions

XML files typically use a standard file extension (.xml). In some cases, other file extensions are used, and the first line of the document identifies it as an XML file. If a non-standard file extension is used, you need to manually select the **XML file** format in the **Data Definition Wizard**.

XML elements and attributes

XML uses elements and attributes to identify the structure and content of data. Analytics can import both elements and attributes.

An **element** is a unit of XML data delimited by tags, and each XML element can enclose other elements. In the following example, the **name** element defines the value “John Smith”:

```
<name>John Smith</name>
```

An **attribute** provides additional information about an element. In the following example, the type attribute specifies that the account element represents a checking account:

```
<account type="checking">991110101</account>
```

In the **Data Definition Wizard**, attribute names are automatically preceded by the @ symbol to distinguish them from element names. For example, an attribute named “type” is displayed as “@type”.

XML sample

XML files usually include a mixture of elements and attributes, and at least one data structure. The following example shows the contents of a simple XML file that contains two client records:

```
<?xml version="1.0"?>
<accounts>
  <client>
    <name>John Smith</name>
    <ID>JS004</ID>
    <account type="checking">991110101</account>
    <account type="savings">991110808</account>
  </client>
  <client>
    <name>Jane Smith</name>
    <ID>JS005</ID>
    <account type="checking">771110103</account>
    <account type="savings">771110303</account>
  </client>
</accounts>
```

Import an XBRL file

You can create an Analytics table by defining and importing an XBRL file. The **Data Definition Wizard** allows you to select the elements to import, and customize column headings and data types for the elements you are importing.

1. Select **File > New > Table**.
2. If the **Select Platform for Data Source** page is displayed, select **Local** and click **Next**.
3. In the **Select Local Data Source** page, select **File** and click **Next**.
4. In the **Select File to Define** dialog box, locate and select the file you want to create the Analytics table from and click **Open**.

   XBRL 2.1 files have a **.xbrl** or **.xml** file extension. The difference between an XBRL file and other XML files is that in the XBRL file the top-level, or root, element tag is `<xbrl>`.
5. In the **File Format** page, verify that the **XBRL 2.1 file** option has been selected and click **Next**.
6. In the **Select XBRL Contexts to Import** page, select the XBRL contexts to include in the Analytics table and click **Next**. For details on this process, see "Selecting XBRL contexts" on page 338.
7. In the **Select Elements to Import** page, select the elements to include in the Analytics table and click **Next**. For details on this process, see "Selecting XBRL elements" on page 337.
8. In the **Preview Data** page, you can modify the name and properties for each field by selecting the appropriate column heading in the preview table, in the bottom half of the page, and updating any of the following properties:
   - **Ignore this field** – If you do not want the field to be included in the Analytics table layout, select this checkbox.
   - **Name** – Keep the name assigned by Analytics for the field in the table layout, or enter a different name.
   - **Column Title** – Enter the column title to display in the default Analytics view. If a column title is not specified the **Name** value is used.
   - **Type** – Select the appropriate data type from the drop-down list. For information about the supported data types in Analytics, see "Data types in Analytics" on page 739.

     The **Decimal** and **Input** text boxes appear automatically when you select the corresponding data type.
   - **Value** – A read-only property that displays the first value in the field. The value is updated based on any edits you make.
   - **Decimal** (numeric fields only) – Specify the appropriate number of decimal places.
   - **Input** (datetime fields only) – Specify the format that matches the data. For more information about date and time formats, see "Formats of date and time source data" on page 347.
9. Click **Next** after you have finished editing the field properties you want to change.
10. In the **Save Data File As** dialog box, enter a name for the Analytics data file, and if necessary modify the location where the file will be saved, and click **Save**.

11. In the **Final** page, verify the settings for the new Analytics table and click **Finish**.
12. Enter a name for the Analytics table you are adding to the project, or keep the default name, and click **OK**.

Selecting XBRL elements

Use the **Select Elements to Import** page of the wizard to select the XBRL elements to include in your Analytics table.

The **Elements** table lists all of the elements associated with the context(s) you previously selected. By default, all the elements are initially selected for inclusion in the Analytics table.

To select the elements to import:

1. Do any of the following:
   - Select or deselect the checkbox next to individual elements.
   - Click **Select All** to select all of the elements.
   - Click **Deselect All** to deselect all of the elements.
   - Click **Reverse Selection** to select all of the deselected elements and deselect all of the selected elements.

   The **Import Size** panel displays the number of individual records that will be imported into the Analytics table and the total number of records in the XBRL file.
2. When you have finished selecting XBRL elements, click **Next**.

Selecting XBRL contexts

Use the **Select XBRL Contexts to Import** page of the wizard to select an XBRL context type and individual contexts.

The XBRL data imported to an Analytics table must all be associated with the same context type (instant, period, or forever). Importing multiple context types into a single Analytics table is not supported.

> **Note**
>
> If you are working with a complex XBRL file you may need to, or may find it easier to, define more than one Analytics table for the various contexts in the file, and then define relations between the tables using the **Relate Tables** command.

To select XBRL contexts:

1. Select the **Context Type** you want to add to the Analytics table.

   All contexts in the XBRL file with a matching context type are displayed in the **Available Contexts** pane.

2. Select one or more contexts in the **Available Contexts** pane and click the right arrow button to move them to the **Selected Contexts** pane.

   You can also click **Add All** to add all available contexts. When you add one or more contexts to the **Selected Contexts** pane, the **Import Size** panel displays the number of individual records that will be imported to the Analytics table.

   When you select an individual context in either pane, the **Context Properties** area displays entity and period information. You can click **View Scenario** if any scenario information is associated with the context.

3. When you have added all the contexts required for the Analytics table, click **Next**.

About XBRL files

eXtensible Business Reporting Language (XBRL) is an XML-based standard for defining and exchanging business and financial data. XBRL can be used to submit financial data to regulators, and to transfer data between companies or between systems within a company. Analytics supports the current XBRL 2.1 standard.

An XBRL instance document is an XML file that uses XBRL elements and conforms to the XBRL standard. XBRL elements are also referred to as **items**. Examples of XBRL instance document types include balance sheets, general ledgers, and financial statements.

An XBRL document is a valid XML file that often uses the standard XML file extension (.xml). The root element `<xbrl>`, the first element in the file, identifies the file as an XBRL file.

About XBRL contexts

An XBRL document includes elements and data that define one or more **contexts**. Contexts categorize the main body of data contained in the file. Each element or item in the file must reference a particular context. Contexts specify the following:

- **Entity** – The company, business unit, etc., to which the data pertains.
- **Period** – The period for which the data is valid. There are three possible periods:
  - **Instant** – The data is valid for a specific date or datetime. For example, the value of a bank balance on 01 Jan 2012 at 9:00 a.m.
  - **Period** – The data is valid for a specific date range or datetime range. For example, financial transactions occurring between 01 Jan 2012 and 31 Dec 2012.
  - **Forever** – The data is not date or time dependent. For example, an account number is valid for an indefinite or undefined period of time.
- **Scenario** (optional) – Additional contextual information about the associated elements. For example, whether the values contained in the elements are actual, projected, or budgeted.

An XBRL document can contain multiple contexts. For example, a document may contain one context for the period 01 Jan 2011 to 31 Dec 2011, and another for the period 01 Jan 2012 to 31 Dec 2012. In the **Data Definition Wizard**, each context is available as a separate block of data that you can choose to import or not.

Defining Analytics Server database profile data

If your company has one or more Windows Analytics server products installed and configured, and you have created a server profile and database profile to connect to the data source, you can create Analytics tables that directly access data from Oracle, SQL Server, or IBM DB2 databases.

Each time you open the Analytics table the most current data is retrieved from the database by rerunning the query used to create the Analytics table. The WHERE and ORDER clauses can be modified after the Analytics table is created.

Analytics Server table limitations

The following limitations apply when you define an Analytics table using a database profile:

- If you use an Analytics Server table in an Analytics join it must be the primary table.
- Only one Analytics Server table can be used in Analytics when relating tables and it must be the parent table.
- You cannot index Analytics Server tables. If you need to order the data you must specify an ORDER clause when you define the table, or modify the table properties in Analytics to add an ORDER clause.
- The **Quick Sort Ascending** and **Quick Sort Descending** options are not enabled in the View window for Analytics Server tables.

Steps

1. Select **File > New > Table**.
2. In the **Select Platform for Data Source** page, select **ACL Server** and select the Windows server profile to use from the drop-down list, and click **Next**.
3. In the **Select ACL Server Data Source** page, select **Database Profile** and select the database profile to use from the drop-down list, and click **Next**.
4. In the **Select Database/Schema** page, select the schema (Oracle) or database (SQL Server and IBM DB2) to access from the **Schema** drop-down list and click **Next**.
5. In the **Select Tables** page, select the database tables, views, and synonyms/aliases to add to your query by selecting the item in the **Available Tables** list and clicking the right-arrow button. You can select up to five tables, but if you select more than one table, each additional table you select must be related to a previously selected table. Selecting multiple tables, particularly tables with large numbers of records, will typically result in longer wait times before data is displayed in Analytics.

   When you select more than one table, Analytics displays the **Identify Relationship** dialog box which you must use to identify the field in the table you are adding that relates to a table that

has already been selected.

6. Click **Next** when you have selected the required table(s).
7. Optional. In the **Condition Clause** page, select the **Edit the CONDITION clause** checkbox to enable the text box for editing, enter the condition in the text box using SQL syntax, and click **Next**. A value is only displayed in the **Condition Clause** text box if two or more related tables are included in the query. The condition clause specifies how the tables are related.
8. In the **Select Columns** page, select the column(s) to add in the **Available Columns** list and click the right-arrow to add it to the **Selected Columns** list, or click **Add All** to add all columns to the **Selected Columns** list. If you selected multiple tables, you must select at least one column from each table by selecting each of the tables in the **Select Source Table** drop-down list and selecting the required columns. When you have finished selecting all of the columns you want to include, click **Next**.
9. Optional. In the **Where and Order Clauses** page, enter a WHERE statement if you want to limit the results returned from the database and/or an ORDER statement if you want to specify that the results should be sorted by one or more columns and click **Next**. For both types of statements, you must enter the statement using SQL syntax without the command keyword. For example, `WHERE Value > 1000.00` must be entered as `Value > 1000.00`
10. In the **Final** page, verify the settings for the new Analytics table and click **Finish**.
11. Enter a name for the Analytics table you are adding to your project and click **OK**.

Defining External Definition files

External definition files store the record and field layout information for a data file in an external file. Analytics can parse the external definition file for the necessary field information and create an Analytics table layout. The Analytics table layout is then used to read the data file directly, without creating an Analytics data file (.fil).

Define an AS400 FDF file

You can define Analytics tables from AS/400 file definition files. This type of file stores field formatting information in an external file, separate from the data file. Analytics uses the formatting information to create an Analytics table layout and reads data directly from the data file.

1. Select **File > New > Table**.
2. If the **Select Platform for Data Source** page is displayed, select **Local** and click **Next**.
3. In the **Select Local Data Source** page, select **Other** and click **Next**.
4. In the **Select External Definition Source** page, select **AS400 FDF** and click **Next**.
5. In the **Select AS/400 FDF file to convert** dialog box, locate and select the external definition file to use and click **Open**.
6. In the **AS/400 Conversion** page, click **Next**.
7. In the **Final** page, verify the settings for the new Analytics table and click **Finish**.
8. Enter a name for the Analytics table you are adding to your project and click **OK**.
9. If the associated data file (.dat) is not found, you are prompted for the file location. In the **Locate Data File <*filename*>** dialog box, locate and select the required file and click **Open**.

Define a Cobol or PL/1 file

You can define Analytics tables from Cobol copybook files or PL/1 files. Both types of files store field formatting information in an external file, separate from the data file. Analytics uses the formatting information to create an Analytics table layout and reads data directly from the data file.

1. Select **File > New > Table**.

   The first page displayed in the **Data Definition Wizard** depends on your configuration. If integration with Analytics Server is enabled the **Select Platform for Data Source** page is displayed, otherwise the **Select Local Data Source** page is displayed.

2. If the **Select Platform for Data Source** page is displayed, select **Local** and click **Next**.
3. In the **Select Local Data Source** page, select **Other** and click **Next**.
4. In the **Select External Definition Source** page, select **Cobol** or **PL/1**, as appropriate, and click **Next**.

5. In the **Select External Definition(s)** page, complete the following steps:
   a. Do one of the following:
      - Select an existing definition file or data set from the **External Definition** drop-down list.
      - Click **Browse**, locate the definition file or data set in the **Select File to Convert** dialog box, and click **Open**.
   b. Complete one of the following steps to select either a single definition or multiple definitions:
      - Add an individual definition to the **Selected Definitions** list by selecting the definition in the **Available Definitions** list and clicking the right-arrow button, or by double-clicking the definition in the **Available Definitions** list.
      - Add multiple definitions by selecting the **Do you want to concatenate multiple selected file definitions?** checkbox, and then select each definition to add and click the right-arrow button, or click **Add all** to add all of the definitions listed in the **Available Definitions** list.
   c. Click **Next**.
6. In the **Select Conversion Properties** page, select either or both of the following properties, as necessary, and click **Next**.
   - **Remove leading file indicator from field names** - If a prefix is specified before each field name, it will be removed from the field name added to the Analytics table layout. For example, `Test-Field1` would have the prefix `Test-` removed.

     If this checkbox is not selected, any prefix values identified will be included in the field name with the hyphen converted to an underscore (i.e. `Test-Field1` will be added as `Test_Field1`).
   - **IBM Variable Length** - If the data file being processed is an IBM variable length file, the record length is not specified in the Analytics table layout.
7. In the **Final** page, verify the settings for the new Analytics table and click **Finish**.
8. Enter a name for the Analytics table you are adding to your project and click **OK**.
9. If the associated data file is not found, you will be prompted for the file location. Complete the following steps to locate the file:
   a. If the **Select file location** dialog box is displayed select **Client** to select a file that is accessible from your computer, or select **Server** and select the server profile to use to access a server file, and click **OK**.
   b. In the **Select file** dialog box, locate and select the data file and click **Open**.

Defining Analytics tables manually

If Analytics is unable to identify the file format of a file-based data source it selects the **Other file format** option on the **File Format** page in the **Data Definition Wizard**. You can also select this option if you want to define a file manually. When this option is selected, the **Data Definition Wizard** walks you through the steps needed to define the file, but you need to provide all of the information required for Analytics to create a table layout from the file.

To define an Analytics table manually:

1. Select **File > New > Table**.

   The first page displayed in the **Data Definition Wizard** depends on your configuration. If integration with Analytics Server is enabled the **Select Platform for Data Source** page is displayed, otherwise the **Select Local Data Source** page is displayed.

2. Complete one of the following steps to select the location of the file:
   - If the **Select Platform for Data Source** page is displayed and you want to use Analytics to define the file, select **Local** and click **Next**. In the **Select Local Data Source** page select **File** and click **Next**.
   - If the **Select Platform for Data Source** page is displayed and you want to use an Analytics Server to define the file, select **ACL Server** and select the server profile from the drop-down list, and then click **Next**. In the **Select ACL Server Data Source** page select **Flat Files** and click **Next**.
   - If the **Select Local Data Source** page is displayed select **File** and click **Next**.
3. In the **Select File to Define** dialog box, locate and select the file you want to create the Analytics table from and click **Open**.
4. In the **Character Set** page, verify that the correct character set option has been selected and click **Next**.
5. In the **File Format** page, the **Other file format** option will be selected if the **Data Definition Wizard** cannot identify the file as a more specific type. This means that the Analytics table must be manually defined. You can also reclassify a file that has been assigned to a different file type if you want to define the Analytics table manually. Click **Next** to proceed to the next page.
6. In the **File Properties** page, complete the following steps:
   a. Verify the file type identified by the **Data Definition Wizard**. If you want to change the type, select the file definition type from the following options:
      - **Fixed Length** – Select this option if all the records in the file are an equal length, and each field is located in the same location in every record.
      - **Variable Length** – Select this option if records in the file vary in length. The next page of the wizard allows you to further classify the file type as a file that contains a single record type that varies in length, a Print Image (Report) file, or a file that contains more than one record type.
      - **Skip to Finish** – Select this option to move to the **Final** page of the **Data Definition Wizard** without defining fields in the Analytics table layout. You should select this option if you want to define the table layout manually in the **Table Layout** dialog box, or if you are

unable to define a file using the **Data Definition Wizard**.

b. Enter a value greater than 0 in the **Bytes to Skip** text box to skip the specified number of bytes from the beginning of the file. For example, if the first 300 bytes only contains header information, you can enter 300 to omit this section of the file from the table definition.
c. Modify the value in the **Record Length** text box to increase or decrease the record length identified by the **Data Definition Wizard**. The record length refers to the length of each record in fixed length files, or the length of the longest record in variable length files. If the values in a field are misaligned to the right, the record length value likely needs to be increased. If the values in a field are misaligned to the left, the record length value likely needs to be decreased.
d. Select the **Hex** checkbox to view the data in hexadecimal format. This option is useful if you are working with unprintable characters or compressed data, such as packed numeric data originating from an IBM mainframe computer.
e. Click **Next**.

7. If you selected **Fixed Length** or **Variable Length** in the previous step, select one of the following options in the **File Type** page and click **Next**.
   - **Data File (Single Record Type)** – Select this option if each field in the record has a fixed start and end point, and each record in the file is the same length.
   - **Print Image File (Report File)** – Select this option if the data file is an electronic version of a printed report that includes consistent formatting. This type of data file has detail records that contain the information being reported on, and often include header and/or footer records that include additional information, such as customer details and totals.

     > **Note**
     > Before you attempt to define a print image file using this option, you should try selecting the **Print Image (Report) File** option in the **File Format** page of the **Data Definition Wizard** (step 5 above), which uses a more straight-forward process for defining print image files.

   - **Multiple Record Type File** – Select this option if the data file includes more than one type of record, but is not formatted as a report.
   - **Skip Field Identification** – Select this option to move to the **Final** page of the **Data Definition Wizard** without defining fields in the Analytics table layout.
8. If you selected **Data File (Single Record Type)** in the previous step, complete the following steps:
   a. In the **Identify Fields** page, complete any of the following actions to modify the fields identified in the record, and click **Next**.
      - Delete an existing field separator by clicking on the field separator line you want to remove.
      - Move an existing field separator by clicking and dragging the field separator line to the new location.
      - Create a new field separator by clicking the grid in the position where you want to add the field separator.

b. In the **Edit Field Properties** page, complete the following steps:
   a. In the preview table in the bottom half of the page, click the column heading of the field you want to edit properties for.
   b. If you do not want the field to be included in the Analytics table layout, select **Ignore this field**.
   c. In **Name**, keep the name assigned by Analytics for the field in the table layout, or enter a different name.
   d. In **Column Title**, enter the column title to display in the default Analytics view. If a column title is not specified the **Name** value is used.
   e. Select the appropriate data type for the field from the **Type** drop-down list. For information about the supported data types in Analytics, see "Data types in Analytics" on page 739.
   f. If the selected data type is Numeric, you can specify the number of decimal places in the **Decimal** text box. The **Decimal** text box appears automatically when you select the corresponding data type.
   g. If the selected data type is Datetime, specify the format that matches the data in the **Input Format** text box. The **Input Format** text box appears automatically when you select the corresponding data type. For more information about date and time formats, see "Formats of date and time source data" on the facing page.
   h. When you have edited all of the fields you want to modify, click **Next**.

9. If you selected **Print Image File (Report File)** or **Multiple Record Type File** in step 7, complete the following steps:
   a. In the **Record Field Introduction** page, the **Data Definition Wizard** displays any records it has identified automatically in the data file. Select **Add/Edit Record Types** if you want to create new record types or modify the existing record types identified by the wizard, and click **Next** to continue.
   b. In the **Identify Record/Line Types** page, complete any of the following steps to specify the record types in the data file:
      - To add a new record type, enter a name for the record type in the **Select the type to define** drop-down list, select the text that defines the record in the preview table using the **Include** or **Exclude** buttons to include or exclude text from the record as appropriate, and select the **Heading line** checkbox if the record is a header record.
      - To edit an existing record type, select the record type you want to edit in the **Select the type to define** drop-down list, modify the unique text that defines the record by selecting text in the preview table using the **Include** or **Exclude** buttons to include or exclude text from the record as appropriate, remove previously selected text by choosing the entry in the list and clicking **Delete**, or select the **Heading line** checkbox if the record is a header record.
   c. In the **Identify Fields** page, complete the steps outlined above in step 8a.
   d. In the **Edit Field Properties** page, complete the steps outlined above in step 8b.
10. In the **Final** page, verify the settings for the new Analytics table and click **Finish**.
11. Enter a name for the Analytics table you are adding to the project, or keep the default name, and click **OK**.

Formats of date and time source data

When you define an Analytics table, the source format (input format) of date, datetime, or time data may be auto-recognized by Analytics. For example, Analytics auto-recognizes dates that use the format YYYYMMDD. If the source format is not auto-recognized, you must manually specify the format.

Source format versus display format

Specifying the format of source datetime data is not the same as specifying how Analytics displays datetime data. The source format controls how Analytics reads datetime data in the source file. There must be a one-to-one correspondence between the source format characters that you specify and the actual format of the source data.

Once source datetime data has been successfully defined in Analytics, you can choose to display it in a variety of different formats. Choosing to display datetime data in different formats does not affect the underlying source format.

For more information about displaying datetime data, see "Date and Time options" on page 133.

Manually specifying the source format

You can manually specify the datetime source format while defining a table in the **Data Definition Wizard**. Or you can specify the format later in the **Table Layout** dialog box in Analytics.

With the exception of certain separator characters, the format you specify must exactly match the format of the source data for the source data to appear correctly in Analytics.

Guidelines for specifying separator characters in datetime formats

Source datetime data often includes separator characters:

- Characters such as slashes (/) between the day, month, and year components of dates
- Characters such as colons (:) between the hour, minute, and second components of times
- A space, or a character such as 'T', between the date and time portions of datetime values
- A character such as 'T' or a decimal point before standalone time values
- For local times with a time zone indicator, a plus (+) or minus (-) sign before the UTC offset

Analytics auto-recognizes some, but not all, of these separator characters in the source data.

Follow the guidelines below when specifying separator characters in datetime formats. Omitting or incorrectly specifying separator characters can prevent datetime data from displaying, or from displaying correctly.

> **Note**
> Specifying particular separator characters in the datetime format can be **required**, **optional**, or **disallowed**, depending on the function of the character.

Function of separator character	Specify in format?	For this source data:	Specify this format:
Separates day, month, and year components of dates	Required	31/12/2014	DD/MM/YYYY
Separates hour, minute, and seconds components of times	Optional	23:59:59	hh:mm:ss hhmmss
Separates the date and time portions of datetime values (single space)	Optional	31/12/2014 23:59:59	DD/MM/YYYY hh:mm:ss DD/MM/YYYYhh:mm:ss DD/MM/YYYY hhmmss DD/MM/YYYYhhmmss
Separates the date and time portions of datetime values ('T' or 't')	Disallowed	31/12/2014T235959	DD/MM/YYYY hhmmss DD/MM/YYYYhhmmss
Prefaces standalone time values ('T' or 't')	Disallowed	T235959	hhmmss
Separates the date and time portions of datetime values that use a Numeric data type (decimal point)	Optional	31122014.235959	DDMMYYYY.hhmmss DDMMYYYYhhmmss
Prefaces standalone time values that use a Numeric data type (decimal point)	Optional	.235959	.hhmmss hhmmss
Prefaces a UTC offset (plus or minus sign)	Required	T235959-0500	hhmmss-hhmm hhmmss+hhmm

Date and time separators

In order for Analytics to read datetime values from source data, the date and time components in the source data must be separated by a space or a separator character. For example:

- 2014/12/31 23:59:59
- 20141231.235959

For datetime values that use a Datetime data type, or a Character data type, Analytics recognizes the following separators:

- <*date*> <*time*> (single space)
- <*date*>T<*time*> (uppercase 'T')
- <*date*>t<*time*> (lowercase 't')

For datetime values that use a Numeric data type, Analytics recognizes only the following separator:

- <*date*>.<*time*> (decimal point)

> **Note**
> Analytics can read datetime values that use a Datetime or Character data type and have a period as a separator – <*date*>.<*time*>. However, the period separator is not officially supported because in some situations results can be unreliable.

Standalone time data

In order for Analytics to read standalone time values from source data – for example, 23:59:59 – the time value in the source data must be prefaced by a space or a separator character, or the time components must be separated by colons. For example:

- 23:59:59
- .235959

For time values that use a Datetime data type, or a Character data type, Analytics recognizes the following separators:

- _<*time*> (single space)
- T<*time*> (uppercase 'T')
- t<*time*> (lowercase 't')
- <*hh*>:<*mm*>:<*ss*> (colons)

For time values that use a Numeric data type, Analytics recognizes only the following separator:

- .<*time*> (decimal point)

> **Note**
> Analytics can read time values that use a Datetime or Character data type and have a period as a separator – .<*time*>. However, the period separator is not officially supported because in some situations results can be unreliable.

Date formats

There are many date formatting conventions in use. In the **Data Definition Wizard**, and the **Table Layout** dialog box, you can select from among several common date formats. If necessary, you can

modify or create a date format to match the source data.

Date formats apply to date data, or to the date portion of datetime data. Several common date formats are shown below:

Common date format	Type	Example using December 31, 2014
YYYY-MM-DD	ISO	2014-12-31
MM/DD/YYYY	American	12/31/2014
DD/MM/YYYY DD.MM.YYYY DD-MM-YYYY	European	31/12/2014 31.12.2014 31-12-2014
YYDDD	Julian	14365

Day, month, and year characters

When you specify a date format, you are specifying which components in the source data represent the day, the month, and the year. In Analytics, the format characters shown below are used to represent the day, month, and year components of a date.

> **Note**
> These characters are the default, and they can be changed in the **Options** dialog box.

If separators such as the slash symbol (/) exist in the source data, you need to insert the same symbol in the same relative position in the date format. Otherwise, Analytics will not interpret the date correctly.

Format characters	Date component
DD	Day (1 - 31)
DDD	Julian day (1 - 366)
MM	Month (1 - 12)
MMM	Month name (Jan - Dec)
YY	Short year format (00 - 99)
YYYY	Long year format (1900 - 9999)

Examples of specifying the date format for source data

Analytics date format	Source data
YYYY-MM-DD	2014-12-31
YYYYMMDD	20141231
MM/DD/YYYY	12/31/2014
MM/DD/YY	12/31/14
DD/MM/YYYY	31/12/2014
YYDDD	14365
MMM DD, YYYY	Dec 31, 2014
DD MMM YYYY	31 Dec 2014

Time formats

Analytics supports the most common time formatting convention – **hh:mm:ss** – and some minor variations of this format. In the **Data Definition Wizard**, and the **Table Layout** dialog box, you can select from among several common time formats. If necessary, you can modify or create a time format to match the source data.

Time formats apply to time data, or to the time portion of datetime data.

Hour, minute, and second characters

When you specify a time format, you are specifying which components in the source data represent the hour, the minutes, and the seconds, and if they are present, the AM/PM indicator, and the UTC offset indicator. In Analytics, the format characters shown below are used to represent the various components of time data.

> **Note**
> The hour, minute, and second characters shown below are the default, and they can be changed in the **Options** dialog box.

Format characters	Time component
hh	Hour (00 - 23)
mm	Minute (00 - 59)
ss	Second (00 - 59)
: (colon)	time component separator
A or P	AM/PM indicator (A and P)
AM or PM	AM/PM indicator (AM and PM)
+ or -	UTC offset indicator (+ and -)

Examples of specifying the time format for source data

Analytics time format	Source data
hh:mm	23:59
hh:mm A	11:59 P
hhmm PM	1159 PM
hh:mm:ss	23:59:59
hh:mm:ss P	11:59:59 P
hhmmss AM	115959 PM
hh:mm:ss+hh:mm	23:59:59-05:00

Importing data using the Data Access window

The Data Access window is a component of Analytics that contains data connectors, which you can use to import data to Analytics from a wide range of sources. The Data Access window also contains features for precisely shaping the set of data you are importing.

For a complete list of the data sources you can connect to using the Data Access window, see "Data sources you can access with Analytics" on page 227.

> **Note**
>
> You can also import data using the Data Definition Wizard. For more information, see "Defining and importing data using the Data Definition Wizard" on page 232.
>
> When connecting to any data source, or importing from any data source, Analytics is strictly read-only. For more information, see "Data access by Analytics is read-only" on page 228.

What are data connectors?

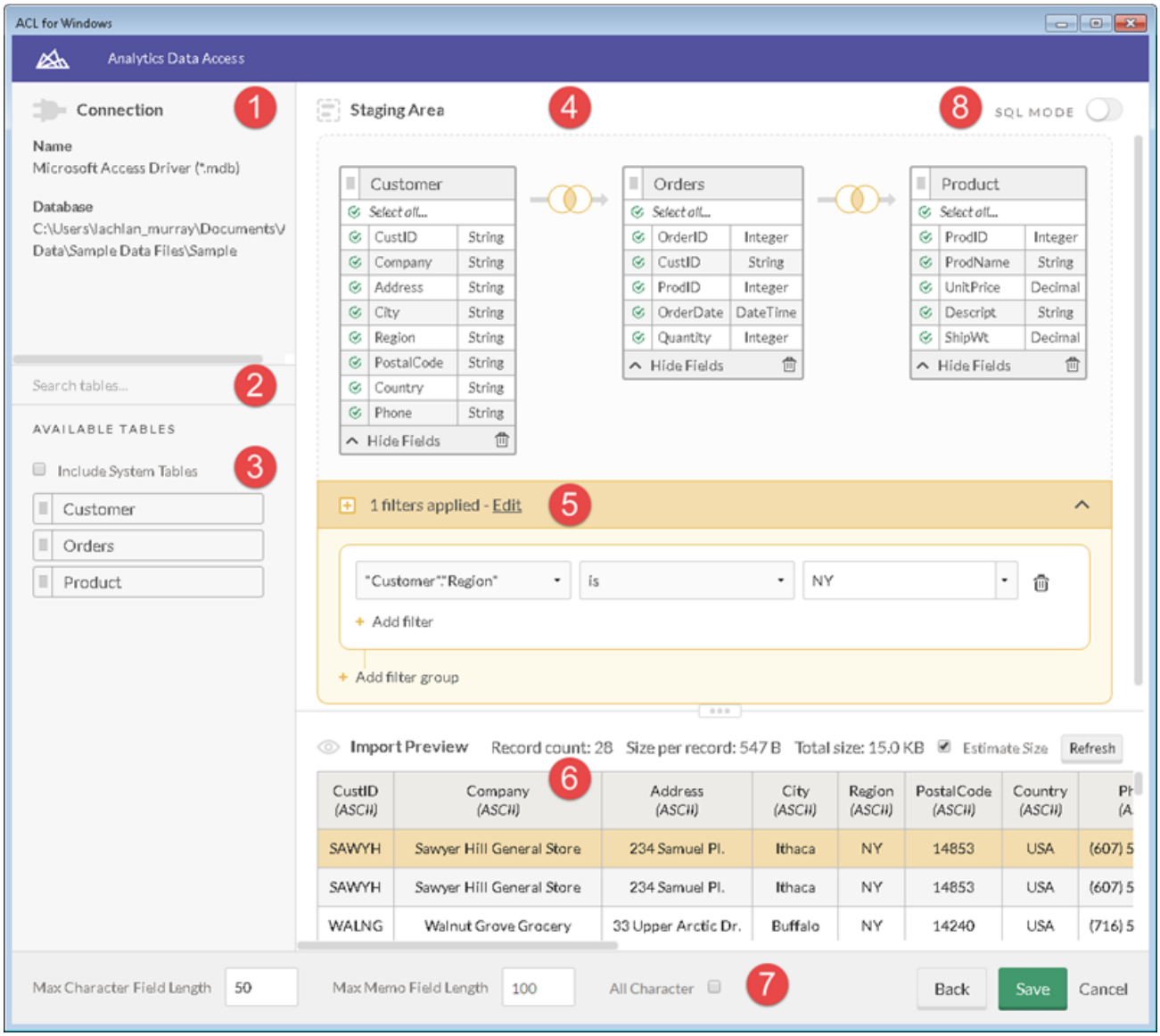
Data connectors are ODBC drivers that provide an interface between Analytics and ODBC-compliant databases and file formats, such as Microsoft SQL Server, Oracle, Salesforce, and Microsoft Excel.

ODBC stands for Open Database Connectivity, a Microsoft standard that use SQL, or Structured Query Language, to allow an application to access data in an external database or file.

> **Note**
>
> The Data Access window is an import-only tool. You can edit the SQL import statement used to access data in an external database or file. Editing the SQL to write to the data source is not supported.

Overview of the Data Access window

The features contained in the Data Access window are explained below.



Number	Feature	Description
1	**Connection**	Displays information about the currently active data connection. Includes the name of the connection, and the name of the database, or the location of the file, containing the source data.
2	**Search tables**	A search box for progressively filtering the list of available tables in the source data. As you enter characters in the search box, the **Available Tables** list is filtered so that it contains only tables names with a matching string of characters.
3	**Available Tables**	The tables in the source data that are available for import.

Number	Feature	Description
		The first 200 tables in the source data are displayed. If additional tables exist, you can click a link to display them in blocks of up to 500 tables at a time.
4	**Staging Area**	The area in the Data Access window that contains the table or tables you have selected for import. The **Staging Area** is also the location where you perform joins between tables, and select which fields in a table are imported.
5	**Filter panel**	A panel for building simple or compound filters that precisely specify which records from a data set are imported.
6	**Import Preview**	A preview of the data exactly as it will be imported into Analytics. As you work with the data by joining tables, omitting fields, and creating filters, you can refresh the preview to see the effect of your changes. The **Estimate Size** option displays an estimate of the number of records in the import, and the size of the Analytics data file (.fil) that will be created.
7	**Field configuration**	Three options that allow you to: ◦ adjust the maximum field lengths for imported character or memo fields ◦ import all fields as a Character data type
8	**SQL Mode**	A text editor that allows you to directly edit the SQL import statement. Users with knowledge of SQL can control aspects of the data import not available through the user interface.

Managing data connections

In the Data Access window, you can rename, copy, or delete a connection created using an Analytics data connector.

You can also clear the list of cached table names stored by a connector.

Rename, copy, or delete a connection

> **Note**
> You cannot use the Data Access window to rename, copy, or delete a DSN connection created using a Windows ODBC driver. These connections can be maintained in Windows.

1. From the Analytics main menu, select **Import > Database and application**.
2. In the **Existing Connections** tab, under **ACL Connectors** or **ACL DSN Connectors (Bundled)**, hover over the connection that you want to maintain, and click the ellipsis icon ⋯.

3. Select one of the following as available:
   - **Create a copy**
   - **Rename connection**
   - **Delete connection**
4. Follow the on-screen prompts to complete the task.

Clear cached table names

When you successfully connect to a data source using the Data Access window, the connector that you use caches the names of the first 200 tables in the data source. This list of table names speeds up subsequent connections to the same data source.

If a schema change in the data source adds, deletes, or renames any tables, you need to manually clear the table name cache. A mismatch between the table name cache and the data source schema causes a connection error.

After clearing the cache and successfully connecting to the data source, the cache is refreshed with the updated list of table names.

1. From the Analytics main menu, select **Import > Database and application**.
2. In the **Existing Connections** tab, under **ACL Connectors**, **ACL DSN Connectors (Bundled)**, or **Windows DSN Connectors**, hover over the connection that you want to maintain, and click the ellipsis icon .
3. Select **Clear cache**.

   The table name cache is cleared.

Data Access log files

Two log files record activities in the Data Access window, and can be used for troubleshooting if a data connection fails:

- **ServerDataAccess.log** – records all Data Access window activities and errors prior to importing the data

  Location: `C:\Users\<user account>\AppData\Local\ACL\ACL for Windows\Data Access\ServerDataAccess.log`

  > **Note**
  > The "Server" in `ServerDataAccess.log` refers to the data access component of Analytics running locally on the computer where Analytics is installed.

- **DataAccess.log** – records information about the import operation and the Analytics project that you are importing data to

  Location: `..\<Analytics project folder>\DataAccess.log`

Working with the Data Access window

You can import data and create an Analytics table by using the Data Access window to connect to source data in either a database or a file.

> **Note**
> In the event of a failed connection, two log files can assist with troubleshooting. For more information, see "Data Access log files" on the previous page.

Before you connect to a database or a cloud data service

Certain requirements or prerequisites exist when using the Data Access window to connect to a database or a cloud data service:

- **An account** – You must have a database or data service account. The account may be a personal account, or an account associated with a role or with your company. This account is completely separate from your Analytics account.
- **Credentials** – You need valid credentials for the database or data service account. The type of account credentials required depend on the data source. You may need to enter a user name and password, an access token, or some other form of credential.
- **Prior configuration** – For data sources such as Salesforce and Concur, prior configuration is required within the cloud data service before you can connect using the Data Access window.

> **Note**
> Configuration of connection prerequisites within a cloud data service is typically performed by the person in your company responsible for administering the service – for example, your company's Salesforce administrator, or Concur administrator.
>
> For connection issues that originate with the cloud data service you need to contact your company's administrator for that service rather than Support.

Connect to the database or the file

From the Analytics main menu, select **Import > Database and application**.

> **Note**
> In the **Data Definition Wizard**, you can also select **Local > Database and Application**.

Use an existing connection

1. In the **Existing Connections** tab, select the connection for the data source you want to connect to.

   > **Tip**
   > You can filter the list of available connections by entering a search string in the **Filter connections** box. Connections are listed alphabetically.

   For some types of connections, you are immediately connected to the source data.

   The existing connections are organized under **ACL Connectors**, **ACL DSN Connectors (Bundled)**, **Windows DSN Connectors**, and **Other Connectors**.

   If you do not have any existing connections, the section or sections do not appear.
2. If you are not connected immediately, do one of the following:
   - If the **Data Connection Settings** panel opens, click **Connect** at the bottom of the panel (you may need to scroll).

     If required, you can update the connection settings before you click **Connect**.
   - If a file selection dialog box opens, navigate to the appropriate file and select it.

Create a new connection

1. Select the **New Connections** tab.
2. Select the connector for the data source you want to connect to.

   > **Tip**
   > You can filter the list of available connectors by entering a search string in the **Filter connections** box. Connectors are listed alphabetically.

   The available connectors are organized under **ACL Connectors**, **ACL DSN Connectors (Bundled)**, **Windows DSN Connectors**, and **Other Connectors**.
3. Do one of the following:
   - If the **Data Connection Settings** panel opens, enter the connection settings, and click **Save and Connect** at the bottom of the panel (you may need to scroll).

     You can accept the default **Connection Name**, or enter a new one.
   - For connectors in the **ACL DSN Connectors (Bundled)** section, the **DSN Configuration** dialog box opens.

i. Click the **Show Required** tab.
ii. Provide values for the required fields, if any, and click **Test Connection**.

The log in page for your data source appears.

iii. Provide the connection details and authenticate your login.
iv. Click **OK** in the connection successful dialog box that appears.
v. In the **DSN Configuration** dialog box, click **OK**.

- If a file selection dialog box opens, navigate to the appropriate file and select it.

> **Note**
> Successful connections made with an Analytics connector are automatically saved to the **Existing Connections** tab.
> Connections made with Windows connectors are preserved for the current data import session only.

Manage data connections

You can rename, copy, or delete a connection created using an ACL connector. For more information, see "Managing data connections" on page 355.

Add one or more tables to the Staging Area

1. In the Data Access window, select the appropriate database schema from the **Schema** drop-down list, if required.

   > **Note**
   > Some data sources may not have a schema, or may have only one schema.

2. Optional. In the **Connection** panel, filter the list of available tables by entering a search string in the **Search tables** box.

   Matches for a literal search string (no wildcards) can appear anywhere in a table name. The search is not case-sensitive.

   You can also use one or more wildcard characters in the search string.

   Show me more

Wildcard	Scope	Example	Matches
`*` `%`	0 or more characters	`invoice-j*`	◦ Invoice-January ◦ Invoice-June ◦ Invoice-July
		`%june`	◦ Invoice-June ◦ PO-June

Wildcard	Scope	Example	Matches
		`%invoice%`	◦ Invoice-June ◦ June-Invoice
`?` `_`	1 character	`invoice??`	◦ invoice-1 ◦ invoice-2
		`j_n`	◦ Jan ◦ Jun

> **Note**
> If you use a wildcard character in the search string, the length of matching strings, and the first and last characters, are evaluated more strictly:
> - `j_n` matches only three-character strings that begin with **j** and end with **n**
> - `j*n` matches strings of any length, but the strings must begin with **j** and end with **n**
> - by contrast, `jan` matches strings of any length, and **jan** can appear anywhere in the string
>
> This matching behavior is the result of Analytics interpreting literal search strings (no wildcards) in the following manner: `jan` = `*jan*`

3. Optional. Scroll to the bottom of the list of tables and click **Show remaining # tables**.

   Analytics displays the first 200 tables in the data source. If additional tables exist, you can click the **Show remaining** link to display them in blocks of up to 500 tables at a time.

   > **Note**
   > The **Search tables** box must be empty for the link to appear.

4. Under **Available Tables**, click a table name to add the table to the **Staging Area**.

   Tables are listed alphabetically. You can add up to ten tables to the staging area if you intend to join the tables. The SAP connector is currently limited to two tables.

   > **Note**
   > You cannot import multiple tables individually with one import operation. The tables must be joined to be imported together.

5. Optional. Select **Include System Tables** if you want to add any system tables to the list of available tables.

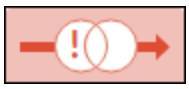
   > **Note**
   > This option does not apply to some data sources.

Join tables

If you added more than one table to the **Staging Area**, you need to join the tables.

For detailed information about joining tables, see "Joining tables in the Data Access window" on page 368. For information about joining Apache Drill tables, see "Joining tables from Apache Drill data sources" on page 371.

1. In the **Staging Area**, click the join icon to access the **Join Settings**.



2. Click the type of join you want:
   - **Inner**
   - **Outer**
   - **Left**
   - **Right**

> **Note**
> Some data connectors, including the Microsoft Excel and Microsoft Access connectors, do not support the **Outer** join type.

3. Select the common key fields by doing the following:
   a. Under **Left Column**, select the left table key field.
   b. Under **Right Column**, select the right table key field.

> **Tip**
> You can filter the list of available fields by entering a search string in the **Left Column** or **Right Column** boxes. Fields are listed alphabetically.

4. Optional. Click **+ Add key** if you need to add an additional key field.
5. Click **Apply** to save the join settings.
6. Create join settings for each additional table you are joining.
7. Optional. In the **Import Preview** panel, click **Refresh** to see a preview of the joined tables.

Select the fields to import

By default, all fields in a table are imported unless you deselect specific fields.

1. If you want to omit one or more fields from the import, click the **Show Fields** dropdown list on the table.
2. Click a field name to deselect it.

> **Tip**
> If you want to deselect most of the fields, click the **Select all** toggle to deselect all fields, and then re-select the fields you want.

3. In the **Import Preview** panel, click **Refresh** to review the fields included in the import.

4. Optional. To import one or more fields as cryptographic hash values:
   a. In the **Import Preview** panel, select the **Hash** checkbox at the top of the columns you want to transform.
   b. At the bottom of the screen, in the **Salt** field, enter an alphanumeric string to use in the hashing function.

      The salt value is limited to 128 characters. Do not use any of the following characters: ( ) "

      If you do not provide a salt value, Analytics generates a random string. Click **Refresh** to view the generated string in the **Salt** field.

   Hash values are one-way transformations and cannot be decoded after you import the fields.

> **Note**
>
> Even though you cannot read the raw values of hashed data, it is still useful when combining or analyzing data. If you want to compare values that are hashed by ACCESSDATA during import with values that are hashed using ACLScript's HASH( ) function, you must convert any numeric or datetime Analytics fields to character values and trim any spaces before hashing the data.
>
> Datetime fields must use the following formats:
>
> - **Datetime** - "YYYY-MM-DD hh:mm:ss"
> - **Date** - "YYYY-MM-DD"
> - **Time** - "hh:mm:ss"

Filter data

By default, all records in a table are imported unless you create one or more filters to omit specific records.

> **Note**
>
> If you use both of the filtering options explained below, conditional filtering is applied first, and then the specified number of records is applied to the results of the conditional filtering.

Specify the number of records to import

You can specify that only a certain number of records, starting from the top of the table, are imported.

1. In **Select first *n* records**, enter the number of records, starting from the top of the table, that you want to import.
2. Optional. In the **Import Preview** panel, click **Refresh** to see the records included in the import.

> **Tip**
>
> To reset the import to all records in the table, enter n in **Select first *n* records**.

Create a conditional filter

1. If you want to conditionally omit records from the import, click **Add filters to limit results** to create a filter.
2. From the **Field** list, select the field you want to use for filtering.

   > **Tip**
   > You can filter the list of available fields by entering a search string in the **Field** box. Fields are listed alphabetically.

   > **Note**
   > If you have joined tables, you can select a field from any of the joined tables.

3. From the **Condition** list, select a conditional operator such as **is**, **equals**, or **greater than**.

   The **in** operator allows you to specify multiple test values. For more information, see "Using the "in" conditional operator" on the next page.

4. In the third field, enter the value to test against.

   > **Note**
   > If you are filtering using a Logical field, the test value may need to be one of the following, depending on the data source:
   > - `'true'` or `'false'` (including the single quotation marks)
   > - `1` or `0` (1= true, 0 = false)
   >
   > If filtering using one of the actual values in the field returns an error, try one of the values above.

5. Optional. To add another filter, do the following:
   a. Click **Add filter**.
   b. Select **AND** or **OR**, depending on how you want the filters to be combined.
   c. Repeat steps 2 to 4 to create the filter.

   You can continue to add filters to specify the exact set of data you want to import.

   > **Note**
   > You cannot mix Boolean operators when you combine multiple filters in a filter group. All filters in a group must be combined using either **AND** or **OR**.

6. Optional. To add a filter group, do the following:
   a. Click **Add filter group**.
   b. Select **AND** or **OR**, depending on how you want the filter groups to be combined.
   c. Repeat steps 2 to 4 to create a filter.

   You can continue to add filters to the filter group, or create additional filter groups, to specify the exact set of data you want to import.

> **Note**
>
> The filters in each filter group are evaluated first, and then the filter groups are evaluated in relation to one another.
>
> You cannot mix Boolean operators when you combine multiple filter groups. All filter groups must be combined using either **AND** or **OR**.

7. Optional. In the **Import Preview** panel, click **Refresh** to see the records included in the import.

Using the "in" conditional operator

The **in** operator allows you to specify multiple test values. For example, you could create a conditional filter on the City field to limit the records that you import to only those for certain cities:

```
New York, San Francisco, Dallas
```

The follow rules apply to the **in** conditional operator:

- Separate test values with commas. Test values can contain spaces. (See the example above.)
- Enclose test values with double quotation marks `" "` if the values contain one or more single quotation marks `'`

  Depending on the data connector, you may also need to escape the single quotation mark character. For example: `"\'abc123\'"`
- Enclose test values with single quotation marks `' '` if the values contain one or more double quotation marks `"`
- Enclose test values with double quotation marks `" "`, or single quotation marks `' '`, if the values contain either of the following characters: comma `,` or backslash `\`

  The backslash must be followed by at least one character. For example: `"\a"` or `"\\"`
- In a single filter, do not use both double quotation marks `" "` and single quotation marks `' '` to enclose test values. Use one method or the other.

Adjust maximum field lengths

If the default maximum field lengths for imported character or memo fields are too short or too long, you can adjust them.

Data that exceeds the maximum field length is truncated when imported to Analytics.

> **Note**
>
> Field lengths cannot be specified individually. A single setting applies to all character fields or all memo fields in all tables in an import.

> **Tip**
> Be careful about making fields shorter based on the first few values in the **Import Preview**. Longer values may occur lower down in a table.

1. At the bottom of the Data Access window, increase or decrease the number of characters in one or both of these fields:
   - **Max Character Field Length**
   - **Max Memo Field Length**
2. In the **Import Preview** panel, click **Refresh** to update the field lengths in the preview.

> **Note**
> You may need to drag a preview column wider to see all the text in the column.

Import all fields as character data

Select **All Character** if you want to import all fields as character data.

Importing all fields as character data can simplify the import process and allow you to get troublesome fields into Analytics without losing data. Once the data is in Analytics, you can assign different data types, such as Numeric or Datetime, to the fields, and specify format details.

The **All Character** option is also useful if you are importing a table with identifier fields automatically assigned the Numeric data type by Analytics when in fact they should use the Character data type.

Edit the SQL import statement

If you understand SQL, you can directly edit the SQL import statement.

Editing the SQL import statement lets you control aspects of the data import not available through the user interface. For example, you can change field names in the SQL import statement.

> **Caution**
> Any changes you make in SQL Mode are lost if you return to the visual editor in the Data Access window.

1. Optional. Click the **SQL Mode** toggle button.
2. Edit the SQL import statement.

> **Note**
> You cannot use ACLScript syntax (commands or functions) in the body of the SQL import statement. You must use valid SQL syntax only.

3. In the **Import Preview** area, click **Refresh** to see the effect of the updated SQL on the data that will be imported.

Preview the import

At any point during the import process you can preview the import to see the effect of joins, field selection, filtering, and field length adjustment.

1. Optional. In the **Import Preview** area, select **Estimate Size** if you want to see an estimate of the number of records in the import, and the size of the Analytics data file (.fil) that will be created.

   > **Caution**
   > Use the **Estimate Size** option with caution. For large data sets and certain data sources, generating the estimate is processor-intensive and can be slow.

2. In the **Import Preview** area, click **Refresh** to see the data that will be imported.

Save the Analytics data file

When you have specified the data set you want, you save the imported data to an Analytics data file.

1. At the bottom of the Data Access window, click **Save**.
2. Specify the name of the Analytics table and click **Save**.

   The data is imported and the new table opens automatically in Analytics.

Update the Analytics table and data file

You can update an Analytics table and associated data file that you imported using the Data Access window. Updating a table refreshes it with the most recent source data.

Guidelines

- **Only the content is refreshed** -Refreshing an Analytics table updates only the content of existing fields. It does not update the table layout.

  You cannot refresh a table if the structure of the source data has changed – for example, if fields have been added or removed. You must re-import the data.

- **The table is open** – If the table is open when you refresh it, you temporarily need disk space equal to twice the size of the table. If you have limited disk space, close the table first before refreshing it.
- **Tables imported with Analytics 12** – Tables that were imported using the Data Access window in version 12 of Analytics are not refreshable, even if you are using a more recent version of Analytics.

  If you want to be able to refresh these tables, re-import them using Analytics 12.5 or later.

Steps

1. In the **Navigator**, right-click the Analytics table that you want to update and select **Refresh from Source**.
2. Click **Yes** in the confirmation dialog box.
3. If the **Password Prompt** appears, enter the password for the data source and click **OK**.

   > **Note**
   > You can also change the user name if you want to use a different account to access the data source.

4. If one or more prompts appear asking if you want to save changes, click **Yes**, unless you do not want to save the changes.

   The table is refreshed.

Joining tables in the Data Access window

Using the Data Access window, you can import up to ten tables in a single import operation.

You must join the tables in order to import them together. You cannot import multiple tables individually with one import operation.

> **Note**
>
> For information about joining Analytics tables once they are in Analytics, see "Joining tables" on page 889.
>
> This topic is about joining tables in the Data Access window as part of the data import process.

How joining tables works

Joining tables in the Data Access window is the process of selecting up to ten tables in the source data to add them to the **Staging Area**, and then joining the tables two at a time until all tables are joined.

You join the first two tables using a common key field – that is, a data element such as "Customer ID" that appears in both tables. When identical values exist in the key fields, the result is a match that joins individual records from the separate tables.

If you are joining more than two tables, you join the second table to a third table using a common key field in those two tables, and so on, until all tables are joined.

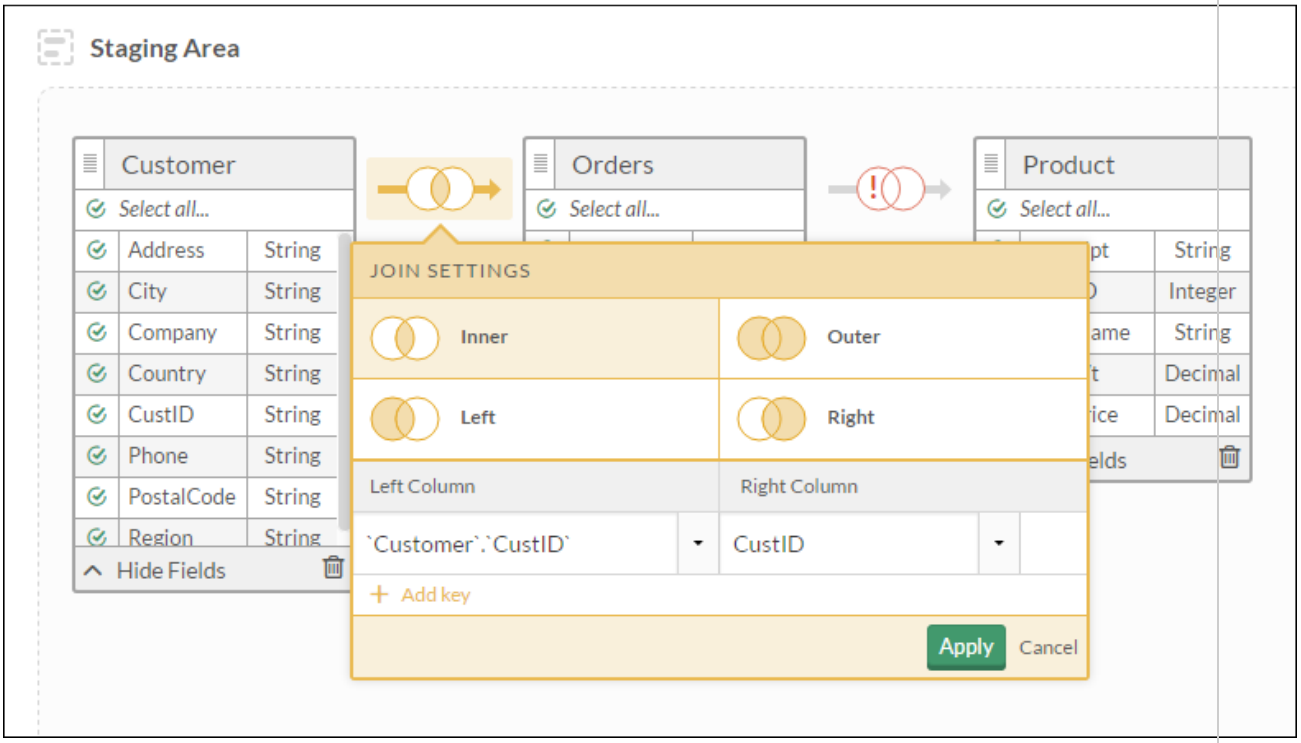
Example

You are working with accounts receivable data and you want a list of customers, the orders for each customer, and the details of the products on each order.

To assemble this data and import it into Analytics, you need to join three tables from the source data system:

- **Join #1** – you join the `Customer` and `Orders` tables using the key field of `CustID`, which appears in both tables
- **Join #2** – you join the `Orders` and `Product` tables using the key field of `ProdID`, which appears in both tables

In the figure below, only join #1 has been completed, so the second join icon is still red.



> **Tip**
>
> You can try this example join yourself in the Data Access window. Use the Microsoft Access connector and connect to this sample Microsoft Access file that comes with Analytics:
>
> `..\ACL Data\Sample Data Files\Sample.mdb`.

Join types

When you join tables you can choose from among four different join types. The join type you choose controls which records from the two original tables are included in the joined table.

Left and right tables

The two original tables are identified as "Left" and "Right" based on the order in which you select them:

- **Left table** - the first table you add to the **Staging Area**
- **Right table** - the second table you add to the **Staging Area**

Joining multiple tables

If you add more than two tables to the **Staging Area**, the left table is to the left of the join icon between the two tables you are joining, and the right table is to the right.

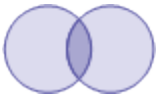
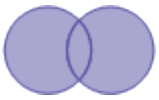
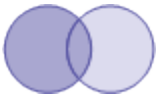
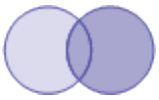
In the example above:

- **Join #1** - `Customer` is the left table and `Orders` is the right table
- **Join #2** - `Orders` is the left table and `Product` is the right table

Notice how `Orders` can be either the right or the left table depending on which join we are referring to.

Records included in the joined table

You can choose to include only matched records from original tables in a joined table, or you can also include unmatched records.

Join type		Records included in joined table			
		Matched left table records	Unmatched left table records	Matched right table records	Unmatched right table records
	Inner	✔		✔	
	Outer	✔	✔	✔	✔
	Left	✔	✔	✔	
	Right	✔		✔	✔

Joining using more than one key field

You may need to use more than one key field to join two tables if the values in a single key field are insufficiently unique to accurately join the tables.

Example

You want to join two tables by vendor ID but some of the vendors have more than one location and you want to keep the records for each location separate. To achieve this result you use both Vendor ID and Location fields as key fields.

If you use only Vendor ID as a key field, records for individual vendor locations are intermixed.

If you use only Location as a key field, records for different vendors are intermixed.

Vendor ID	Location
A-4538	Vancouver
A-4538	Burnaby
A-4538	Richmond
B-2204	Vancouver
B-2204	Burnaby

Joining tables from Apache Drill data sources

Using the visual editor in the Data Access window, you can join only two tables from an Apache Drill data source.

To join more than two tables from a Drill data source you must use **SQL Mode** and construct a join statement that does not use parentheses.

Joins of three or more tables constructed using the visual editor place parentheses into join statements, which are not supported for imports from Drill.

Connecting to Active Directory

Active Directory is Microsoft's Directory Server that provides a LDAP-compliant database containing objects such as users, groups, and computers. Use the Active Directory data connector to import your company's Active Directory data.

> **Note**
> You cannot use this connector independently of Analytics. You can configure a DSN for the connector driver using the Windows **ODBC Data Source Administrator**, however you must test the DSN connection from within Analytics and not from the connector's Windows DSN configuration dialog.

Before you start

To connect to Active Directory, you must gather the following:

- the domain name or IP address of the Active Directory server
- the correct connection port
- the connecting user account, including the distinguished name of the user and the password

For help gathering the connection prerequisites, contact the Active Directory administrator in your organization. If your administrator cannot help you, you or your administrator should contact Active Directory Support.

Create an Active Directory connection

1. From the Analytics main menu, select **Import > Database and application**.
2. From the **New Connections** tab, in the **ACL Connectors** section, select **Active Directory.**

   > **Tip**
   > You can filter the list of available connectors by entering a search string in the **Filter connections** box. Connectors are listed alphabetically.

3. In the **Data Connection Settings** panel, enter the connection settings and at the bottom of the panel, click **Save and Connect**.

   You can accept the default **Connection Name**, or enter a new one.

The connection for Active Directory is saved to the **Existing Connections** tab. In the future, you can reconnect to Active Directory from the saved connection.

Once the connection is established, the Data Access window opens to the **Staging Area** and you can begin importing data. For help importing data from Active Directory, see "Working with the Data Access window" on page 357.

Connection settings

Basic settings

Setting	Description	Example
User	The distinguished name of a user. Together with Password, this field is used to authenticate against the Active Directory server.	MYDOMAIN\test
Password	The password for the distinguished name of the specified user. Together with User, this field is used to authenticate against the Active Directory server. **Note** You may connect without providing a password if your Active Directory server permits anonymous connections. Based on your server's security configuration, anonymous connections may be able to list available tables. However, such connections may not be able to select data from some or all of the tables listed. For more information about your Active Directory security configuration, contact your company's administrator.	
Server	The domain name or IP of the Active Directory server. This does not need to include the LDAP:\\ portion, only the server domain name or IP.	10.120.1.110

Setting	Description	Example
Port	The port the Active Directory server is running on. The default value is 389. Together with Server, this property is used to specify the Active Directory server.	389
Base DN	The base portion of the distinguished name, used for limiting results to specific subtrees. Specifying a base DN may greatly improve performance when returning entries for large servers by limiting the number of entries that need to be examined.	DC=myConnection,DC=com
LDAP Version	The LDAP version used to connect to and communicate with the server. Valid options are 2 and 3 for LDAP versions 2 and 3.	2
Authentication Mechanism	The authentication mechanism to be used when connecting to the Active Directory server: ◦ **SIMPLE (default)** - default plaintext authentication is used to log in to the server ◦ **DIGESTMD5** - the more secure DIGEST-MD5 authentication is used ◦ **NEGOTIATE** - NTLM/Negotiate authentication will be used	SIMPLE
Scope	Whether to limit the scope of the search to: ◦ **WholeSubtree** - the whole subtree (BaseDN and all of its descendants) ◦ **SingleLevel** - a single level (BaseDN and its direct descendants) ◦ **BaseObject** - the base object (BaseDN only) **Tip** Limiting scope can greatly improve the search performance.	BaseObject

Advanced settings

Setting	Description	Example
Use Default DC	Used to connect to the default Domain Controller and authenticate using the current user credentials.	false
Convert Datetime to GMT	Converts datetime fields to GMT time zone during import. If false, the datetime value is converted to the operating system time zone of the machine running Analytics.	true
Limit Key Size	The maximum length of a primary key column. Setting the size to 0 will make the key length revert to the original length. This property makes the connector override the reported length of all the primary key columns.	255
Map to Long Varchar	Controls whether or not a column is returned as SQL_LONGVARCHAR. Use this setting to map any column larger than the specified size so they are reported as SQL_LONGVARCHAR instead of SQL_VARCHAR.	-1
Map to WVarchar	Controls whether or not string types map to SQL_WVARCHAR instead of SQL_VARCHAR. It is set by default. String columns must be mapped to SQL_WVARCHAR to accommodate various international character sets, so MapToWVarchar is set to true by default. You may set it to false to use SQL_VARCHAR instead.	true
Pseudo Columns	Indicates whether or not to include pseudo columns as columns to the table. This setting is particularly helpful in Entity Framework, which does not allow you to set a value for a pseudo column unless it is a table column.	MyTable=*

Setting	Description	Example
	The value of this connection setting is of the format "Table1=Column1, Table1=Column2, Table2=Column3". You can use the "*" character to include all tables and all columns.	
Upper Case Identifiers	Report all identifiers in uppercase, including table and column names.	false
SSL Server Certificate	The certificate to be accepted from the server when connecting using TLS/SSL. You can specify any of the following: ◦ full PEM certificate ◦ path to a local file containing the certificate ◦ the public key ◦ the MD5 Thumbprint (hex values can also be either space or colon separated) ◦ the SHA1 Thumbprint (hex values can also be either space or colon separated) If using a TLS/SSL connection, this property can be used to specify the TLS/SSL certificate to be accepted from the server. Any other certificate that is not trusted by the machine will be rejected.	C:\cert.cer
Support Enhanced SQL	Enhances SQL functionality beyond what can be supported through the API directly, by enabling in-memory client-side processing: ◦ **true** - the connector offloads as much of the SELECT statement processing as possible to Active Directory and then processes the rest of the query in memory. In this way the driver can execute unsupported predicates, joins, and aggregation ◦ **false** - the connector limits SQL execution to what is supported by the Active Directory API	false

Setting	Description	Example
	Note This setting must be false to support filtering using the where clause syntax. ### Execution of predicates The connector determines which of the clauses are supported by the data source and then pushes them to the source to get the smallest superset of rows that would satisfy the query. It then filters the rest of the rows locally. The filter operation is streamed, which enables the driver to filter effectively for even very large datasets. ### Execution of Joins The connector uses various techniques to join in memory. The driver trades off memory utilization against the requirement of reading the same table more than once. ### Execution of Aggregates The connector retrieves all rows necessary to process the aggregation in memory.	

Filtering the rows returned

The Active Directory connector uses an SQL filtering syntax that aligns closely with the LDAP search syntax. Some fields contain delimited data that represents multiple object attributes. Your WHERE clause must account for each value in these delimited fields as if they are distinct values, rather than a single string.

For more information about LDAP search filters, see MSDN Search Filter Syntax.

Filtering User on ObjectCategory and ObjectClass

Scenario

You are working with the **User** table and you want to import records where the **ObjectClass** has the following attributes:

- person
- user

You also want to limit the records to those where the **ObjectCategory** has the Computer attribute, and not Person.

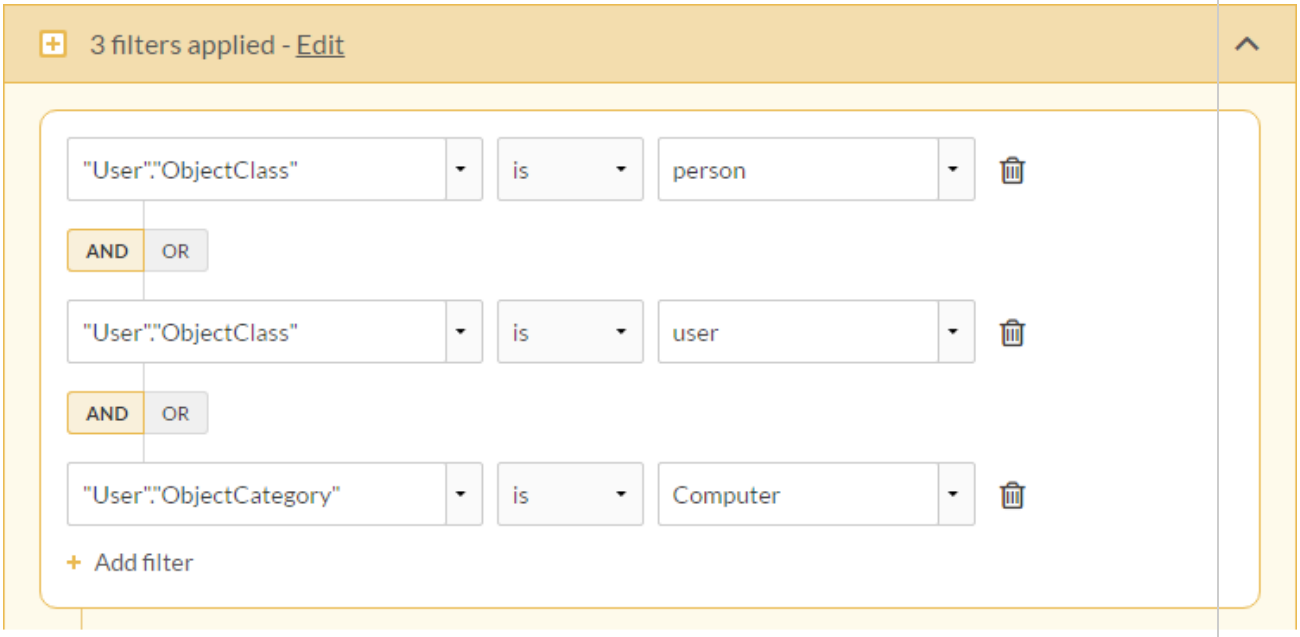
Connecting to the table

First, you connect to the Active Directory server, and select the **User** table (subset of fields shown).

ObjectCategory *(ASCII)*	ObjectClass *(ASCII)*	
CN=Person,CN=Schema,CN=Configuration,DC...	top;person;organizationalPerson;user	
CN=Person,CN=Schema,CN=Configuration,DC...	top;person;organizationalPerson;user	
CN=Person,CN=Schema,CN=Configuration,DC...	top;person;organizationalPerson;user	
CN=Computer,CN=Schema,CN=Configuration,...	top;person;organizationalPerson;user;computer	
CN=Person,CN=Schema,CN=Configuration,DC...	top;person;organizationalPerson;user	
CN=Person,CN=Schema,CN=Configuration,DC...	top;person;organizationalPerson;user	
CN=Computer,CN=Schema,CN=Configuration,...	top;person;organizationalPerson;user;computer	
CN=Person,CN=Schema,CN=Configuration,DC...	top;person;organizationalPerson;user	
CN=Computer,CN=Schema,CN=Configuration,...	top;person;organizationalPerson;user;computer	
CN=Person,CN=Schema,CN=Configuration,DC...	top;person;organizationalPerson;user	

Filtering the records

To limit the records to those you want to import, you apply a filter that treats each delimited value as a discrete field.



You then use **SQL Mode** to verify the WHERE clause that the filter constructs:

```
WHERE
  (
    "User"."ObjectClass" = N'person' AND
    "User"."ObjectClass" = N'user' AND
    "User"."ObjectCategory" = N'Computer'
  )
```

Filter results

Once the filter is applied, the table includes records that match the WHERE clause and you import the table.

ObjectCategory *(ASCII)*	ObjectClass *(ASCII)*	
CN=Computer,CN=Schema,CN=Configuration,	top;person;organizationalPerson;user;computer	
CN=Computer,CN=Schema,CN=Configuration,	top;person;organizationalPerson;user;computer	
CN=Computer,CN=Schema,CN=Configuration,	top;person;organizationalPerson;user;computer	
CN=Computer,CN=Schema,CN=Configuration,	top;person;organizationalPerson;user;computer	
CN=Computer,CN=Schema,CN=Configuration,	top;person;organizationalPerson;user;computer	
CN=Computer,CN=Schema,CN=Configuration,	top;person;organizationalPerson;user;computer	
CN=Computer,CN=Schema,CN=Configuration,	top;person;organizationalPerson;user;computer	

Joining Active Directory tables

Due to the data model used in LDAP-compliant databases such as Active Directory, SQL joins are not recommended. Joins may yield unexpected results.

If you need to join one or more tables from an Active Directory data source, you can import multiple tables using the Data Access window and then join them in Analytics. Use filters to limit the number of records and increase efficiency.

Connecting to Amazon Athena

Amazon Athena is an interactive query service that enables users to query data in Amazon S3, using standard SQL. You can use the Amazon Athena data connector to import your organization's Amazon Athena data.

> **Note**
> The Amazon Athena data connector is provided by our data partner, CData. For information on any of the connection fields, refer the documentation available at the CData website.

Before you start

To connect to Amazon Athena, you must gather the following:

- the S3 Staging Directory location where Amazon Athena stores the results of queries
- the AWS account access key
- the AWS account secret key
- the AWS region where your Amazon Web Server is hosted

For help gathering the connection prerequisites, contact the Amazon Athena administrator in your organization. If your administrator cannot help you, you or your administrator should contact Amazon Athena Support.

Create an Amazon Athena connection

1. From the Analytics main menu, select **Import > Database and application**.
2. From the **New Connections** tab, in the **ACL DSN Connectors (Bundled)** section, select **Amazon Athena.**

   > **Tip**
   > You can filter the list of available connectors by entering a search string in the **Filter connections** box. Connectors are listed alphabetically.

   The DSN Configuration dialog box appears.
3. In the **DSN Configuration** dialog box, click **Show Required**.
4. Provide values for the required fields, if any.

5. Click **Test Connection**.

> **Note**
> When you click the **Test Connection** button, it validates whether the connection and authentication details provided are correct. If you click **OK** without clicking Test Connection, the connection details are saved without any validation and may not work later if the values provided are incorrect.

   The log in page for Amazon Athena appears.

6. Provide the connection details and authenticate your login.

   When the connection is successful, a dialog box opens displaying the success message.

7. Click **OK** in the connection successful dialog box that appears.

   The connection to Amazon Athena is established successfully.

8. In the **DSN Configuration** dialog box, click **OK**.

The connection for Amazon Athena is saved to the **Existing Connections** tab. In the future, you can reconnect to Amazon Athena from the saved connection.

Once the connection is established, the Data Access window opens to the **Staging Area** and you can begin importing data. For help importing data from Amazon Athena, see "Working with the Data Access window" on page 357.

Rename the connection

When you create the Amazon Athena data connector, you cannot provide a Connection name of your choice and the connection is saved to the **Existing Connections** tab with the same name as your data source. If you want to rename it, navigate to the **Existing Connections** tab, expand the **ACL Connectors (DSN)** section, click the ellipsis icon corresponding to Amazon Athena and select **Rename connection**.

Connecting to Amazon DynamoDB

Amazon DynamoDB is a cloud data service. You can use the Amazon DynamoDB data connector to import your company's DynamoDB data.

> **Note**
> Analytics provides DynamoDB as an optional connector and if it is not available in your Data Access window, it is likely that the connector was not selected during installation. For more information, see "Install optional Analytics data connectors and Python engine" on page 2567.

Before you start

To connect to DynamoDB, you must gather the following:

- hosting region of your Amazon Web Services
- AWS account access key
- AWS account secret key

For help gathering the connection prerequisites, contact the DynamoDB administrator in your organization. If your administrator cannot help you, you or your administrator should contact DynamoDB Support.

Create a DynamoDB connection

1. From the Analytics main menu, select **Import > Database and application**.
2. From the **New Connections** tab, in the **ACL Connectors** section, select **DynamoDB.**

   > **Tip**
   > You can filter the list of available connectors by entering a search string in the **Filter connections** box. Connectors are listed alphabetically.

3. In the **Data Connection Settings** panel, enter the connection settings and at the bottom of the panel, click **Save and Connect**.

   You can accept the default **Connection Name**, or enter a new one.

The connection for DynamoDB is saved to the **Existing Connections** tab. In the future, you can reconnect to DynamoDB from the saved connection.

Once the connection is established, the Data Access window opens to the **Staging Area** and you can begin importing data. For help importing data from DynamoDB, see "Working with the Data Access window" on page 357.

Connection settings

Basic settings

Setting	Description	Example
Endpoint	Endpoint for the communication. If you are connecting to AWS, then specify the endpoint for the DynamoDB service on AWS. If you are connecting to DynamoDB Local, then specify the IP address of the server.	dynamodb.*[Region]*.amazonaws.com, where *[Region]* is the AWS region to use for your connection
DynamoDB Local	Specifies whether to connect to a DynamoDB Local server or the DynamoDB service on AWS. DynamoDB Local is a client-side database that supports the complete DynamoDB API, but does not manipulate any tables or data in DynamoDB itself.	Enabled
Port	The number of the TCP port that the DynamoDB Local server uses to listen for client connections. The default port number used by DynamoDB Local is 8000.	8000
Region	The hosting region for your Amazon Web Services.	NORTHERNVIRGINIA
Authentication	Specifies how the driver authenticates connections to DynamoDB: ◦ Disabled - The driver authenticates connections to DynamoDB by using an access key and a secret key. ◦ Enabled - The driver authenticates connections to DynamoDB by using a profile from a credentials file.	Disabled
Credentials File	The full path and name of the credentials file, where MFA credentials are saved. The default location is *%APPDATA%\\CData\\AmazonDynamoDB Data Provider\\CredentialsFile.txt*	

Setting	Description	Example
Profile Name	The name of the profile to use from the AWS credentials file.	
Access Key	Your AWS account access key. This value is accessible from your AWS security credentials page.	
Secret Key	Your AWS account secret key. This value is accessible from your AWS security credentials page.	
Enable Temporary Session	Specifies whether the driver uses temporary credentials: ◦ Disabled - The driver does not use temporary credentials. Enabled - The driver authenticates connections to DynamoDB by using temporary credentials. Temporary credentials consist of an access key, a secret key, and a session token, which are only valid for a limited amount of time.	
Temporary Session Token	The session token to use when connecting to DynamoDB using temporary security credentials, which are valid only for a limited period of time.	3600

Advanced settings

Setting	Description	Example
Number of Retries	The maximum number of times that the driver should retransmit a request to the DynamoDB database if the request fails from a recoverable error.	10
Limit Throughput	The percentage of the total provisioned read units that the driver is allowed to consume. By default, the driver is allowed to consume up to 30% of the provisioned throughput. Make sure to use a value that is appropriate for the number of client applications that will be using a specific table concurrently. For example, if you set this value to	30

Setting	Description	Example
	100, then one client will use all of the provisioned throughput units, preventing any other clients from using the table until more throughput becomes available.	
Active Metadata Location	Specifies whether to use the schema definition from a Database or Local file.	Database
Database	The name of the database table containing the schema definition that you want the driver to use when connecting to DynamoDB.	
Local File	The full path of a local JSON file containing the schema definition that you want the driver to use when connecting to DynamoDB.	

Data connector updates

When you upgrade Analytics, the Robots Agent, or AX Server, you should test any of your scripts that import data using one of the Analytics data connectors (ACCESSDATA command).

The possibility exists that changes made by third-party data sources or ODBC driver vendors required updates to one or more of the data connectors. Scripted data connections may need to be updated in order to continue working correctly.

- **Re-run the import** – The easiest way to update a connection is to manually perform an import using the Data Access window in the upgraded version of Analytics. Copy the ACCESSDATA command from the log and use it to update your script.

  > **Note**
  > Before connecting to a data source and re-running the import, clear the connector cache to flush the existing set of table names.
  > In the **Existing Connections** tab in the Data Access window, beside the connector name, select ··· **> Clear cache**.

- **Update field specifications** – You may also need to update field specifications in the script body to align with table schema changes in the data source or ODBC driver. Possible changes include field names, field data types, and field and record lengths.
- **Check the results of any filtering** – You should also check the results of any filtering that you apply as part of the data import. Confirm that the import filtering is including and excluding records correctly.

Connecting to Amazon Redshift

Amazon Redshift is a cloud data warehouse service used for business intelligence. Use the Amazon Redshift data connector to import your company's Amazon Redshift data.

> **Note**
> Analytics provides Amazon Redshift as an optional connector and if it is not available in your Data Access window, it is likely that the connector was not selected during installation. For more information, see "Install optional Analytics data connectors and Python engine" on page 2567.

Before you start

To connect to Amazon Redshift, you must gather the following:

- the server and database name from the ODBC URL of the Amazon Redshift cluster
- username of the user account that has permission to connect to the database
- password of the user account that has permission to connect to the database
- the port number that was specified when the cluster was launched (ensure the port is open if you have a firewall)

> **Note**
> When gathering the required connection information:
>
> - you can obtain the ODBC URL from the AWS Management Console by looking at the database properties for your cluster
> - verify that the connecting account has database permissions, not just Amazon Redshift permissions
>
> For more information about configuring an ODBC connection, see the online Amazon AWS documentation.

For help gathering the connection prerequisites, contact the Amazon Redshift administrator in your organization. If your administrator cannot help you, you or your administrator should contact Amazon Redshift Support.

Create an Amazon Redshift connection

1. From the Analytics main menu, select **Import > Database and application**.
2. From the **New Connections** tab, in the **ACL Connectors** section, select **Amazon Redshift.**

> **Tip**
> You can filter the list of available connectors by entering a search string in the **Filter connections** box. Connectors are listed alphabetically.

3. In the **Data Connection Settings** panel, enter the connection settings and at the bottom of the panel, click **Save and Connect**.

   You can accept the default **Connection Name**, or enter a new one.

The connection for Amazon Redshift is saved to the **Existing Connections** tab. In the future, you can reconnect to Amazon Redshift from the saved connection.

Once the connection is established, the Data Access window opens to the **Staging Area** and you can begin importing data. For help importing data from Amazon Redshift, see "Working with the Data Access window" on page 357.

Connection settings

Basic settings

Setting	Description	Example
Server	Host name or IP address of the Redshift cluster. You can obtain this value from the AWS Management Console.	
Port	The TCP port number of the Amazon Redshift server.	5439
Database	Name of the Redshift database. If you do not specify database, the connection uses the user's default database.	
User	The user name to access the Redshift server. If you are using keys to set driver options, UID takes precedence over Username.	

Setting	Description	Example
Password	The password used to authenticate the user.	
SSL Authentication	The SSL certificate verification mode to use when connecting to Redshift. The values available are as follows: ◦ **allow** - By default, connect without using SSL. If the server requires SSL connections, then use SSL. ◦ **disable** - Connect without using SSL. ◦ **prefer** - Connect using SSL if available. Otherwise, connect without using SSL. ◦ **require** - Connect only using SSL. ◦ **verify-ca** - Connect only using SSL and a trusted certificate authority. ◦ **verify-full** - Connect only using SSL, a trusted certificate authority, and a server name that matches the certificate.	require
Enable HTTP Proxy Connection	Specifies whether the driver can pass the IAM authentication processes through a proxy server.	
Proxy Server	The host name or IP address of a proxy server through which to pass IAM authentication processes.	
Proxy Port	Port number that the proxy server uses to listen for client connections.	

Advanced settings

Setting	Description	Example
Use Unicode	Specifies whether the driver returns Redshift data as Unicode or regular SQL types.	
Show Boolean Column As String	Specifies the SQL data type that the driver uses to return Boolean data. If enabled, the driver returns	

Setting	Description	Example
	Boolean columns as SQL_ VARCHAR data with a length of 5, else as SQL_BIT data.	
Text as LongVarChar	Specifies the SQL data type that the driver uses to return Text data. If disabled, the driver returns text columns as SQL_VARCHAR data.	
Max Varchar	The maximum data length for VARCHAR columns.	255
Max LongVarChar	The maximum data length for LongVarChar columns.	8190
Max Bytea	The maximum data length for Bytea columns.	255

Connecting to Amazon S3

Amazon Simple Storage Service (S3) is an object storage service that can be accessed through a web service interface. You can use the Amazon S3 data connector to import your organization's Amazon S3 data.

> **Note**
> The Amazon S3 data connector is provided by our data partner, CData. For information on any of the connection fields, refer the documentation available at the CData website.

Before you start

To connect to Amazon S3, you must gather the following:

- the AWS account access key
- the AWS account secret key
- the AWS region where your Amazon Web Server is hosted

For help gathering the connection prerequisites, contact the Amazon S3 administrator in your organization. If your administrator cannot help you, you or your administrator should contact Amazon S3 Support.

Create an Amazon S3 connection

1. From the Analytics main menu, select **Import > Database and application**.
2. From the **New Connections** tab, in the **ACL DSN Connectors (Bundled)** section, select **Amazon S3.**

   > **Tip**
   > You can filter the list of available connectors by entering a search string in the **Filter connections** box. Connectors are listed alphabetically.

   The DSN Configuration dialog box appears.
3. In the **DSN Configuration** dialog box, click **Show Required**.
4. Provide values for the required fields, if any.

5. Click **Test Connection**.

> **Note**
> When you click the **Test Connection** button, it validates whether the connection and authentication details provided are correct. If you click **OK** without clicking Test Connection, the connection details are saved without any validation and may not work later if the values provided are incorrect.

   The log in page for Amazon S3 appears.

6. Provide the connection details and authenticate your login.

   When the connection is successful, a dialog box opens displaying the success message.

7. Click **OK** in the connection successful dialog box that appears.

   The connection to Amazon S3 is established successfully.

8. In the **DSN Configuration** dialog box, click **OK**.

The connection for Amazon S3 is saved to the **Existing Connections** tab. In the future, you can reconnect to Amazon S3 from the saved connection.

Once the connection is established, the Data Access window opens to the **Staging Area** and you can begin importing data. For help importing data from Amazon S3, see "Working with the Data Access window" on page 357.

Rename the connection

When you create the Amazon S3 data connector, you cannot provide a Connection name of your choice and the connection is saved to the **Existing Connections** tab with the same name as your data source. If you want to rename it, navigate to the **Existing Connections** tab, expand the **ACL Connectors (DSN)** section, click the ellipsis icon corresponding to Amazon S3 and select **Rename connection**.

Connecting to Apache Cassandra

Apache Cassandra is a NoSQL database management system. Use the Apache Cassandra data connector to import your company's Cassandra data.

> **Note**
> Analytics provides Cassandra as an optional connector and if it is not available in your Data Access window, it is likely that the connector was not selected during installation. For more information, see "Install optional Analytics data connectors and Python engine" on page 2567.

Before you start

To connect to Cassandra, you must gather the following:

- the database server's host name or IP address
- the correct connection port
- your username and password if using authentication

For help gathering the connection prerequisites, contact the Cassandra administrator in your organization. If your administrator cannot help you, you or your administrator should contact Cassandra Support.

Create a Cassandra connection

1. From the Analytics main menu, select **Import > Database and application**.
2. From the **New Connections** tab, in the **ACL Connectors** section, select **Cassandra.**

   > **Tip**
   > You can filter the list of available connectors by entering a search string in the **Filter connections** box. Connectors are listed alphabetically.

3. In the **Data Connection Settings** panel, enter the connection settings and at the bottom of the panel, click **Save and Connect**.

   You can accept the default **Connection Name**, or enter a new one.

The connection for Cassandra is saved to the **Existing Connections** tab. In the future, you can reconnect to Cassandra from the saved connection.

Once the connection is established, the Data Access window opens to the **Staging Area** and you can begin importing data. For help importing data from Cassandra, see "Working with the Data Access window" on page 357.

Connection settings

Basic settings

Setting	Description	Example
Host	The IP address or host name of the Cassandra server.	
Port	The TCP port for the Cassandra database.	9042
Default keyspace	The default keyspace (schema) to connect to in Cassandra.	
Authentication Mechanism	The authentication mechanism to use to connect to Cassandra server. Available options are as follows: ◦ No Authentication ◦ User Name and Password	No Authentication
User name	The user name to use to access the Cassandra server.	
Password	The password corresponding to the user name provided.	

Advanced settings

Setting	Description	Example
Query mode	Specifies the query mode to use when sending queries to Cassandra. The options available are: ◦ **SQL** - Uses SQL_QUERY_MODE and executes all queries in SQL. ◦ **CQL** - Uses CQL_QUERY_MODE and executes all queries in CQL.	SQL with CQL Fallback

Setting	Description	Example
	◦ **SQL with CQL Fallback** - Uses SQL_WITH_CQL_FALLBACK_QUERY_MODE and executes all queries in SQL by default. If a query fails, then the driver executes the query in CQL.	
Tunable consistency	The specific Cassandra replica or the number of Cassandra replicas that must process a query for the query to be successful.	ONE
Load Balancing Policy	Specifies the load balancing policy to be used.	
Binary column length	The default column length to report for BLOB columns.	4000
String column length	The default column length to report for ASCII, TEXT, and VARCHAR columns.	4000
Virtual table name separator	The separator for naming a virtual table built from a collection. The name of a virtual table consists of the name of the original table, then the separator, and then the name of the collection.	_vt_
Enable Token Aware	Specifies whether to use a token-aware policy to improve load balancing and latency.	
Enable Latency Aware	Specifies whether the driver must use a latency-awareness algorithm to distribute the load away from slower-performing nodes.	
Enable Null Values Insertion	Specifies whether the driver must insert all NULL values as specified in INSERT statements.	
Enable Case Sensitive	Specifies whether the driver differentiates between capital and lower-case letters in schema, table, and column names. If this option is enabled, all schemas, tables, and columns must be double quotation marks (").	
Use SQL_WVARCHAR for string	Specifies whether to use SQL_	

Setting	Description	Example
data type	WVARCHAR for text and varchar types.	
Enable paging	Specifies whether to split large result sets into pages.	
Rows per page	When the **Enable Paging** option is enabled, use this option to specify the maximum number of rows to display on each page.	10000
SSL Options	Specifies how the driver uses SSL to connect to the Cassandra server. The options available are: ◦ **No SSL** - The driver does not use SSL. ◦ **One-way Server Verification** - If the **Enable Server Hostname Verification** option is enabled, the client verifies the Cassandra server using SSL. Otherwise, the driver connects to the Cassandra server using SSL but the client and server do not verify each other. ◦ **Two-way Server and Client Verification** - If the **Enable Server Hostname Verification** option is enabled, the client and the Cassandra server verify each other using SSL. Otherwise, the driver connects to the Cassandra server using SSL but the client and server do not verify each other.	No SSL
Enable Server Hostname Verification	Specifies whether the driver mandates the host name of the server to match the host name in the SSL certificate.	
Ssltrustedcertspath	The full path to the .pem file containing the certificate for verifying the server.	
Client-side Certificate	The full path to the .pem file containing the certificate for verifying the client.	
Client-side Private Key	The full path to the file containing the private key used to verify the	

Setting	Description	Example
	client.	
Key File Password	The password for the private key file that is specified in the Client-side Private Key field.	

Querying Cassandra

One advantage of the Apache Cassandra design is the ability to store data that is denormalized into fewer tables. By taking advantage of nested data structures such as sets, lists, and maps, transactions can be simplified. However, Analytics does not support accessing this type of data. By re-normalizing the data contained within collections (sets, lists, and maps) into virtual tables, the connector allows users to directly interact with the data but leave the storage of the data in its denormalized form in Cassandra.

If a table contains any collection columns, when the table is queried for the first time, the connector creates the following virtual tables:

- A "base" table, which contains the same data as the real table except for the collection columns.
- A virtual table for each collection column, which expands the nested data.

Virtual tables refer to the data in the real table, enabling the connector to access the denormalized data. By querying the virtual tables, you can access the contents of Cassandra collections via ODBC.

The base table and virtual tables appear as additional tables in the list of tables that exist in the database. The base table uses the same name as the real table that it represents. The virtual tables that represent collections are named using the name of the real table, a separator (**_vt_** by default), and the name of the column.

Example

ExampleTable is a Cassandra database table that contains an integer primary key column named **pk_int**, a list column, a map column, and a set column (named **StringSet**).

Source table with collections

pk_int	List	Map	StringSet
1	["1","2","3"]	{"S1" : "a", "S2" : "b"}	{"a", "b", "c"}

pk_int	List	Map	StringSet
3	["100","101","102","105"]	{"S1" : "t"}	{"a","e"}

The connector generates multiple virtual tables to represent this single table. The first virtual table is the base table:

Base table

pk_int
1
3

The base table contains the same data as the original database table except for the collections, which are omitted from this table and expanded in other virtual tables.

The following tables show the virtual tables that re-normalize the data from the **List**, **Map**, and **StringSet** columns:

List

pk_int	List#index	List#value
1	0	1
1	1	2
1	2	3
3	0	100
3	1	101
3	2	102
3	3	105

Map

pk_int	Map#key	Map#value
1	S1	a
1	S2	b
3	S1	t

StringSet

pk_int	StringSet#value
1	a
1	b
1	c
3	a
3	e

The foreign key columns in the virtual tables reference the primary key columns in the real table, and indicate which real table row the virtual table row corresponds to. The columns with names that end with #index or #key indicate the position of the data within the original list or map. The columns with names that end with #value contain the expanded data from the collection.

Data connector updates

When you upgrade Analytics, the Robots Agent, or AX Server, you should test any of your scripts that import data using one of the Analytics data connectors (ACCESSDATA command).

The possibility exists that changes made by third-party data sources or ODBC driver vendors required updates to one or more of the data connectors. Scripted data connections may need to be updated in order to continue working correctly.

- **Re-run the import** – The easiest way to update a connection is to manually perform an import using the Data Access window in the upgraded version of Analytics. Copy the ACCESSDATA command from the log and use it to update your script.

> **Note**
> Before connecting to a data source and re-running the import, clear the connector cache to flush the existing set of table names.
> In the **Existing Connections** tab in the Data Access window, beside the connector name, select ··· **> Clear cache**.

- **Update field specifications** – You may also need to update field specifications in the script body to align with table schema changes in the data source or ODBC driver. Possible changes include field names, field data types, and field and record lengths.
- **Check the results of any filtering** – You should also check the results of any filtering that you apply as part of the data import. Confirm that the import filtering is including and excluding records correctly.

Apache Cassandra data connector changes

Specific changes made to the Apache Cassandra data connector are listed below.

Analytics version	Change
14.2	The connector no longer supports connecting to Apache Cassandra 2.0. Connections can be made to Apache Cassandra 2.1, 2.2, and 3.0.

Connecting to Apache Drill

Drill is an Apache open-source SQL query engine for high-performance analysis on the semi-structured data coming from Big Data applications.

> **Note**
> Analytics provides Drill as an optional connector and if it is not available in your Data Access window, it is likely that the connector was not selected during installation. For more information, see "Install optional Analytics data connectors and Python engine" on page 2567.

Before you start

To connect to Drill, you must gather the following:

- the correct authentication type (basic or none)
- username and password if using basic authentication
- the correct connection type (Direct to Drillbit embedded mode or Zookeeper Quorum distributed mode)
- host and port for Drillbit or each node if using distributed mode

  In distributed mode, host and port information must be entered in the **Quorum** field in comma-delimited format:

  ```
  <host name/ip address> : <port number>, <host name/ip address> : <port number>, . . .
  ```

- name of the Drillbit cluster if using distributed mode

For help gathering the connection prerequisites, contact the Apache Drill administrator in your organization. If your administrator cannot help you, you or your administrator should contact Apache Drill Support.

For advanced connection properties, see the Apache Drill online help on *Configuring ODBC on Windows*.

Create a Drill connection

1. From the Analytics main menu, select **Import > Database and application**.
2. From the **New Connections** tab, in the **ACL Connectors** section, select **Apache Drill.**

> **Tip**
> You can filter the list of available connectors by entering a search string in the **Filter connections** box. Connectors are listed alphabetically.

3. In the **Data Connection Settings** panel, enter the connection settings and at the bottom of the panel, click **Save and Connect**.

   You can accept the default **Connection Name**, or enter a new one.

The connection for Apache Drill is saved to the **Existing Connections** tab. In the future, you can reconnect to Apache Drill from the saved connection.

Once the connection is established, the Data Access window opens to the **Staging Area** and you can begin importing data. For help importing data from Apache Drill, see "Working with the Data Access window" on page 357.

Connection settings

Basic settings

Setting	Description	Example
Connection Type	Specifies the driver connection type. The options available are: ◦ **Direct to Drillbit** - The driver connects to a single Drill server. ◦ **Zookeeper Quorum** - The driver connects to a ZooKeeper cluster.	Direct to Drillbit
Quorum	Specify the server(s) in your ZooKeeper cluster. Separate multiple servers using a comma (,).	
Cluster ID	Name of the ZooKeeper cluster that the driver connects to.	drillbits1
Host	The IP address or host name of the Drill server.	localhost
Port	The TCP port that the Drill server	31010

Setting	Description	Example
	uses to listen for client connections.	
Authentication Type	Specifies how the driver authenticates the connection to Drill. The options available as: ◦ **No Authentication** - The driver does not authenticate the connection to Drill. ◦ **Basic Authentication** - The driver authenticates the connection using a user name and a password.	No Authentication
User	User name to authenticate to the Drill server.	
Password	Password to authenticate to the Drill server.	
Catalog	Name of the synthetic catalog under which all of the schemas/databases are organized. This catalog name is used as the value for SQL_DATABASE_NAME or CURRENT CATALOG.	DRILL
Default Schema	Name of the database schema to use when a schema is not explicitly specified in a query. You can issue queries on other schemas by explicitly specifying the schema in the query.	
Disable Async	Specifies whether the driver supports asynchronous queries.	

Advanced settings

Setting	Description	Example

Advanced Properties	For configuring the driver. Separate each advanced property by using a semicolon (;), and then surround all the advanced properties in a connection string by using braces ({ }).	`CastAnyToVarchar=true;`
		`HandshakeTimeout=5;`
		`QueryTimeout=180;`
		`TimestampTZDisplayTimezone=local;`
		`ExcludedSchemas=sys,INFORMATION_`
		`SCHEMA;`
		`NumberOfPrefetchBuffers=5`

Connecting to Apache HBase

Apache HBase is the Hadoop database that is a distributed, scalable, big data store. You can use the Apache HBase data connector to import your company's HBase data.

> **Note**
> Analytics provides HBase as an optional connector and if it is not available in your Data Access window, it is likely that the connector was not selected during installation. For more information, see "Install optional Analytics data connectors and Python engine" on page 2567.

Before you start

To connect to HBase, you must gather the following:

- the database server's host name or IP address
- the correct connection port
- your username and password if using authentication

For help gathering the connection prerequisites, contact the HBase administrator in your organization. If your administrator cannot help you, you or your administrator should contact HBase Support.

Create an HBase connection

1. From the Analytics main menu, select **Import > Database and application**.
2. From the **New Connections** tab, in the **ACL Connectors** section, select **HBase.**

   > **Tip**
   > You can filter the list of available connectors by entering a search string in the **Filter connections** box. Connectors are listed alphabetically.

3. In the **Data Connection Settings** panel, enter the connection settings and at the bottom of the panel, click **Save and Connect**.

   You can accept the default **Connection Name**, or enter a new one.

The connection for HBase is saved to the **Existing Connections** tab. In the future, you can reconnect to HBase from the saved connection.

Once the connection is established, the Data Access window opens to the **Staging Area** and you can begin importing data. For help importing data from HBase, see "Working with the Data Access window" on page 357.

Connection settings

Basic settings

Setting	Description	Example
Host	The IP address or host name of the HBase server.	
Port	The port for the Apache HBase server.	9090
Authentication Type	The authentication mechanism to use for the connection to the HBase server. The options available are: ◦ **No Authentication** - The driver does not authenticate the connection. ◦ **Basic Authentication** - The driver authenticates the connection using an HBase user name and password.	No Authentication
User	User name to access the HBase instance.	
Password	Password corresponding to the user name to access the HBase instance.	

Advanced settings

Setting	Description	Example
Maximum rows per fetch	Maximum number of rows that a query can return per request.	4096
Schema definition row limit	Number of rows that the driver samples when generating a schema.	1024

Connecting to Apache Hive

Apache Hive is a cloud data service. You can use the Apache Hive data connector to import your company's Hive data.

> **Note**
> Analytics provides Hive as an optional connector and if it is not available in your Data Access window, it is likely that the connector was not selected during installation. For more information, see "Install optional Analytics data connectors and Python engine" on page 2567.

Before you start

To connect to Hive, you must gather the following:

- username and password
- the correct connection port
- the authentication scheme used
- the server's host name or IP address
- the transport mode to communicate with the server
- Read access

For help gathering the connection prerequisites, contact the Hive administrator in your organization. If your administrator cannot help you, you or your administrator should contact Hive Support.

Create a Hive connection

1. From the Analytics main menu, select **Import > Database and application**.
2. From the **New Connections** tab, in the **ACL Connectors** section, select **Hive.**

   > **Tip**
   > You can filter the list of available connectors by entering a search string in the **Filter connections** box. Connectors are listed alphabetically.

3. In the **Data Connection Settings** panel, enter the connection settings and at the bottom of the panel, click **Save and Connect**.

   You can accept the default **Connection Name**, or enter a new one.

The connection for Hive is saved to the **Existing Connections** tab. In the future, you can reconnect to Hive from the saved connection.

Once the connection is established, the Data Access window opens to the **Staging Area** and you can begin importing data. For help importing data from Hive, see "Working with the Data Access window" on page 357.

Connection settings

Basic settings

Setting	Description	Example
Hive Server Type	Specifies the Hive Server instance to connect to.	Hive Server 2
Service Discovery Mode	Specifies how the Hive Server services are discovered. The options available are: ◦ **No Service Discovery** - The driver connects to Hive without using a discovery service. ◦ **ZooKeeper** - The driver discovers Hive Server services through the ZooKeeper service.	No Service Discovery
Host(s)	IP address or host name of the Hive server.	
Port	The port for the connection to the Hive Server instance.	10000
Database	Name of the database schema to use when a schema is not explicitly specified in a query.	
ZooKeeper Namespace	The namespace configured on ZooKeeper for the Hive Server 2 znodes.	
Authentication Mechanism	Specifies the authentication mechanism to use. The options available are: ◦ **No Authentication** ◦ **Kerberos** ◦ **User Name** ◦ **Username and Password** ◦ **Windows Azure HDInsight Service**	No Authentication
Realm	The realm of the Hive Server 2 host.	

Setting	Description	Example
Host FQDN	Fully qualified domain name of the Hive Server host.	_HOST
Service Name	The Kerberos service principal name of the Hive server.	
User Name	User name to authenticate to the Hive Server.	
Password	Password for the username to authenticate to the Hive Server.	
Delegation UID	User ID of the delegated user to whom the driver must delegate all Hive operations, rather than to the authenticated user for the connection.	
Thrift Transport	Specifies the transport protocol to use in the Thrift layer. The options available are: ◦ **Binary** ◦ **SASL** ◦ **HTTP**	Binary

Advanced settings

Setting	Description	Example
Enable SSL	Specifies whether the client uses an SSL encrypted connection to communicate with the Hive server.	
Allow Common Name Host Name Mismatch	Specifies whether a CA-issued SSL certificate name must match the host name of the Hive server.	
Allow Self-signed Server Certificate	Specifies whether the driver allows a connection to the Hive server that uses a self-signed certificate, even if this certificate is not in the list of trusted certificates.	
Trusted Certificates	The full path of the .pem file containing trusted CA certificates, for verifying the server when using SSL.	

Setting	Description	Example
Two-way SSL	Specifies whether two-way SSL is enabled.	
Client Certificate File	The full path to the .pem file containing the SSL certificate of the client.	
Client Private Key File	The full path to the .pem file containing the SSL private key of the client.	
Client Private Key Password	The password of the private key file specified in the **Client Private Key File** field.	
Use Native Query	Specifies whether the driver uses native HiveQL queries. If this option is not selected the driver converts the queries emitted by an application into an equivalent form in HiveQL.	
Fast SQLPrepare	Specifies whether the driver defers query execution to SQLExecute.	
Driver Config Take Precedence	Specifies whether driver-wide configuration settings take precedence over connection and DSN settings.	
Use Async Exec	Specifies whether to execute queries synchronously or asynchronously.	
Async Exec Poll Interval	The time in milliseconds between each poll for the query execution status.	100
Get Tables with Query	Specifies whether the driver uses the SHOW TABLES query to retrieve table names from the database. If disabled, the driver uses GetTables Thrift API call.	
Unicode SQL character types	Specifies the SQL types to be returned for string data types. When enabled, the driver returns SQL_WVARCHAR for STRING and VARCHAR columns, and returns SQL_WCHAR for CHAR columns.	

Setting	Description	Example
Show System Table	Specifies whether the driver returns the hive_system table for catalog function calls such as SQLTables and SQLColumns.	
Use only SSPI	Specifies whether the driver handles Kerberos authentication with the SSPI plugin or with MIT Kerberos.	
Invalid Session Auto Recover	Specifies whether the driver opens a new session automatically when the existing session is no longer valid.	
Rows fetched per block	Maximum number of rows that a query returns at a time.	10000
Default string column length	Maximum number of characters that can be contained in STRING columns.	255
Binary column length	Maximum data length for BINARY columns.	32767
Decimal column scale	Maximum number of digits to the right of the decimal point for numeric data types.	10
Socket Timeout	Number of seconds that an operation can remain idle before it is closed.	60
HTTP Path	The partial URL corresponding to the Hive server.	

Hive connection fields

Column unique names

Hive connections that are made through the Data Access window use a connection string parameter EnableUniqueColumnName that is set to 0 by default. This parameter must have a value of 0 to ensure that correct column names are retrieved when connecting.

If you create a Hive connection using a DSN rather than from the Data Access window, this value is set to 1 by default. You must change it to 0 in the Windows registry for your connection to work.

> **Note**
> Scripts that use DSN connections established in versions of ACL older than 13.1 continue to work after upgrading to version 13.1.

Connecting to Apache Spark

Apache Spark is a cloud data service. You can use the Apache Spark data connector to import your company's Spark data.

> **Note**
> Analytics provides Spark as an optional connector and if it is not available in your Data Access window, it is likely that the connector was not selected during installation. For more information, see "Install optional Analytics data connectors and Python engine" on page 2567.

Before you start

To connect to Spark, you must gather the following:

- the username and password
- the correct connection port
- the authentication scheme used
- the server's host name or IP address
- the transport mode to communicate with the server
- Read access

For help gathering the connection prerequisites, contact the Spark administrator in your organization. If your administrator cannot help you, you or your administrator should contact Spark Support.

Create a Spark connection

1. From the Analytics main menu, select **Import > Database and application**.
2. From the **New Connections** tab, in the **ACL Connectors** section, select **Spark.**

   > **Tip**
   > You can filter the list of available connectors by entering a search string in the **Filter connections** box. Connectors are listed alphabetically.

3. In the **Data Connection Settings** panel, enter the connection settings and at the bottom of the panel, click **Save and Connect**.

   You can accept the default **Connection Name**, or enter a new one.

The connection for Spark is saved to the **Existing Connections** tab. In the future, you can reconnect to Spark from the saved connection.

Once the connection is established, the Data Access window opens to the **Staging Area** and you can begin importing data. For help importing data from Spark, see "Working with the Data Access window" on page 357.

Connection settings

Basic settings

Setting	Description	Example
Spark Server Type	Specifies the Spark Server instance to connect to.	SharkServer2
Host(s)	IP address or host name of the Spark server.	
Port	The port for the connection to the Spark Server instance.	10000
Database	Name of the database schema to use when a schema is not explicitly specified in a query.	default
Authentication Mechanism	Specifies the authentication mechanism to use. The options available are: ◦ **No Authentication** ◦ **Kerberos** ◦ **Username** ◦ **Username and Password** ◦ **Windows Azure HDInsight Emulator** ◦ **Windows Azure HDInsight Service**	No Authentication
Realm	The realm of the Spark Thrift Server host.	
Host FQDN	Fully qualified domain name of the Spark Thrift Server host.	_HOST
Service Name	The Kerberos service principal name of the Spark server.	
User Name	User name to authenticate to the Spark Server.	
Password	Password for the username to authenticate to the Spark Server.	

Setting	Description	Example
Delegation UID	User ID of the delegated user to whom the driver must delegate all Spark operations, rather than to the authenticated user for the connection.	
Thrift Transport	Specifies the transport protocol to use in the Thrift layer. The options available are: ◦ **Binary** ◦ **SASL** ◦ **HTTP**	Binary

Advanced settings

Setting	Description	Example
Enable SSL	Specifies whether the client uses an SSL encrypted connection to communicate with the Spark server.	
Allow Common Name Host Name Mismatch	Specifies whether a CA-issued SSL certificate name must match the host name of the Spark server.	
Allow Self-signed Server Certificate	Specifies whether the driver allows a connection to the Spark server that uses a self-signed certificate, even if this certificate is not in the list of trusted certificates.	
Trusted Certificates	The full path of the .pem file containing trusted CA certificates, for verifying the server when using SSL.	
Two-way SSL	Specifies whether two-way SSL is enabled.	
Client Certificate File	The full path to the .pem file containing the SSL certificate of the client.	
Client Private Key File	The full path to the .pem file containing the SSL private key of the client.	
Client Private Key Password	The password of the private key file specified in the **Client Private Key**	

Setting	Description	Example
	File field.	
Use Native Query	Specifies whether the driver uses native HiveQL queries. If this option is not selected the driver converts the queries emitted by an application into an equivalent form in HiveQL.	
Fast SQLPrepare	Specifies whether the driver defers query execution to SQLExecute.	
Driver Config Take Precedence	Specifies whether driver-wide configuration settings take precedence over connection and DSN settings.	
Use Async Exec	Specifies whether to execute queries synchronously or asynchronously.	
Async Exec Poll Interval	The time in milliseconds between each poll for the query execution status.	100
Get Tables with Query	Specifies whether the driver uses the SHOW TABLES query to retrieve table names from the database. If disabled, the driver uses GetTables Thrift API call.	1
Unicode SQL character types	Specifies the SQL types to be returned for string data types. When enabled, the driver returns SQL_WVARCHAR for STRING and VARCHAR columns, and returns SQL_WCHAR for CHAR columns.	
Show System Table	Specifies whether the driver returns the spark_system table for catalog function calls such as SQLTables and SQLColumns.	
Use only SSPI	Specifies whether the driver handles Kerberos authentication with the SSPI plugin or with MIT Kerberos.	
Invalid Session Auto Recover	Specifies whether the driver opens a new session automatically when the existing session is no longer valid.	

Setting	Description	Example
Rows fetched per block	Maximum number of rows that a query returns at a time.	10000
Default string column length	Maximum number of characters that can be contained in STRING columns.	255
Binary column length	Maximum data length for BINARY columns.	32767
Decimal column scale	Maximum number of digits to the right of the decimal point for numeric data types.	10
Socket Timeout	Number of seconds that an operation can remain idle before it is closed.	60
HTTP Path	The partial URL corresponding to the Spark server.	/spark

Spark connection fields

Column unique names

Spark connections that are made through the Data Access window use a connection string parameter EnableUniqueColumnName that is set to 0 by default. This parameter must have a value of 0 to ensure that correct column names are retrieved when connecting.

If you create a Spark connection using a DSN rather than from the Data Access window, this value is set to 1 by default. You must change it to 0 in the Windows registry for your connection to work.

> **Note**
> Scripts that use DSN connections established in versions of ACL older than 13.1 continue to work after upgrading to version 13.1.

Connecting to AWS Data Management

AWS Management is an intuitive web-based interface for managing Amazon Web Services.You can use the AWS Data Management data connector to import your organization's AWS Management data.

> **Note**
> The AWS Data Management data connector is provided by our data partner, CData. For information on any of the connection fields, refer the documentation available at the CData website.

Before you start

To connect to Amazon Management, you must gather the following:

- the AWS account access key
- the AWS account secret key
- the AWS region where your Amazon Web Server is hosted

For help gathering the connection prerequisites, contact the AWS Data Management administrator in your organization. If your administrator cannot help you, you or your administrator should contact AWS Data Management Support.

Create an AWS Management connection

1. From the Analytics main menu, select **Import > Database and application**.
2. From the **New Connections** tab, in the **ACL DSN Connectors (Bundled)** section, select **AWS Data Management.**

   > **Tip**
   > You can filter the list of available connectors by entering a search string in the **Filter connections** box. Connectors are listed alphabetically.

   The DSN Configuration dialog box appears.
3. In the **DSN Configuration** dialog box, click **Show Required**.
4. Provide values for the required fields, if any.

5. Click **Test Connection**.

> **Note**
> When you click the **Test Connection** button, it validates whether the connection and authentication details provided are correct. If you click **OK** without clicking Test Connection, the connection details are saved without any validation and may not work later if the values provided are incorrect.

   The log in page for AWS Data Management appears.

6. Provide the connection details and authenticate your login.

   When the connection is successful, a dialog box opens displaying the success message.

7. Click **OK** in the connection successful dialog box that appears.

   The connection to AWS Data Management is established successfully.

8. In the **DSN Configuration** dialog box, click **OK**.

The connection for AWS Data Management is saved to the **Existing Connections** tab. In the future, you can reconnect to AWS Data Management from the saved connection.

Once the connection is established, the Data Access window opens to the **Staging Area** and you can begin importing data. For help importing data from AWS Data Management, see "Working with the Data Access window" on page 357.

Rename the connection

When you create the AWS Data Management data connector, you cannot provide a Connection name of your choice and the connection is saved to the **Existing Connections** tab with the same name as your data source. If you want to rename it, navigate to the **Existing Connections** tab, expand the **ACL Connectors (DSN)** section, click the ellipsis icon corresponding to AWS Data Management and select **Rename connection**.

Connecting to Azure Data Management

Azure Data Management provides options to manage your Azure data. You can use the Azure Data Management data connector to import your organization's Azure data.

> **Note**
> The Azure Data Management data connector is provided by our data partner, CData. For information on any of the connection fields, refer the documentation available at the CData website.

Before you start

To connect to Azure Data Management, you must gather the credentials to connect to the Azure Data Management server.

For help gathering the connection prerequisites, contact the Azure Data Management administrator in your organization. If your administrator cannot help you, you or your administrator should contact Azure Data Management Support.

Create an Azure Data Management connection

1. From the Analytics main menu, select **Import > Database and application**.
2. From the **New Connections** tab, in the **ACL DSN Connectors (Bundled)** section, select **Azure Data Management.**

   > **Tip**
   > You can filter the list of available connectors by entering a search string in the **Filter connections** box. Connectors are listed alphabetically.

   The DSN Configuration dialog box appears.
3. In the **DSN Configuration** dialog box, click **Show Required**.
4. Provide values for the required fields, if any.

5. Click **Test Connection**.

> **Note**
> When you click the **Test Connection** button, it validates whether the connection and authentication details provided are correct. If you click **OK** without clicking Test Connection, the connection details are saved without any validation and may not work later if the values provided are incorrect.

   The log in page for Azure Data Management appears.

6. Provide the connection details and authenticate your login.

   When the connection is successful, a dialog box opens displaying the success message.

7. Click **OK** in the connection successful dialog box that appears.

   The connection to Azure Data Management is established successfully.

8. In the **DSN Configuration** dialog box, click **OK**.

The connection for Azure Data Management is saved to the **Existing Connections** tab. In the future, you can reconnect to Azure Data Management from the saved connection.

Once the connection is established, the Data Access window opens to the **Staging Area** and you can begin importing data. For help importing data from Azure Data Management, see "Working with the Data Access window" on page 357.

Rename the connection

When you create the Azure Data Management data connector, you cannot provide a Connection name of your choice and the connection is saved to the **Existing Connections** tab with the same name as your data source. If you want to rename it, navigate to the **Existing Connections** tab, expand the **ACL Connectors (DSN)** section, click the ellipsis icon corresponding to Azure Data Management and select **Rename connection**.

Connecting to Azure Table Storage

Azure Table Storage is a cloud-based NoSQL datastore that allows you to store massive amounts of structured and non-relational data. You can use the Azure Table Storage data connector to import your organization's Azure Table Storage data.

> **Note**
> The Azure Table Storage data connector is provided by our data partner, CData. For information on any of the connection fields, refer the documentation available at the CData website.

Before you start

To connect to Azure Table Storage, you must gather the credentials to connect to the Azure Table Storage server.

For help gathering the connection prerequisites, contact the Azure Table Storage administrator in your organization. If your administrator cannot help you, you or your administrator should contact Azure Table Storage Support.

Create an Azure Table Storage connection

1. From the Analytics main menu, select **Import > Database and application**.
2. From the **New Connections** tab, in the **ACL DSN Connectors (Bundled)** section, select **Azure Table Storage.**

   > **Tip**
   > You can filter the list of available connectors by entering a search string in the **Filter connections** box. Connectors are listed alphabetically.

   The DSN Configuration dialog box appears.
3. In the **DSN Configuration** dialog box, click **Show Required**.
4. Provide values for the required fields, if any.

5. Click **Test Connection**.

> **Note**
> When you click the **Test Connection** button, it validates whether the connection and authentication details provided are correct. If you click **OK** without clicking Test Connection, the connection details are saved without any validation and may not work later if the values provided are incorrect.

   The log in page for Azure Table Storage appears.

6. Provide the connection details and authenticate your login.

   When the connection is successful, a dialog box opens displaying the success message.

7. Click **OK** in the connection successful dialog box that appears.

   The connection to Azure Table Storage is established successfully.

8. In the **DSN Configuration** dialog box, click **OK**.

The connection for Azure Table Storage is saved to the **Existing Connections** tab. In the future, you can reconnect to Azure Table Storage from the saved connection.

Once the connection is established, the Data Access window opens to the **Staging Area** and you can begin importing data. For help importing data from Azure Table Storage, see "Working with the Data Access window" on page 357.

Rename the connection

When you create the Azure Table Storage data connector, you cannot provide a Connection name of your choice and the connection is saved to the **Existing Connections** tab with the same name as your data source. If you want to rename it, navigate to the **Existing Connections** tab, expand the **ACL Connectors (DSN)** section, click the ellipsis icon corresponding to Azure Table Storage and select **Rename connection**.

Connecting to Box

Box is a cloud-based content management service, which enables file sharing and collaboration for businesses. You can use the Box data connector to import your organization's Box data .

> **Note**
> The Box data connector is provided by our data partner, CData. For information on any of the connection fields, refer the documentation available at the CData website.

Before you start

To connect to Box, you must gather the credentials to connect to the Box server.

For help gathering the connection prerequisites, contact the Box administrator in your organization. If your administrator cannot help you, you or your administrator should contact Box Support.

Create a Box connection

1. From the Analytics main menu, select **Import > Database and application**.
2. From the **New Connections** tab, in the **ACL DSN Connectors (Bundled)** section, select **Box.**

   > **Tip**
   > You can filter the list of available connectors by entering a search string in the **Filter connections** box. Connectors are listed alphabetically.

   The DSN Configuration dialog box appears.
3. In the **DSN Configuration** dialog box, click **Test Connection**.

   The Box Log in page appears.
4. To log in:
   - If you have the Box account credentials, provide the credentials.
   - If your organization uses SSO, click **Use Single Sign On (SSO)**.
5. Click **Authorize**.

   If you are using SSO, provide the credentials and log in.
6. On the page that appears, click **Grant access to Box**.

   The page is authorized successfully.
7. Click **OK** in the connection successful dialog box that appears.

   The connection to Box is established successfully.

8. In the **DSN Configuration** dialog box, click **OK**.

The connection for Box is saved to the **Existing Connections** tab with the connection name as **Box**. In future, you can reconnect to Box using the saved connection.

Once the connection is established, the Data Access window opens to the **Staging Area** and you can begin importing data. For help importing data from Box, see "Working with the Data Access window" on page 357.

> **Note**
> When connected to Box through multi-factor authentication (MFA), the authentication token expires roughly after one hour and causes any existing connections and imports happening to fail. Once the authentication failure happens, the existing connection will not work. You must create a new connection to Box for your further operations.

Rename the connection

When you create the Box data connector, you cannot provide a Connection name of your choice and the connection is saved to the **Existing Connections** tab with the same name as your data source. If you want to rename it, navigate to the **Existing Connections** tab, expand the **ACL Connectors (DSN)** section, click the ellipsis icon corresponding to Box and select **Rename connection**.

Connecting to Cloudera Impala

Cloudera Impala is a cloud data service. You can use the Cloudera Impala data connector to import your company's Impala data.

> **Note**
> Analytics provides Impala as an optional connector and if it is not available in your Data Access window, it is likely that the connector was not selected during installation. For more information, see "Install optional Analytics data connectors and Python engine" on page 2567.

Before you start

To connect to Impala, you must gather the following:

- username
- password
- Read access

For help gathering the connection prerequisites, contact the Impala administrator in your organization. If your administrator cannot help you, you or your administrator should contact Impala Support.

Create an Impala connection

1. From the Analytics main menu, select **Import > Database and application**.
2. From the **New Connections** tab, in the **ACL Connectors** section, select **Impala.**

   > **Tip**
   > You can filter the list of available connectors by entering a search string in the **Filter connections** box. Connectors are listed alphabetically.

3. In the **Data Connection Settings** panel, enter the connection settings and at the bottom of the panel, click **Save and Connect**.

   You can accept the default **Connection Name**, or enter a new one.

The connection for Impala is saved to the **Existing Connections** tab. In the future, you can reconnect to Impala from the saved connection.

Once the connection is established, the Data Access window opens to the **Staging Area** and you can begin importing data. For help importing data from Impala, see "Working with the Data Access window" on page 357.

Connection settings

Basic settings

Setting	Description	Example
Host	IP address or host name of the Impala server.	
Port	Port for the connection to the Impala server instance.	
Database	Name of the Impala database to use by default.	
Authentication Mechanism	The authentication mechanism to use. Options available are: ◦ **No Authentication** ◦ **Kerberos** ◦ **SASL User Name** ◦ **User Name and Password**	No Authentication
Realm	Realm of the Impala host.	
Host FQDN	Fully qualified domain name of the Impala host.	_HOST
Service Name	Kerberos service principal name of the Impala server.	impala
User Name	User name to access the Impala server.	
Password	Password to authenticate access to the Impala server.	
Transport Buffer Size	Number of bytes to reserve in memory for buffering unencrypted data from the network.	1000
Use Simple Authentication and Security Layer (SASL)	Specifies whether the driver uses SASL to handle authentication.	
Delegation UID	When a user ID is specified for this option, the Impala driver delegates all operations against Impala to the specified user, rather than to the authenticated user for the connection.	

Advanced settings

Setting	Description	Example
Enable SSL	Specifies whether the client uses an SSL encrypted connection to communicate with the Impala.	
Allow Common Name Host Name Mismatch	Specifies whether a CA-issued SSL certificate name must match the host name of the Impala server.	
Allow Self-signed Server Certificate	Specifies whether the driver allows a connection to an Impala server that uses a self-signed certificate.	
Trusted Certificates	Full path of the .pem file containing the trusted CA certificates, for verifying the server when using SSL.	
Use Native Query	Specifies whether the driver uses native Impala SQL queries. If this option is not selected, the driver converts the queries emitted by an application into an equivalent form in Impala SQL. If the application is Impala-aware and already emits Impala SQL, then enable this option to avoid the extra overhead of query transformation.	
Enable Simulated Transactions	Specifies whether the driver should simulate transactions. When disabled, the driver returns an error if it attempts to run a query that contains transaction statements.	
Use SQL Unicode Types	Specifies the SQL types to be returned for string data types. When enabled, the driver returns SQL_WVARCHAR for STRING and VARCHAR columns, and returns SQL_WCHAR for CHAR columns.	
Rows fetched per block	Maximum number of rows that a query returns at a time.	10000
Socket timeout	Number of seconds that the TCP socket waits for a response from the server before timing out the request and returning an error	30

Setting	Description	Example
	message. When set to 0, the TCP socket does not time out any requests.	
String Column Length	Maximum number of characters that can be contained in STRING columns.	32767

Data connector updates

When you upgrade Analytics, the Robots Agent, or AX Server, you should test any of your scripts that import data using one of the Analytics data connectors (ACCESSDATA command).

The possibility exists that changes made by third-party data sources or ODBC driver vendors required updates to one or more of the data connectors. Scripted data connections may need to be updated in order to continue working correctly.

- **Re-run the import** – The easiest way to update a connection is to manually perform an import using the Data Access window in the upgraded version of Analytics. Copy the ACCESSDATA command from the log and use it to update your script.

  > **Note**
  > Before connecting to a data source and re-running the import, clear the connector cache to flush the existing set of table names.
  > In the **Existing Connections** tab in the Data Access window, beside the connector name, select ··· **> Clear cache**.

- **Update field specifications** – You may also need to update field specifications in the script body to align with table schema changes in the data source or ODBC driver. Possible changes include field names, field data types, and field and record lengths.
- **Check the results of any filtering** – You should also check the results of any filtering that you apply as part of the data import. Confirm that the import filtering is including and excluding records correctly.

Connecting to Concur

Concur is a cloud-based travel and expense management service. Use the Concur data connector to import your company's Concur data.

> **Note**
> Analytics provides Concur as an optional connector and if it is not available in your Data Access window, it is likely that the connector was not selected during installation. For more information, see "Install optional Analytics data connectors and Python engine" on page 2567.

Before you start

Concur credentials

To connect to Concur, you need the following:

- Concur Access Token

If you do not have a Concur Access Token, you must gather the following:

- Concur user name and password
- Concur Client ID
- Concur API server host name (default is "www.concursolutions.com")

With these credentials, you can use the **Get Token** option in the Data Access window to generate the access token required to authenticate your connection.

> **Note**
> Make note of your token's expiry date when you obtain it. Concur authentication tokens expire after one year.

For help gathering the connection prerequisites, contact the Concur administrator in your organization. If your administrator cannot help you, you or your administrator should contact Concur Support.

Acquire a Concur Client ID

If you do not have a Concur Access Token, you or your company's Concur administrator needs to acquire a Concur Client ID. You can use the Client ID to generate the Access Token.

You acquire a Client ID in the Administration area of Concur Web Services through the Register Partner Application option.

In Concur Web Services, the Client ID is called either:

- Application Authorization Key
- Consumer Key

> **Note**
> If the Register Partner Application option does not appear in Concur Web Services, you or your Concur administrator needs to contact Concur Support. Your company may not have the appropriate Concur license.

Create a Concur connection

1. From the Analytics main menu, select **Import > Database and application**.
2. From the **New Connections** tab, in the **ACL Connectors** section, select **Concur.**

   > **Tip**
   > You can filter the list of available connectors by entering a search string in the **Filter connections** box. Connectors are listed alphabetically.

3. In the **Data Connection Settings** panel, enter the connection settings and at the bottom of the panel, click **Save and Connect**.

   You can accept the default **Connection Name**, or enter a new one.

The connection for Concur is saved to the **Existing Connections** tab. In the future, you can reconnect to Concur from the saved connection.

Once the connection is established, the Data Access window opens to the **Staging Area** and you can begin importing data. For help importing data from Concur, see "Working with the Data Access window" on page 357.

Connection settings

Basic settings

Setting	Description	Example
Host	Host name of the Concur server.	https://concursolutions.com
OAuth Client Id	The client ID assigned when you register your application with an OAuth authorization server.	
Username	The user account used to authenticate to Concur.	

Setting	Description	Example
Password	The password to authenticate the user to Concur.	
Access Token	The access token for authenticated users to access protected resources on Concur services.	

Advanced settings

Setting	Description	Example
Enable Double Buffering	Specifies whether the adapter retrieves Concur data using double-buffering. However, when double-buffering is enabled, the adapter might consume more memory and resources.	
Use HTTPS	Specifies whether the data source endpoints are encrypted using HTTPS.	
User Param	The value to use for the user query parameter, when querying tables with endpoints that are under the `/api/v3.0/expense` endpoint group.	all
Use Windows Proxy Settings	Specifies whether to use the system proxy settings or not. To use custom proxy settings, you must disable this option. The value set for this option takes precedence over other proxy settings.	
Proxy Host	The hostname or IP address of a proxy to route HTTP traffic through.	206.174.193.118
Proxy Port	The TCP port the ProxyServer proxy is running on.	80
Proxy User	The user name to authenticate to the ProxyServer proxy.	user@domain
Proxy Password	The password for the Proxy user to authenticate to the ProxyServer proxy.	

Data you can access in Concur

The Data Access window allows you to access a subset of the data in Concur. Data is available from the Concur APIs listed below.

> **Note**
>
> If an API is not listed below, you cannot access it using the Data Access window. For information about the data available from each endpoint, see the Concur API documentation.
>
> Concur stores data in structures which do not follow the rules of data typing and structure that apply to traditional relational tables and columns. Therefore, the data needs to be mapped to a relational form. To achieve this, the connector maps the Concur data to an ODBC-compatible format.

Module	API Endpoint
Common	Lists
	List Items
	Locations
Expense	Entries
	Expense Form
	Expense Group Configuration
	Itemizations
	Quick Expense
	Reports
Insights	Opportunities
Receipt Image	Receipt Image
Travel	Trips

Accessing expense data for multiple users

> **Note**
>
> To access expense data for multiple users, the Concur account that you use to connect to Concur must have the appropriate permissions.
>
> If you need help with Concur account permissions, contact the Concur administrator in your company. If your administrator cannot help you, you or your administrator should contact Concur Support.

You can access data from the Expense module for multiple Concur accounts by setting the optional **User Param** field in the **Advanced Options** to All.

You can also enter the user name of a specific Concur user in this field to view expense data for an individual.

Connecting to Couchbase

Couchbase is a NoSQL document-oriented database. Use the Couchbase data connector to import your company's Couchbase data.

> **Note**
> Analytics provides Couchbase as an optional connector and if it is not available in your Data Access window, it is likely that the connector was not selected during installation. For more information, see "Install optional Analytics data connectors and Python engine" on page 2567.

Before you start

To connect to Couchbase, you must gather the following:

- at least one node in your Couchbase instance with the query and index services enabled
- the database server's host name
- the correct connection port
- appropriate credentials for your chosen authentication method

  If your Couchbase instance requires authentication, you must provide either a JSON string or a JSON file that specifies the name and password for one or more buckets. For more information, see "JSON authentication format" on page 437.

- a valid SSL certificate if connecting over SSL

For help gathering the connection prerequisites, contact the Couchbase administrator in your organization. If your administrator cannot help you, you or your administrator should contact Couchbase Support.

Create a Couchbase connection

1. From the Analytics main menu, select **Import > Database and application**.
2. From the **New Connections** tab, in the **ACL Connectors** section, select **Couchbase.**

   > **Tip**
   > You can filter the list of available connectors by entering a search string in the **Filter connections** box. Connectors are listed alphabetically.

3. In the **Data Connection Settings** panel, enter the connection settings and at the bottom of the

panel, click **Save and Connect**.

You can accept the default **Connection Name**, or enter a new one.

The connection for Couchbase is saved to the **Existing Connections** tab. In the future, you can reconnect to Couchbase from the saved connection.

Once the connection is established, the Data Access window opens to the **Staging Area** and you can begin importing data. For help importing data from Couchbase, see "Working with the Data Access window" on page 357.

Connection settings

Setting	Description	Example
Server	The host name or IP address of the Couchbase server. This value can be set to an HTTP or HTTPS URL.	couchbase-server.com
Port	The port the Couchbase server is running on.	8093
Authentication Mechanism	Authentication mechanism to use when connecting to Couchbase server. ◦ **No Authentication** - Driver does not authenticate the connection. ◦ **Use Credentials** - Driver authenticates the connection using the names and passwords specified in the credentials string. ◦ **Use Credential File** - Driver authenticates the connection using the names and passwords specified in the credentials file.	
Credentials	A JSON string that specifies the name and password of one or more users or buckets, for authenticating to a Couchbase server instance. The credentials string must use the following format: `[{"user": "[UserName1]", "pass":"[Password1]"}, {"user": "[UserName2]", "pass":"[Password2]"}]`	
Credentials File	Full path to a JSON file that contains credentials for authenticating to a Couchbase server	

Setting	Description	Example
	instance. The JSON file must contain the names and passwords of multiple users or buckets.	
Enable SSL	Specifies whether to use SSL when connecting to the Couchbase server.	false
SSL Certificate	The certificate to be accepted from the server when **Enable SSL** is set to true. You can provide any of the following: ◦ a full PEM certificate ◦ path to a local file containing the certificate ◦ public key ◦ MD5 or SHA1 thumbprint (hex values can also be either space or colon separated) Any other certificate that is not trusted by the machine is rejected.	C:\cert.cer

Advanced settings

Setting	Description	Example
Query Mode	Specifies the query mode to use when sending queries to Couchbase server. ◦ **SQL**- The driver executes all queries in SQL. ◦ **N1QL** - The driver executes all queries in N1QL.	SQL
Consistency	The level of data consistency to enforce during index scans. Set this property to one of the following values: ◦ **NOT_BOUNDED** ◦ **AT_PLUS** ◦ **REQUEST_PLUS** ◦ **STATEMENT_PLUS**	REQUEST_PLUS

Setting	Description	Example
Enable Load Balancing	Specifies whether the driver supports load balancing and failover between nodes in a Couchbase cluster.	
Sample Size	The number of documents that the driver samples to detect the structure of the data when generating a schema definition using the `SchemaMapOperation` property.	100
Type Name List	A comma-separated list of the attributes that the buckets use to specify document types. Each list item must be a bucket name surrounded by back quotes (`), a colon (:), and an attribute name surrounded by back quotes (`).	`` `product`:`type`,`store`:`type`, `customer`:`jsonType`,`sales`:`type` ``

Querying the Couchbase instance

JSON authentication format

```
[{"user" : "userName1", "pass" : "passWord1"},
{"user" : "userName2", "pass" : "passWord2"}]
```

SQL queries vs N1QL API calls

The connector uses normal SQL queries against the Couchbase Server, translating standard SQL-92 queries into equivalent N1QL client API calls. This translation allows standard queries to run against your Couchbase Server instance. If a query cannot be fully translated, then the translated parts of the query are passed down as one or more N1QL queries to the Couchbase Server instance for processing while the untranslated parts of the query are processed by the Connector.

> **Note**
> The names of data structures are case-sensitive, therefore you must ensure the casing of structures such as tables, columns, or buckets in your queries match the structures in the database.

Schema definition

Couchbase is able to store data that follows different rules of data typing and structure compared to traditional relational tables and columns. Couchbase data is organized into buckets and documents, which can contain nested arrays or arrays of differently typed elements. This data needs to be mapped to a relational form. To achieve this, the connector generates a schema definition that maps the Couchbase data to an ODBC-compatible format.

When you connect to a database that does not already have the necessary schema definition, the connector automatically generates one by doing the following:

1. For each document type identified in the database, the connector samples data from multiple documents to detect the structure of the data.
2. The connector organizes all the documents into collections based on their type, and saves these collections as part of the schema definition. Using the schema definition, the driver exposes collections as tables.
3. For each array detected in the database, the connector generates a virtual table to expand the data, and saves these virtual tables as part of the schema. Using the schema, the driver exposes virtual tables as normal tables.
4. The connector defines a Couchbase data type for each column and maps each Couchbase data type to the SQL data type that is best able to represent the greatest number of values.

Base tables

Base tables represent data from collections of Couchbase documents. Documents appear as rows, and all attributes that are not arrays appear as columns. In each base table, the connector creates a primary key column named **PK** that identifies which Couchbase document each row comes from.

In the connector, the name of the base table is the document type that it represents. In Couchbase, the name of the base table is the bucket that the data comes from.

Virtual tables

Virtual tables provide support for arrays. Each virtual table contains the data from one array, and each row in the table represents an element from the array. If an element contains an array, then the connector creates additional virtual tables as needed to expand the nested data.

In each virtual table, the connector creates a primary key column name that identifies the document that the array comes from and references the column from the related base table. The connector also creates an index column (with the suffix **_IDX** in its name) to indicate the position of the element within the array.

Example

The following example shows the base tables and virtual tables that the connector would generate if it connected to a Couchbase database named **ExampleDatabase**, which contains two documents named **Customer_123221** and **Order_221354**.

The **Customer_123221** document is of type Customer and contains the following attributes. The **SavedAddresses** attribute is an array:

```
{
  "Type": "Customer",
  "Name": "John Doe",
  "SavedAddresses": ["123 Main St.", "456 1st Ave"]
}
```

The **Order_221354** document is of type Order and contains the following attributes. The **CreditCard** attribute is an object, and the **Items** attribute is an array of objects:

```
{
  "Type": "Order",
  "CustomerID":"Customer_123221",
  "CreditCard":
    {
      "Type":"Visa",
       "CardNumber":"4111 1111 1111 1111",
       "Expiry":"12/12",
       "CVN":"123"
    },
  "Items":
    [
      {"ItemID":89123, "Quantity":1},
      {"ItemID":92312, "Quantity":5}
    ]
}
```

When Analytics connects to **ExampleDatabase** and generates the schema, the connector creates a collection for each document type and exposes these collections as two base tables, which are shown below:

Base table Customer

PK	Name
"Customer_123221"	John Doe

Base table Order

PK	Customer ID	CreditCard _Type	CreditCard _Number	CreditCard _Expiry	CreditCard _CVN
"Order_ 221354"	"Customer_ 123221"	"Visa"	"4111 1111 1111 1111"	"12/12"	"123"

The **SavedAddresses** array from the **Customer_123221** document and the **Items** array from the **Order_221354** document do not appear in these base tables. Instead, the connector generates a virtual table for each array:

SavedAddresses table

PK	SavedAddresses_IDX	SavedAddresses
"Customer_123221"	0	"123 Main St."
"Customer_123221"	1	"456 1st Ave"

Items table

PK	Items_IDX	ItemID	Quantity
"Order_221354"	0	89123	1
"Order_221354"	1	92312	5

Connecting to DocuSign

DocuSign is an electronic agreements management tool.You can use the DocuSign data connector to import your organization's DocuSign data.

> **Note**
> The DocuSign data connector is provided by our data partner, CData. For information on any of the connection fields, refer the documentation available at the CData website.

Before you start

To connect to DocuSign, you must gather the credentials to connect to the DocuSign server.

For help gathering the connection prerequisites, contact the DocuSign administrator in your organization. If your administrator cannot help you, you or your administrator should contact DocuSign Support.

Create a DocuSign connection

1. From the Analytics main menu, select **Import > Database and application**.
2. From the **New Connections** tab, in the **ACL DSN Connectors (Bundled)** section, select **DocuSign.**

   > **Tip**
   > You can filter the list of available connectors by entering a search string in the **Filter connections** box. Connectors are listed alphabetically.

   The DSN Configuration dialog box appears.
3. In the **DSN Configuration** dialog box, click **Show Required**.
4. Provide values for the required fields, if any.
5. Click **Test Connection**.

   > **Note**
   > When you click the **Test Connection** button, it validates whether the connection and authentication details provided are correct. If you click **OK** without clicking Test Connection, the connection details are saved without any validation and may not work later if the values provided are incorrect.

The log in page for DocuSign appears.

6. Provide the connection details and authenticate your login.

   When the connection is successful, a dialog box opens displaying the success message.

7. Click **OK** in the connection successful dialog box that appears.

   The connection to DocuSign is established successfully.

8. In the **DSN Configuration** dialog box, click **OK**.

The connection for DocuSign is saved to the **Existing Connections** tab. In the future, you can reconnect to DocuSign from the saved connection.

Once the connection is established, the Data Access window opens to the **Staging Area** and you can begin importing data. For help importing data from DocuSign, see "Working with the Data Access window" on page 357.

Rename the connection

When you create the DocuSign data connector, you cannot provide a Connection name of your choice and the connection is saved to the **Existing Connections** tab with the same name as your data source. If you want to rename it, navigate to the **Existing Connections** tab, expand the **ACL Connectors (DSN)** section, click the ellipsis icon corresponding to DocuSign and select **Rename connection**.

Connecting to Dynamics CRM

Microsoft Dynamics CRM is a customer relationship management (CRM) software for managing an organization customer base. The tools enables users to organize, automate, and synchronize sales, marketing, customer service, and technical support. You can use the Dynamics CRM data connector to import your organization's Dynamics CRM data.

> **Note**
> The Dynamics CRM data connector is provided by our data partner, CData. For information on any of the connection fields, refer the documentation available at the CData website.

Before you start

To connect to Dynamics CRM, you must gather the following information:

- the root URL of your organization
- user credentials
- the type of Dynamics CRM server to which you are connecting

For help gathering the connection prerequisites, contact the Dynamics CRM administrator in your organization. If your administrator cannot help you, you or your administrator should contact Dynamics CRM Support.

Create a Dynamics CRM connection

1. From the Analytics main menu, select **Import > Database and application**.
2. From the **New Connections** tab, in the **ACL DSN Connectors (Bundled)** section, select **Dynamics CRM.**

   > **Tip**
   > You can filter the list of available connectors by entering a search string in the **Filter connections** box. Connectors are listed alphabetically.

   The DSN Configuration dialog box appears.
3. In the **DSN Configuration** dialog box, click **Show Required**.
4. Provide values for the required fields, if any.

5. Click **Test Connection**.

> **Note**
> When you click the **Test Connection** button, it validates whether the connection and authentication details provided are correct. If you click **OK** without clicking Test Connection, the connection details are saved without any validation and may not work later if the values provided are incorrect.

   The log in page for Dynamics CRM appears.

6. Provide the connection details and authenticate your login.

   When the connection is successful, a dialog box opens displaying the success message.

7. Click **OK** in the connection successful dialog box that appears.

   The connection to Dynamics CRM is established successfully.

8. In the **DSN Configuration** dialog box, click **OK**.

The connection for Dynamics CRM is saved to the **Existing Connections** tab. In the future, you can reconnect to Dynamics CRM from the saved connection.

Once the connection is established, the Data Access window opens to the **Staging Area** and you can begin importing data. For help importing data from Dynamics CRM, see "Working with the Data Access window" on page 357.

Rename the connection

When you create the Dynamics CRM data connector, you cannot provide a Connection name of your choice and the connection is saved to the **Existing Connections** tab with the same name as your data source. If you want to rename it, navigate to the **Existing Connections** tab, expand the **ACL Connectors (DSN)** section, click the ellipsis icon corresponding to Dynamics CRM and select **Rename connection**.

Connecting to Dynamics GP

Microsoft Dynamics GP is a business accounting software to manage finance, inventory, and operations of organizations. You can use the Dynamics GP data connector to import your organization's Dynamics GP data.

> **Note**
> The Dynamics GP data connector is provided by our data partner, CData. For information on any of the connection fields, refer the documentation available at the CData website.

Before you start

To connect to Dynamics GP, you must gather the following information:

- the unique identifier of your organization
- the URL of the Dynamics GP server
- user credentials to connect to the Dynamics GP server

For help gathering the connection prerequisites, contact the Dynamics GP administrator in your organization. If your administrator cannot help you, you or your administrator should contact Dynamics GP Support.

Create a Dynamics GP connection

1. From the Analytics main menu, select **Import > Database and application**.
2. From the **New Connections** tab, in the **ACL DSN Connectors (Bundled)** section, select **Dynamics GP.**

   > **Tip**
   > You can filter the list of available connectors by entering a search string in the **Filter connections** box. Connectors are listed alphabetically.

   The DSN Configuration dialog box appears.
3. In the **DSN Configuration** dialog box, click **Show Required**.
4. Provide values for the required fields, if any.

5. Click **Test Connection**.

> **Note**
> When you click the **Test Connection** button, it validates whether the connection and authentication details provided are correct. If you click **OK** without clicking Test Connection, the connection details are saved without any validation and may not work later if the values provided are incorrect.

   The log in page for Dynamics GP appears.

6. Provide the connection details and authenticate your login.

   When the connection is successful, a dialog box opens displaying the success message.

7. Click **OK** in the connection successful dialog box that appears.

   The connection to Dynamics GP is established successfully.

8. In the **DSN Configuration** dialog box, click **OK**.

The connection for Dynamics GP is saved to the **Existing Connections** tab. In the future, you can reconnect to Dynamics GP from the saved connection.

Once the connection is established, the Data Access window opens to the **Staging Area** and you can begin importing data. For help importing data from Dynamics GP, see "Working with the Data Access window" on page 357.

Rename the connection

When you create the Dynamics GP data connector, you cannot provide a Connection name of your choice and the connection is saved to the **Existing Connections** tab with the same name as your data source. If you want to rename it, navigate to the **Existing Connections** tab, expand the **ACL Connectors (DSN)** section, click the ellipsis icon corresponding to Dynamics GP and select **Rename connection**.

Connecting to Dynamics NAV

Microsoft Dynamics NAV is an enterprise resource planning (ERP) solution that helps businesses to automate sales, purchases, operations, accounting, and manage inventory. You can use the Dynamics NAV data connector to import your organization's Dynamics NAV data.

> **Note**
> The Dynamics NAV data connector is provided by our data partner, CData. For information on any of the connection fields, refer the documentation available at the CData website.

Before you start

To connect to Dynamics NAV, you must gather the following information:

- the root URL of the Dynamics NAV server
- the instance of the Dynamics NAV server
- user credentials to connect to the Dynamics NAV server
- the authentication scheme used for connecting to the server

For help gathering the connection prerequisites, contact the Dynamics NAV administrator in your organization. If your administrator cannot help you, you or your administrator should contact Dynamics NAV Support.

Create a Dynamics NAV connection

1. From the Analytics main menu, select **Import > Database and application**.
2. From the **New Connections** tab, in the **ACL DSN Connectors (Bundled)** section, select **Dynamics NAV.**

   > **Tip**
   > You can filter the list of available connectors by entering a search string in the **Filter connections** box. Connectors are listed alphabetically.

   The DSN Configuration dialog box appears.
3. In the **DSN Configuration** dialog box, click **Show Required**.
4. Provide values for the required fields, if any.

5. Click **Test Connection**.

> **Note**
> When you click the **Test Connection** button, it validates whether the connection and authentication details provided are correct. If you click **OK** without clicking Test Connection, the connection details are saved without any validation and may not work later if the values provided are incorrect.

   The log in page for Dynamics NAV appears.

6. Provide the connection details and authenticate your login.

   When the connection is successful, a dialog box opens displaying the success message.

7. Click **OK** in the connection successful dialog box that appears.

   The connection to Dynamics NAV is established successfully.

8. In the **DSN Configuration** dialog box, click **OK**.

The connection for Dynamics NAV is saved to the **Existing Connections** tab. In the future, you can reconnect to Dynamics NAV from the saved connection.

Once the connection is established, the Data Access window opens to the **Staging Area** and you can begin importing data. For help importing data from Dynamics NAV, see "Working with the Data Access window" on page 357.

Rename the connection

When you create the Dynamics NAV data connector, you cannot provide a Connection name of your choice and the connection is saved to the **Existing Connections** tab with the same name as your data source. If you want to rename it, navigate to the **Existing Connections** tab, expand the **ACL Connectors (DSN)** section, click the ellipsis icon corresponding to Dynamics NAV and select **Rename connection**.

Connecting to Dynamics 365 Business Central

Microsoft Dynamics 365 Business Central is an enterprise resource planning (ERP) system. You can use the Dynamics 365 Business Central data connector to import your organization's Dynamics 365 Business Central data.

> **Note**
> The Dynamics 365 Business Central data connector is provided by our data partner, CData. For information on any of the connection fields, refer the documentation available at the CData website.

Before you start

To connect to Dynamics 365 Business Central, you must gather the following information:

- name of the Company being used for Dynamics 365 Business Central
- URL to your Dynamics 365 organization
- the Microsoft Online tenant to access data

For help gathering the connection prerequisites, contact the Dynamics 365 Business Central administrator in your organization. If your administrator cannot help you, you or your administrator should contact Dynamics 365 Business Central Support.

Create a Dynamics 365 Business Central connection

1. From the Analytics main menu, select **Import > Database and application**.
2. From the **New Connections** tab, in the **ACL DSN Connectors (Bundled)** section, select **Dynamics 365 Business Central.**

   > **Tip**
   > You can filter the list of available connectors by entering a search string in the **Filter connections** box. Connectors are listed alphabetically.

   The DSN Configuration dialog box appears.
3. In the **DSN Configuration** dialog box, click **Show Required**.

4. Provide values for the required fields, if any.
5. Click **Test Connection**.

> **Note**
> When you click the **Test Connection** button, it validates whether the connection and authentication details provided are correct. If you click **OK** without clicking Test Connection, the connection details are saved without any validation and may not work later if the values provided are incorrect.

   The log in page for Dynamics 365 Business Central appears.

6. Provide the connection details and authenticate your login.

   When the connection is successful, a dialog box opens displaying the success message.

7. Click **OK** in the connection successful dialog box that appears.

   The connection to Dynamics 365 Business Central is established successfully.

8. In the **DSN Configuration** dialog box, click **OK**.

The connection for Dynamics 365 Business Central is saved to the **Existing Connections** tab. In the future, you can reconnect to Dynamics 365 Business Central from the saved connection.

Once the connection is established, the Data Access window opens to the **Staging Area** and you can begin importing data. For help importing data from Dynamics 365 Business Central, see "Working with the Data Access window" on page 357.

Rename the connection

When you create the Dynamics 365 Business Central data connector, you cannot provide a Connection name of your choice and the connection is saved to the **Existing Connections** tab with the same name as your data source. If you want to rename it, navigate to the **Existing Connections** tab, expand the **ACL Connectors (DSN)** section, click the ellipsis icon corresponding to Dynamics 365 Business Central and select **Rename connection**.

Connecting to Dynamics 365 Finance and Operations

Microsoft Dynamics 365 Finance and Operations is an enterprise resource planning (ERP) system for organizations to automate your warehouse processes to reduce operational costs. You can use the Dynamics 365 Finance and Operations data connector to import your organization's Dynamics 365 Finance and Operations data.

> **Note**
> The Dynamics 365 Finance and Operations data connector is provided by our data partner, CData. For information on any of the connection fields, refer the documentation available at the CData website.

Before you start

To connect to Dynamics 365 Finance and Operations, you must gather the following information:

- URL to your Dynamics 365 organization
- the Microsoft Online tenant to access data
- whether to import data from all companies or only the default company

For help gathering the connection prerequisites, contact the Dynamics 365 Finance and Operations administrator in your organization. If your administrator cannot help you, you or your administrator should contact Dynamics 365 Finance and Operations Support.

Create a Dynamics 365 Finance and Operations connection

1. From the Analytics main menu, select **Import > Database and application**.
2. From the **New Connections** tab, in the **ACL DSN Connectors (Bundled)** section, select **Dynamics 365 Finance and Operations.**

   > **Tip**
   > You can filter the list of available connectors by entering a search string in the **Filter connections** box. Connectors are listed alphabetically.

   The DSN Configuration dialog box appears.

3. In the **DSN Configuration** dialog box, click **Show Required**.
4. Provide values for the required fields, if any.
5. Click **Test Connection**.

> **Note**
> When you click the **Test Connection** button, it validates whether the connection and authentication details provided are correct. If you click **OK** without clicking Test Connection, the connection details are saved without any validation and may not work later if the values provided are incorrect.

   The log in page for Dynamics 365 Finance and Operations appears.

6. Provide the connection details and authenticate your login.

   When the connection is successful, a dialog box opens displaying the success message.

7. Click **OK** in the connection successful dialog box that appears.

   The connection to Dynamics 365 Finance and Operations is established successfully.

8. In the **DSN Configuration** dialog box, click **OK**.

The connection for Dynamics 365 Finance and Operations is saved to the **Existing Connections** tab. In the future, you can reconnect to Dynamics 365 Finance and Operations from the saved connection.

Once the connection is established, the Data Access window opens to the **Staging Area** and you can begin importing data. For help importing data from Dynamics 365 Finance and Operations, see "Working with the Data Access window" on page 357.

Rename the connection

When you create the Dynamics 365 Finance and Operations data connector, you cannot provide a Connection name of your choice and the connection is saved to the **Existing Connections** tab with the same name as your data source. If you want to rename it, navigate to the **Existing Connections** tab, expand the **ACL Connectors (DSN)** section, click the ellipsis icon corresponding to Dynamics 365 Finance and Operations and select **Rename connection**.

Connecting to Dynamics 365 Sales

Microsoft Dynamics 365 Sales is an adaptive sales accelerator solution that enables sales teams to manage customer relationships, boost sales revenue, and close faster. You can use the Dynamics 365 Sales data connector to import your organization's Dynamics 365 Sales data.

> **Note**
> The Dynamics 365 Sales data connector is provided by our data partner, CData. For information on any of the connection fields, refer the documentation available at the CData website.

Before you start

To connect to Dynamics 365 Sales, you must gather the following information:

- URL to your Dynamics 365 organization
- the Microsoft Online tenant to access data

For help gathering the connection prerequisites, contact the Dynamics 365 Sales administrator in your organization. If your administrator cannot help you, you or your administrator should contact Dynamics 365 Sales Support.

Create a Dynamics 365 Sales connection

1. From the Analytics main menu, select **Import > Database and application**.
2. From the **New Connections** tab, in the **ACL DSN Connectors (Bundled)** section, select **Dynamics 365 Sales.**

   > **Tip**
   > You can filter the list of available connectors by entering a search string in the **Filter connections** box. Connectors are listed alphabetically.

   The DSN Configuration dialog box appears.
3. In the **DSN Configuration** dialog box, click **Show Required**.
4. Provide values for the required fields, if any.

5. Click **Test Connection**.

> **Note**
> When you click the **Test Connection** button, it validates whether the connection and authentication details provided are correct. If you click **OK** without clicking Test Connection, the connection details are saved without any validation and may not work later if the values provided are incorrect.

   The log in page for Dynamics 365 Sales appears.

6. Provide the connection details and authenticate your login.

   When the connection is successful, a dialog box opens displaying the success message.

7. Click **OK** in the connection successful dialog box that appears.

   The connection to Dynamics 365 Sales is established successfully.

8. In the **DSN Configuration** dialog box, click **OK**.

The connection for Dynamics 365 Sales is saved to the **Existing Connections** tab. In the future, you can reconnect to Dynamics 365 Sales from the saved connection.

Once the connection is established, the Data Access window opens to the **Staging Area** and you can begin importing data. For help importing data from Dynamics 365 Sales, see "Working with the Data Access window" on page 357.

Rename the connection

When you create the Dynamics 365 Sales data connector, you cannot provide a Connection name of your choice and the connection is saved to the **Existing Connections** tab with the same name as your data source. If you want to rename it, navigate to the **Existing Connections** tab, expand the **ACL Connectors (DSN)** section, click the ellipsis icon corresponding to Dynamics 365 Sales and select **Rename connection**.

Connecting to Edgar Online

Edgar Online is a solution that creates and distributes financial data and public filings for equities, mutual funds, and other publicly traded assets. You can use the Edgar Online data connector to import data your organization's Edgar Online data.

> **Note**
> The Edgar Online data connector is provided by our data partner, CData. For information on any of the connection fields, refer the documentation available at the CData website.

Before you start

To connect to Edgar Online, you must gather the AppKey of the currently authenticated user to connect to the Edgar Online account.

For help gathering the connection prerequisites, contact the Edgar Online administrator in your organization. If your administrator cannot help you, you or your administrator should contact Edgar Online Support.

Create an Edgar Online connection

1. From the Analytics main menu, select **Import > Database and application**.
2. From the **New Connections** tab, in the **ACL DSN Connectors (Bundled)** section, select **Edgar Online.**

   > **Tip**
   > You can filter the list of available connectors by entering a search string in the **Filter connections** box. Connectors are listed alphabetically.

   The DSN Configuration dialog box appears.
3. In the **DSN Configuration** dialog box, click **Show Required**.
4. Provide values for the required fields, if any.

5. Click **Test Connection**.

> **Note**
> When you click the **Test Connection** button, it validates whether the connection and authentication details provided are correct. If you click **OK** without clicking Test Connection, the connection details are saved without any validation and may not work later if the values provided are incorrect.

   The log in page for Edgar Online appears.

6. Provide the connection details and authenticate your login.

   When the connection is successful, a dialog box opens displaying the success message.

7. Click **OK** in the connection successful dialog box that appears.

   The connection to Edgar Online is established successfully.

8. In the **DSN Configuration** dialog box, click **OK**.

The connection for Edgar Online is saved to the **Existing Connections** tab. In the future, you can reconnect to Edgar Online from the saved connection.

Once the connection is established, the Data Access window opens to the **Staging Area** and you can begin importing data. For help importing data from Edgar Online, see "Working with the Data Access window" on page 357.

Rename the connection

When you create the Edgar Online data connector, you cannot provide a Connection name of your choice and the connection is saved to the **Existing Connections** tab with the same name as your data source. If you want to rename it, navigate to the **Existing Connections** tab, expand the **ACL Connectors (DSN)** section, click the ellipsis icon corresponding to Edgar Online and select **Rename connection**.

Connecting to Email

Use the Email data connector to import email messages for a single account using the standard mail protocols IMAP or POP. When you connect to your email server, each table name represents a mailbox folder on the server, and each record represents an email message.

> **Note**
> You cannot use this connector independently of Analytics. You can configure a DSN for the connector driver using the Windows **ODBC Data Source Administrator**, however you must test the DSN connection from within Analytics and not from the connector's Windows DSN configuration dialog.

This connector retrieves email messages for a single account stored on mail servers only. It does not connect to features such as chat, or to-dos that are included with some email clients.

Before you start

To connect to Email, you must gather the following:

- the domain name or IP address of the Email server
- the correct connection port
- the connecting user account, including the user name and password

> **Note**
> Your email server must use either the IMAP or POP protocol.

For help gathering the connection prerequisites, contact the Email administrator in your organization. If your administrator cannot help you, you or your administrator should contact Email Support.

Create an Email connection

1. From the Analytics main menu, select **Import > Database and application**.
2. From the **New Connections** tab, in the **ACL Connectors** section, select **Email.**

> **Tip**
> You can filter the list of available connectors by entering a search string in the **Filter connections** box. Connectors are listed alphabetically.

3. In the **Data Connection Settings** panel, enter the connection settings and at the bottom of the panel, click **Save and Connect**.

   You can accept the default **Connection Name**, or enter a new one.

The connection for Email is saved to the **Existing Connections** tab. In the future, you can reconnect to Email from the saved connection.

Once the connection is established, the Data Access window opens to the **Staging Area** and you can begin importing data. For help importing data from Email, see "Working with the Data Access window" on page 357.

Connection settings

Basic settings

Setting	Description	Example
Protocol	The type of email server to connect to: ◦ IMAP ◦ POP	IMAP
User	The user of the Email account used to authenticate. Together with Password, this field is used to authenticate to the email servers.	recipient@example.com
Password	The password of the email account used to authenticate. Together with User, this field is used to authenticate to the email servers.	
Server	The name or address of the mail server. This property specifies the IP address or the domain name of the mail server. It must be set before a connection is attempted and cannot be changed once a connection is in progress.	imap.gmail.com
Port	The port of the mail server. The default value is: ◦ **IMAP** - 143 (non-SSL) or 993 (SSL) ◦ **POP** - 110 (non-SSL) or 995 (SSL) A valid port number (a value	993

Setting	Description	Example
	between 1 and 65535) is required for the connection to take place. The property must be set before a connection is attempted and cannot be changed once a connection is established.	

Advanced settings

Setting	Description	Example
Convert Datetime to GMT	Converts datetime fields to GMT time zone during import. If false, the datetime value is converted to the operating system time zone of the machine running Analytics.	true
Email Service	The name of the email service you are using. Optimizes the IMAP connection for the service you are working with. You may use one of the following: ◦ AOL ◦ Gmail ◦ Outlook ◦ Yahoo ◦ Other **Note** Using the Email connector, you cannot retrieve the contents of the **Subject** field for the Task or Calendar mailboxes from an Outlook 365 account. If you are connecting to an Outlook 365 account, consider using the Exchange connector instead. For more information, see "Connecting to Exchange" on page 471	Other
Include Message	Whether to include message body content and attachment data or not.	false

Setting	Description	Example
	Caution This setting affects performance and may cause your query to timeout if you are working with many records.	
Is HTML	Determines whether the Message Body is HTML or plain-text.	true
Keep Alive	Determines whether to keep the connection alive across instances.	true
List Mailboxes	Whether to list all mailboxes or just the subscribed IMAP mailboxes. IMAP Only: ◦ All ◦ Subscribed	All
Limit Key Size	The maximum length of a primary key column. Setting the size to 0 will make the key length revert to the original length. This property makes the connector override the reported length of all the primary key columns.	255
Map to Long Varchar	Controls whether or not a column is returned as SQL_LONGVARCHAR. Use this setting to map any column larger than the specified size so they are reported as SQL_LONGVARCHAR instead of SQL_VARCHAR.	-1
Map To WVarchar	Controls whether or not string types map to SQL_WVARCHAR instead of SQL_VARCHAR. It is set by default. String columns must be mapped to SQL_WVARCHAR to accommodate various international character sets, so MapToWVarchar is set to true by default. You may set it to false to use SQL_VARCHAR instead.	true
Max Items	Maximum number of items to return.	-1

Setting	Description	Example
	The default value is -1. This value ensures that all items are returned.	
Pseudo Columns	Indicates whether or not to include pseudo columns as columns to the table. This setting is particularly helpful in Entity Framework, which does not allow you to set a value for a pseudo column unless it is a table column. The value of this connection setting is of the format "Table1=Column1, Table1=Column2, Table2=Column3". You can use the "*" character to include all tables and all columns.	MyTable=*
SSL Server Cert	The certificate to be accepted from the server when connecting using TLS/SSL. You can specify any of the following: ◦ full PEM certificate ◦ path to a local file containing the certificate ◦ the public key ◦ the MD5 Thumbprint (hex values can also be either space or colon separated) ◦ the SHA1 Thumbprint (hex values can also be either space or colon separated) If using a TLS/SSL connection, this property can be used to specify the TLS/SSL certificate to be accepted from the server. Any other certificate that is not trusted by the machine will be rejected.	C:\cert.cer
Support Enhanced SQL	Enhances SQL functionality beyond what can be supported through the API directly, by enabling in-memory client-side processing: ◦ **true** - the connector offloads as much of the SELECT statement processing as possible to IMAP and then processes the rest of the query in memory. In this way the driver can execute unsupported predicates, joins,	

Setting	Description	Example
	and aggregation ◦ **false** - the connector limits SQL execution to what is supported by the IMAP API ## Execution of predicates The connector determines which of the clauses are supported by the data source and then pushes them to the source to get the smallest superset of rows that would satisfy the query. It then filters the rest of the rows locally. The filter operation is streamed, which enables the driver to filter effectively for even very large datasets. ## Execution of Joins The connector uses various techniques to join in memory. The driver trades off memory utilization against the requirement of reading the same table more than once. ## Execution of Aggregates The connector retrieves all rows necessary to process the aggregation in memory.	
UID Mode	If true, permanent message Ids are used instead of the default temporary Ids.	false
Upper Case Identifiers	Report all identifiers in uppercase, including table and column names.	false
Proxy Authentication Scheme	The authentication type to use to authenticate to the ProxyServer proxy. This value specifies the authentication type to use to authenticate to the HTTP proxy specified by ProxyServer and ProxyPort.	BASIC

Setting	Description	Example
	Note The connector will use the system proxy settings by default, without further configuration needed; if you want to connect to another proxy, you will need to set ProxyAutoDetect to false, in addition to ProxyServer and ProxyPort. To authenticate, set ProxyAuthScheme and set ProxyUser and ProxyPassword, if needed. The authentication type can be one of the following: ◦ **BASIC** - The driver performs HTTP BASIC authentication ◦ **DIGEST** - The driver performs HTTP DIGEST authentication. ◦ **NEGOTIATE** - The driver retrieves an NTLM or Kerberos token based on the applicable protocol for authentication ◦ **PROPRIETARY** - The driver does not generate an NTLM or Kerberos token. You must supply this token in the Authorization header of the HTTP request	
Proxy Auto Detect	Indicates whether to use the system proxy settings or not. Set ProxyAutoDetect to FALSE to use custom proxy settings. This takes precedence over other proxy settings.	true
Proxy User	A user name to be used to authenticate to the ProxyServer proxy. The ProxyUser and ProxyPassword options are used to connect and authenticate against the HTTP proxy specified in ProxyServer. You can select one of the available	john_doe@example.com

Setting	Description	Example
	authentication types in ProxyAuthScheme. If you are using HTTP authentication, set this to the username of a user recognized by the HTTP proxy. If you are using Windows or Kerberos authentication, set this property to a username in one of the following formats: ◦ user@domain ◦ domain\user	
Proxy Password	A password to be used to authenticate to the ProxyServer proxy. This property is used to authenticate to an HTTP proxy server that supports NTLM (Windows), Kerberos, or HTTP authentication. To specify the HTTP proxy, you can set ProxyServer and ProxyPort. To specify the authentication type, set ProxyAuthScheme. If you are using HTTP authentication, additionally set ProxyUser and ProxyPassword to HTTP proxy. If you are using NTLM authentication, set ProxyUser and ProxyPassword to your Windows password. You may also need these to complete Kerberos authentication	
Proxy Server	The hostname or IP address of a proxy to route HTTP traffic through. The hostname or IP address of a proxy to route HTTP traffic through. The driver can use the HTTP, Windows (NTLM), or Kerberos authentication types to authenticate to an HTTP proxy. By default, the driver uses the system proxy. If you need to use another proxy, set ProxyAutoDetect to false.	206.174.193.115
Proxy Port	The TCP port the ProxyServer proxy is running on.	80
Proxy SSL Type	The SSL type to use when	AUTO

Setting	Description	Example
	connecting to the ProxyServer proxy: ◦ **AUTO** - If the URL is an HTTPS URL, the connector will use the TUNNEL option. If the URL is an HTTP URL, the connector will use the NEVER option (default) ◦ **ALWAYS** - the connection is always SSL enabled ◦ **NEVER** - the connection is not SSL enabled ◦ **TUNNEL** - the connection is through a tunneling proxy: The proxy server opens a connection to the remote host and traffic flows back and forth through the proxy	

Retrieving message bodies and attachments

By default, the message body and any attachments are only returned when you select one record from a table. If more than one record is returned, these fields are left blank.

If you want to retrieve the message body and attachments for more than one record, you must set the **Include Message** option in the **Advanced Settings**. Returning these fields is resource-intensive and doing so for a number of records affects performance. If you need to examine the message body or attachments, try using other fields to identify the messages you want to analyze in detail. Then query this subset of messages individually to examine the message body and attachments.

Data connector updates

When you upgrade Analytics, the Robots Agent, or AX Server, you should test any of your scripts that import data using one of the Analytics data connectors (ACCESSDATA command).

The possibility exists that changes made by third-party data sources or ODBC driver vendors required updates to one or more of the data connectors. Scripted data connections may need to be updated in order to continue working correctly.

- **Re-run the import** – The easiest way to update a connection is to manually perform an import using the Data Access window in the upgraded version of Analytics. Copy the ACCESSDATA command from the log and use it to update your script.

> **Note**
> Before connecting to a data source and re-running the import, clear the connector cache to flush the existing set of table names.
> In the **Existing Connections** tab in the Data Access window, beside the connector name, select ··· **> Clear cache**.

- **Update field specifications** – You may also need to update field specifications in the script body to align with table schema changes in the data source or ODBC driver. Possible changes include field names, field data types, and field and record lengths.
- **Check the results of any filtering** – You should also check the results of any filtering that you apply as part of the data import. Confirm that the import filtering is including and excluding records correctly.

Email data connector changes

Specific changes made to the Email data connector are listed below.

Analytics version	Change
14.2	Existing fields `To` and `From` now contain email address only. New fields `FullTo` and `FullFrom` contain email address and email alias.
15.0	Default value for the **Max Items** field in the connector is **-1**. When this value is specified, the connector returns all items during import. After upgrading to **15.0**, if value for **Max Items** is specified as **100** or any other value, the connector returns items only from the specified number of records. If you were using the ACCESSDATA command in a previous version of Analytics, when upgrading to 15.0, to return all items, open the script, update the value for `maxitems` to **-1**, and re-run the script.

Connecting to Epicor ERP

Epicor ERP is an enterprise resource planning (ERP) solution that provides capabilities for job costing, supply chain management, inventory control, and manufacturing management to help organizations make profitable decisions in the manufacturing environment. You can use the Epicor ERP data connector to import your organization's Epicor ERP data.

> **Note**
> The Epicor ERP data connector is provided by our data partner, CData. For information on any of the connection fields, refer the documentation available at the CData website.

Before you start

To connect to Epicor ERP, you must gather the following information:

- the name of the ERP application installed
- the URL of the server where ERP instance is hosted
- user credentials to authenticate to the ERP instance
- the service you want to retrieve data from

For help gathering the connection prerequisites, contact the Epicor ERP administrator in your organization. If your administrator cannot help you, you or your administrator should contact Epicor ERP Support.

Create an Epicor ERP connection

1. From the Analytics main menu, select **Import > Database and application**.
2. From the **New Connections** tab, in the **ACL DSN Connectors (Bundled)** section, select **Epicor ERP.**

   > **Tip**
   > You can filter the list of available connectors by entering a search string in the **Filter connections** box. Connectors are listed alphabetically.

   The DSN Configuration dialog box appears.
3. In the **DSN Configuration** dialog box, click **Show Required**.
4. Provide values for the required fields, if any.

5. Click **Test Connection**.

> **Note**
> When you click the **Test Connection** button, it validates whether the connection and authentication details provided are correct. If you click **OK** without clicking Test Connection, the connection details are saved without any validation and may not work later if the values provided are incorrect.

   The log in page for Epicor ERP appears.

6. Provide the connection details and authenticate your login.

   When the connection is successful, a dialog box opens displaying the success message.

7. Click **OK** in the connection successful dialog box that appears.

   The connection to Epicor ERP is established successfully.

8. In the **DSN Configuration** dialog box, click **OK**.

The connection for Epicor ERP is saved to the **Existing Connections** tab. In the future, you can reconnect to Epicor ERP from the saved connection.

Once the connection is established, the Data Access window opens to the **Staging Area** and you can begin importing data. For help importing data from Epicor ERP, see "Working with the Data Access window" on page 357.

Rename the connection

When you create the Epicor ERP data connector, you cannot provide a Connection name of your choice and the connection is saved to the **Existing Connections** tab with the same name as your data source. If you want to rename it, navigate to the **Existing Connections** tab, expand the **ACL Connectors (DSN)** section, click the ellipsis icon corresponding to Epicor ERP and select **Rename connection**.

Connecting to Exact Online

Exact Online is an accounting and CRM solution that provides integrated packages for manufacturing, wholesale distribution, invoicing, logistics, inventory management, and payroll activities. You can use the Exact Online data connector to import your organization's Exact Online data.

> **Note**
> The Exact Online data connector is provided by our data partner, CData. For information on any of the connection fields, refer the documentation available at the CData website.

Before you start

To connect to Exact Online, you must gather the following information:

- the region of the Exact Online server to which you are connecting
- the client ID assigned when you register your application with an OAuth authorization server
- the client secret assigned when you register your application with an OAuth authorization server

For help gathering the connection prerequisites, contact the Exact Online administrator in your organization. If your administrator cannot help you, you or your administrator should contact Exact Online Support.

Create an Exact Online connection

1. From the Analytics main menu, select **Import > Database and application**.
2. From the **New Connections** tab, in the **ACL DSN Connectors (Bundled)** section, select **Exact Online.**

   > **Tip**
   > You can filter the list of available connectors by entering a search string in the **Filter connections** box. Connectors are listed alphabetically.

   The DSN Configuration dialog box appears.
3. In the **DSN Configuration** dialog box, click **Show Required**.
4. Provide values for the required fields, if any.

5. Click **Test Connection**.

> **Note**
> When you click the **Test Connection** button, it validates whether the connection and authentication details provided are correct. If you click **OK** without clicking Test Connection, the connection details are saved without any validation and may not work later if the values provided are incorrect.

   The log in page for Exact Online appears.

6. Provide the connection details and authenticate your login.

   When the connection is successful, a dialog box opens displaying the success message.

7. Click **OK** in the connection successful dialog box that appears.

   The connection to Exact Online is established successfully.

8. In the **DSN Configuration** dialog box, click **OK**.

The connection for Exact Online is saved to the **Existing Connections** tab. In the future, you can reconnect to Exact Online from the saved connection.

Once the connection is established, the Data Access window opens to the **Staging Area** and you can begin importing data. For help importing data from Exact Online, see "Working with the Data Access window" on page 357.

Rename the connection

When you create the Exact Online data connector, you cannot provide a Connection name of your choice and the connection is saved to the **Existing Connections** tab with the same name as your data source. If you want to rename it, navigate to the **Existing Connections** tab, expand the **ACL Connectors (DSN)** section, click the ellipsis icon corresponding to Exact Online and select **Rename connection**.

Connecting to Exchange

Use the Exchange data connector to import data from Microsoft's Exchange email and calendar server. You can import data from a single Exchange account.

> **Note**
> You cannot use this connector independently of Analytics. You can configure a DSN for the connector driver using the Windows **ODBC Data Source Administrator**, however you must test the DSN connection from within Analytics and not from the connector's Windows DSN configuration dialog.

Before you start

To connect to Exchange, you must gather the following:

- the domain name or IP address of the Exchange server (Exchange Web Services URL)
- the version of the Exchange platform you are connecting to
- the connecting user account, including the user name and password

For help gathering the connection prerequisites, contact the Exchange administrator in your organization. If your administrator cannot help you, you or your administrator should contact Exchange Support.

Create an Exchange connection

1. From the Analytics main menu, select **Import > Database and application**.
2. From the **New Connections** tab, in the **ACL Connectors** section, select **Exchange.**

   > **Tip**
   > You can filter the list of available connectors by entering a search string in the **Filter connections** box. Connectors are listed alphabetically.

3. In the **Data Connection Settings** panel, enter the connection settings and at the bottom of the panel, click **Save and Connect**.

   You can accept the default **Connection Name**, or enter a new one.

The connection for Exchange is saved to the **Existing Connections** tab. In the future, you can reconnect to Exchange from the saved connection.

Once the connection is established, the Data Access window opens to the **Staging Area** and you can begin importing data. For help importing data from Exchange, see "Working with the Data Access window" on page 357.

Connection settings

Basic settings

Setting	Description	Example
User	The user of the Exchange account used to authenticate. Together with Password, this field is used to authenticate to the server.	recipient@example.com
Password	The password of the exchange account used to authenticate. Together with User, this field is used to authenticate to the server.	
Server	The address of the Exchange server to which you are connecting. This should be set to the Exchange Web Services URL. For Exchange Online, you should set it to https://outlook.office365.com/EWS/Exchange.asmx.	https://outlook.office365.com/EWS/Exchange.asmx
Platform	The Platform associated with the Exchange server.	Exchange_Online

Advanced settings

Setting	Description	Example
Convert Datetime to GMT	Converts datetime fields to GMT time zone during import. If false, the datetime value is converted to the operating system time zone of the machine running Analytics.	true
AuthScheme	The scheme used for authentication. Together with Password and User, this field is used to authenticate against the server. BASIC is the default option. Use the following options to select your authentication scheme:	BASIC

Setting	Description	Example
	◦ **NTLM** - uses your Windows credentials for authentication. ◦ **BASIC** - uses HTTP Basic authentication. **Note** Microsoft has announced that is deprecating Basic authentication for Exchange Web Services in October 2020. Consider using an alternative authentication scheme. ◦ **DIGEST** - uses HTTP Digest authentication. ◦ **NEGOTIATE** - negotiates an authentication mechanism with the server. Set AuthScheme to NEGOTIATE to use Kerberos authentication. ◦ **KERBEROSDELEGATION** - uses delegation through the Kerberos protocol. Set the User and Password of the account you want to impersonate.	
Impersonation Type	The type of identifier to use with impersonation while sending requests to the Exchange site: ◦ **PrincipalName** - represents the user principal name (UPN) of the account to use for impersonation. This should be the UPN for the domain where the user account exists ◦ **SID** - represents the security descriptor definition language (SDDL) form of the security identifier (SID) for the account to use for impersonation ◦ **PrimarySmtpAddress** - represents the primary Simple Mail Transfer Protocol (SMTP) address of the account to use for Exchange impersonation. If the primary SMTP address is supplied, it will cost an extra Active Directory directory	PrincipalName

Setting	Description	Example
	service lookup in order to obtain the SID of the user. It is recommended that you use the SID or UPN if they are available ◦ **SmtpAddress** – represents the Simple Mail Transfer Protocol (SMTP) address of the account to use for Exchange Impersonation. If the SMTP address is supplied, it will cost an extra Active Directory lookup in order to obtain the SID of the user. It is recommended that you use the SID or UPN if they are availables	
Impersonation User	The user to impersonate while sending requests to the Exchange site.	
Include Content	Whether to include message body content for all records returned. **Caution** This setting affects performance and may cause your query to timeout if you are working with many records.	false
Limit Key Size	The maximum length of a primary key column. Setting the size to 0 will make the key length revert to the original length. This property makes the connector override the reported length of all the primary key columns.	255
Map to Long Varchar	Controls whether or not a column is returned as SQL_LONGVARCHAR. Use this setting to map any column larger than the specified size so they are reported as SQL_LONGVARCHAR instead of SQL_VARCHAR.	-1
Map To WVarchar	Controls whether or not string types map to SQL_WVARCHAR instead of SQL_VARCHAR. It is set by default.	true

Setting	Description	Example
	String columns must be mapped to SQL_WVARCHAR to accommodate various international character sets, so MapToWVarchar is set to true by default. You may set it to false to use SQL_ VARCHAR instead.	
Pseudo Columns	Indicates whether or not to include pseudo columns as columns to the table. This setting is particularly helpful in Entity Framework, which does not allow you to set a value for a pseudo column unless it is a table column. The value of this connection setting is of the format "Table1=Column1, Table1=Column2, Table2=Column3". You can use the "*" character to include all tables and all columns.	MyTable=*
SSL Server Cert	The certificate to be accepted from the server when connecting using TLS/SSL. You can specify any of the following: ◦ full PEM certificate ◦ path to a local file containing the certificate ◦ the public key ◦ the MD5 Thumbprint (hex values can also be either space or colon separated) ◦ the SHA1 Thumbprint (hex values can also be either space or colon separated) If using a TLS/SSL connection, this property can be used to specify the TLS/SSL certificate to be accepted from the server. Any other certificate that is not trusted by the machine will be rejected.	C:\cert.cer
Support Enhanced SQL	Enhances SQL functionality beyond what can be supported through the API directly, by enabling in-memory client-side processing: ◦ **true** - the connector offloads as much of the SELECT statement	

Setting	Description	Example
	processing as possible to IMAP and then processes the rest of the query in memory. In this way the driver can execute unsupported predicates, joins, and aggregation ◦ **false** - the connector limits SQL execution to what is supported by the IMAP API ## Execution of predicates The connector determines which of the clauses are supported by the data source and then pushes them to the source to get the smallest superset of rows that would satisfy the query. It then filters the rest of the rows locally. The filter operation is streamed, which enables the driver to filter effectively for even very large datasets. ## Execution of Joins The connector uses various techniques to join in memory. The driver trades off memory utilization against the requirement of reading the same table more than once. ## Execution of Aggregates The connector retrieves all rows necessary to process the aggregation in memory.	
Upper Case Identifiers	Report all identifiers in uppercase, including table and column names.	false
Proxy Authentication Scheme	The authentication type to use to authenticate to the ProxyServer proxy. This value specifies the authentication type to use to authenticate to the HTTP proxy specified by ProxyServer and ProxyPort.	BASIC

Setting	Description	Example
	Note The connector will use the system proxy settings by default, without further configuration needed; if you want to connect to another proxy, you will need to set ProxyAutoDetect to false, in addition to ProxyServer and ProxyPort. To authenticate, set ProxyAuthScheme and set ProxyUser and ProxyPassword, if needed. The authentication type can be one of the following: ◦ **BASIC** - The driver performs HTTP BASIC authentication ◦ **DIGEST** - The driver performs HTTP DIGEST authentication. ◦ **NEGOTIATE** - The driver retrieves an NTLM or Kerberos token based on the applicable protocol for authentication ◦ **PROPRIETARY** - The driver does not generate an NTLM or Kerberos token. You must supply this token in the Authorization header of the HTTP request	
Proxy Auto Detect	Indicates whether to use the system proxy settings or not. Set ProxyAutoDetect to FALSE to use custom proxy settings. This takes precedence over other proxy settings.	true
Proxy User	A user name to be used to authenticate to the ProxyServer proxy. The ProxyUser and ProxyPassword options are used to connect and authenticate against the HTTP proxy specified in ProxyServer. You can select one of the available	john_doe@example.com

Setting	Description	Example
	authentication types in ProxyAuthScheme. If you are using HTTP authentication, set this to the username of a user recognized by the HTTP proxy. If you are using Windows or Kerberos authentication, set this property to a username in one of the following formats: ◦ user@domain ◦ domain\user	
Proxy Password	A password to be used to authenticate to the ProxyServer proxy. This property is used to authenticate to an HTTP proxy server that supports NTLM (Windows), Kerberos, or HTTP authentication. To specify the HTTP proxy, you can set ProxyServer and ProxyPort. To specify the authentication type, set ProxyAuthScheme. If you are using HTTP authentication, additionally set ProxyUser and ProxyPassword to HTTP proxy. If you are using NTLM authentication, set ProxyUser and ProxyPassword to your Windows password. You may also need these to complete Kerberos authentication	
Proxy Server	The hostname or IP address of a proxy to route HTTP traffic through. The hostname or IP address of a proxy to route HTTP traffic through. The driver can use the HTTP, Windows (NTLM), or Kerberos authentication types to authenticate to an HTTP proxy. By default, the driver uses the system proxy. If you need to use another proxy, set ProxyAutoDetect to false.	206.174.193.115
Proxy Port	The TCP port the ProxyServer proxy is running on.	80
Proxy SSL Type	The SSL type to use when	AUTO

Setting	Description	Example
	connecting to the ProxyServer proxy: ◦ **AUTO** - If the URL is an HTTPS URL, the connector will use the TUNNEL option. If the URL is an HTTP URL, the connector will use the NEVER option (default) ◦ **ALWAYS** - the connection is always SSL enabled ◦ **NEVER** - the connection is not SSL enabled ◦ **TUNNEL** - the connection is through a tunneling proxy: The proxy server opens a connection to the remote host and traffic flows back and forth through the proxy	

Impersonating user accounts

Users with the required permissions and the *ApplicationImpersonation* role in Exchange can read mailbox data for other users two ways:

- using the **Impersonation Type** and **Impersonation User** fields to configure the connection
- using the **ImpersonationUser** filter in a WHERE clause in SQL mode:

```
SELECT * FROM "Exchange"."DRAFTS" "DRAFTS"  WHERE Imperson-
ationUser='user@example.com'
```

Retrieving the message body

By default, the message body is only returned when you select one record from a table. If more than one record is returned, this field is left blank.

If you want to retrieve the message body for more than one record with a single query, you must set the **Include Content** option in the **Advanced Settings**.

> **Note**
> Using this connector, you can list attachment filenames, however you cannot access the contents of attachment files. You can only access the contents of the message body.

Returning the message body is resource-intensive and doing so for a number of records affects performance. If you need to examine the message body, try using other fields to identify the

messages you want to analyze in detail. Then query this subset of messages individually to examine the message body for each one.

Filtering limitations

The following filter condition and field combinations are not supported:

Fields	Unsupported operators
All DateTime fields	◦ On (=)
SenderName	◦ Is (=) ◦ Starts with (LIKE "%*value*") ◦ Contains (LIKE "%*value*%")
SenderEmailAddress	
FromName	

Data connector updates

When you upgrade Analytics, the Robots Agent, or AX Server, you should test any of your scripts that import data using one of the Analytics data connectors (ACCESSDATA command).

The possibility exists that changes made by third-party data sources or ODBC driver vendors required updates to one or more of the data connectors. Scripted data connections may need to be updated in order to continue working correctly.

- **Re-run the import** – The easiest way to update a connection is to manually perform an import using the Data Access window in the upgraded version of Analytics. Copy the ACCESSDATA command from the log and use it to update your script.

  > **Note**
  > Before connecting to a data source and re-running the import, clear the connector cache to flush the existing set of table names.
  > In the **Existing Connections** tab in the Data Access window, beside the connector name, select ··· **> Clear cache**.

- **Update field specifications** – You may also need to update field specifications in the script body to align with table schema changes in the data source or ODBC driver. Possible changes include field names, field data types, and field and record lengths.
- **Check the results of any filtering** – You should also check the results of any filtering that you apply as part of the data import. Confirm that the import filtering is including and excluding records correctly.

Exchange data connector changes

Specific changes made to the Exchange data connector are listed below.

Analytics version	Exchange table	Change
14.2	n/a	Exchange schema renamed in connector from `Exchange` to `EWS`. Example of required update in ACCESSDATA command: ◦ **Old** - `SELECT * FROM "Exchange"."Calendar" "Calendar"` ◦ **New** - `SELECT * FROM "EWS"."Calendar" "Calendar"`
	Calendar	**Field renamed:** ◦ Recurrence_DayOfWeek renamed to Recurrence_DaysOfWeek **Field added:** ◦ ModifiedOccurrences_Aggregate ◦ DeletedOccurrences_Aggregate **Data type changed:** ◦ Recurrence_StartDate changed from date to datetime ◦ Recurrence_EndDate changed from date to datetime
	Inbox and SentItems	**Field renamed:** ◦ ToRecipients_Name renamed to ToRecipients_Names ◦ ToRecipients_EmailAddress renamed to ToRecipients_EmailAddresses ◦ ToRecipients_ItemId renamed to ToRecipients_ItemIds ◦ CcRecipients_Name renamed to CcRecipients_Names ◦ CcRecipients_EmailAddress renamed to CcRecipients_EmailAddresses ◦ CcRecipients_ItemId renamed to CcRecipients_ItemIds ◦ BccRecipients_Name renamed to BccRecipients_Names ◦ BccRecipients_EmailAddress renamed to BccRecipients_EmailAddresses ◦ BccRecipients_ItemId renamed to BccRecipients_ItemIds
	Tasks	**Field renamed:** ◦ Recurrence_NumberOfOccurences renamed to Recurrence_NumberOfOccurrences

Connecting to Google BigQuery

Google BigQuery is a cloud data service. You can use the Google BigQuery data connector to import your company's BigQuery data.

> **Note**
> Analytics provides Google BigQuery as an optional connector and if it is not available in your Data Access window, it is likely that the connector was not selected during installation. For more information, see "Install optional Analytics data connectors and Python engine" on page 2567.

Before you start

To connect to Google BigQuery, you must gather the following:

- authentication details
- catalog

For help gathering the connection prerequisites, contact the Google BigQuery administrator in your organization. If your administrator cannot help you, you or your administrator should contact Google BigQuery Support.

Create a BigQuery connection

1. From the Analytics main menu, select **Import > Database and application**.
2. From the **New Connections** tab, in the **ACL Connectors** section, select **Google BigQuery.**

   > **Tip**
   > You can filter the list of available connectors by entering a search string in the **Filter connections** box. Connectors are listed alphabetically.

3. In the **Data Connection Settings** panel, enter the connection settings and at the bottom of the panel, click **Save and Connect**.

   You can accept the default **Connection Name**, or enter a new one.

The connection for Google BigQuery is saved to the **Existing Connections** tab. In the future, you can reconnect to Google BigQuery from the saved connection.

Once the connection is established, the Data Access window opens to the **Staging Area** and you can begin importing data. For help importing data from Google BigQuery, see "Working with the Data Access window" on page 357.

Connection settings

Basic settings

Setting	Description	Example
OAuth Mechanism	Specifies the authentication mechanism to authenticate the driver: ◦ **User Authentication** - The driver authenticates as a user, through a Google user account. ◦ **Service Authentication** - The driver authenticates as a service, through a Google service account. When using **User Authentication**, click **Login** to access the Google Sign in page.	User Authentication
Confirmation Code	The code that you obtain from Google for generating a refresh token.	
Refresh Token	The refresh token that you obtain from Google for authorizing access to BigQuery. The refresh token is generated automatically after you provide the confirmation code.	
Email	When using Service Authentication, provide the service account email ID.	
Key File Path	When configuring Service Authentication, set this option to the full path to the .p12 or .json key file that is used to authenticate the service account email address.	
Catalog (Project)	The name of your Google BigQuery project.	

Advanced settings

Setting	Description	Example
Proxy Host	The hostname or IP address of a proxy to route HTTP traffic through. The driver can use the HTTP, Windows (NTLM), or Kerberos authentication types to authenticate to an HTTP proxy.	
Proxy Port	The TCP port the proxy server is running on. The default value is 80.	80
Proxy Username	A user name to be used to authenticate to the proxy server.	
Proxy Password	The password to authenticate to the proxy server.	
Max Requests Per Second (0=unlimited)	Provide the maximum number of requests that can be made per second. To allow an unlimited number of requests per second with no throttling, type 0.	0
Row Per Block	The maximum number of rows that the driver must fetch for each data request.	16384
Default String Column Length	The maximum number of characters to be contained in STRING columns.	65536
Dataset Name for Large Result Sets	ID of the BigQuery dataset to use to store temporary tables for large result sets. Specify a value for this option only if you want to enable support for large result sets. This field is enabled only if you select the **Allow Large Result Sets** option.	_odbc_temp_tables
Temporary Table Expiration Time (ms)	Time (in seconds) until the temporary table expires. To have the table set to never expire, specify the value 0.	3600000

Setting	Description	Example
Language Dialect	Specifies whether the driver executes queries using standard SQL syntax or the legacy BigQuery SQL syntax. ◦ Enabled - Standard SQL syntax ◦ Disabled - Legacy BigQuery SQL syntax	
Enable SQLPrepare Metadata with Legacy SQL (slower)	Specifies whether to use BigQuery's legacy SQL dialect for this query.	
Allow Large Result Sets	Specifies whether to query results larger than 128MB when using Legacy SQL.	

Data connector updates

When you upgrade Analytics, the Robots Agent, or AX Server, you should test any of your scripts that import data using one of the Analytics data connectors (ACCESSDATA command).

The possibility exists that changes made by third-party data sources or ODBC driver vendors required updates to one or more of the data connectors. Scripted data connections may need to be updated in order to continue working correctly.

- **Re-run the import** – The easiest way to update a connection is to manually perform an import using the Data Access window in the upgraded version of Analytics. Copy the ACCESSDATA command from the log and use it to update your script.

  > **Note**
  > Before connecting to a data source and re-running the import, clear the connector cache to flush the existing set of table names.
  > In the **Existing Connections** tab in the Data Access window, beside the connector name, select ··· **> Clear cache**.

- **Update field specifications** – You may also need to update field specifications in the script body to align with table schema changes in the data source or ODBC driver. Possible changes include field names, field data types, and field and record lengths.
- **Check the results of any filtering** – You should also check the results of any filtering that you apply as part of the data import. Confirm that the import filtering is including and excluding records correctly.

Connecting to Jira

Jira is a cloud or server-based platform for software issue tracking and project management. Use the Jira data connector to access your company's Jira data.

Before you start

Jira credentials

To connect to Jira, you must gather the following:

- Jira username
- Jira API token (cloud account), or Jira password (server instance)
- the host name of your company's Jira platform

For help gathering the connection prerequisites, contact the Jira administrator in your organization. If your administrator cannot help you, you or your administrator should contact Jira Support.

Obtaining an API Token

To connect to a Jira cloud account, an API token is necessary for account authentication. To generate a token, log in to your Atlassian account, navigate to the API tokens page, and click **Create API token**. Copy the generated token and enter it in the **API Token** field in the **Data Connection Settings** panel.

Jira "Issues" table

The Jira "Issues" table can contain many custom fields created by your organization. For this reason, the Jira API treats the Issues table dynamically, assembling the fields contained in the table each time you connect to the table. If your Jira administrator makes a change to the custom fields in the Issues table, Analytics scripts that assume a specific selection of fields may break.

To resolve this problem, manually perform a Jira import using the Data Access window in Analytics. Copy the ACCESSDATA command from the log and use it to update your script. Update field specifications in the script body to align with changes to the Issues table.

Create a Jira connection

1. From the Analytics main menu, select **Import > Database and application**.
2. From the **New Connections** tab, in the **ACL Connectors** section, select **Jira.**

> **Tip**
> You can filter the list of available connectors by entering a search string in the **Filter connections** box. Connectors are listed alphabetically.

3. In the **Data Connection Settings** panel, enter the connection settings and at the bottom of the panel, click **Save and Connect**.

   You can accept the default **Connection Name**, or enter a new one.

The connection for Jira is saved to the **Existing Connections** tab. In the future, you can reconnect to Jira from the saved connection.

Once the connection is established, the Data Access window opens to the **Staging Area** and you can begin importing data. For help importing data from Jira, see "Working with the Data Access window" on page 357.

Connection settings

Basic settings

Setting	Description	Example
Host	The host name of the Jira cloud account, or Jira server instance.	https://your_org.atlassian.net
Instance Type	The type of Jira platform used by your company: cloud-based, or on-premise server	Cloud
User	The user account used to authenticate to Jira.	admin_1@your_org.com
API Token	For a Jira cloud account, the Jira API token for the currently authenticated user	
Password	For a Jira server instance, the Jira password used to authenticate the user	
Include Custom Fields	In addition to the standard fields in	true

Setting	Description	Example
	Jira tables, include any custom fields added by your company.	

Advanced settings

Setting	Description	Example
Limit Key Size	The maximum length of a primary key column. Setting the size to 0 will make the key length revert to the original length. This property makes the connector override the reported length of all the primary key columns.	255
Map to Long Varchar	Controls whether or not a column is returned as SQL_LONGVARCHAR. Use this setting to map any column larger than the specified size so they are reported as SQL_LONGVARCHAR instead of SQL_VARCHAR.	-1
Map to WVarchar	Controls whether or not string types map to SQL_WVARCHAR instead of SQL_VARCHAR. It is set by default. String columns must be mapped to SQL_WVARCHAR to accommodate various international character sets, so MapToWVarchar is set to true by default. You may set it to false to use SQL_VARCHAR instead.	true
Pseudo Columns	Indicates whether or not to include pseudo columns as columns to the table. This setting is particularly helpful in Entity Framework, which does not allow you to set a value for a pseudo column unless it is a table column. The value of this connection setting is of the format "Table1=Column1, Table1=Column2, Table2=Column3". You can use the "*" character to include all tables	MyTable=*

Setting	Description	Example
	and all columns.	
SSL Server Certificate	The certificate to be accepted from the server when connecting using TLS/SSL. You can specify any of the following: ◦ full PEM certificate ◦ path to a local file containing the certificate ◦ the public key ◦ the MD5 Thumbprint (hex values can also be either space or colon separated) ◦ the SHA1 Thumbprint (hex values can also be either space or colon separated) If using a TLS/SSL connection, this property can be used to specify the TLS/SSL certificate to be accepted from the server. Any other certificate that is not trusted by the machine will be rejected.	C:\cert.cer
Convert Datetime to GMT	Converts datetime fields to GMT time zone during import. If false, the datetime value is converted to the operating system time zone of the machine running Analytics.	true
Proxy Authentication Scheme	The authentication type to use to authenticate to the ProxyServer proxy. This value specifies the authentic-ation type to use to authenticate to the HTTP proxy specified by ProxyServer and ProxyPort.	BASIC

Setting	Description	Example
	Note The connector will use the system proxy settings by default, without further configuration needed; if you want to connect to another proxy, you will need to set ProxyAutoDetect to false, in addition to ProxyServer and ProxyPort. To authenticate, set ProxyAuthScheme and set ProxyUser and ProxyPassword, if needed. The authentication type can be one of the following: ◦ **BASIC** - The driver performs HTTP BASIC authentication ◦ **DIGEST** - The driver performs HTTP DIGEST authentication. ◦ **NEGOTIATE** - The driver retrieves an NTLM or Kerberos token based on the applicable protocol for authentication ◦ **PROPRIETARY** - The driver does not generate an NTLM or Kerberos token. You must supply this token in the Authorization header of the HTTP request	
Proxy Auto Detect	Indicates whether to use the system proxy settings or not. Set ProxyAutoDetect to FALSE to use custom proxy settings. This takes precedence over other proxy settings.	true
Proxy User	A user name to be used to authenticate to the ProxyServer proxy. The ProxyUser and ProxyPassword options are used to connect and authenticate against the HTTP proxy specified in ProxyServer. You can select one of the available	john_doe@example.com

Setting	Description	Example
	authentication types in ProxyAuthScheme. If you are using HTTP authentication, set this to the username of a user recognized by the HTTP proxy. If you are using Windows or Kerberos authentication, set this property to a username in one of the following formats: ◦ user@domain ◦ domain\user	
Proxy Password	A password to be used to authenticate to the ProxyServer proxy. This property is used to authenticate to an HTTP proxy server that supports NTLM (Windows), Kerberos, or HTTP authentication. To specify the HTTP proxy, you can set ProxyServer and ProxyPort. To specify the authentication type, set ProxyAuthScheme. If you are using HTTP authentication, additionally set ProxyUser and ProxyPassword to HTTP proxy. If you are using NTLM authentication, set ProxyUser and ProxyPassword to your Windows password. You may also need these to complete Kerberos authentication.	
Proxy Server	The hostname or IP address of a proxy to route HTTP traffic through. The hostname or IP address of a proxy to route HTTP traffic through. The driver can use the HTTP, Windows (NTLM), or Kerberos authentication types to authenticate to an HTTP proxy. By default, the driver uses the system proxy. If you need to use another proxy, set ProxyAutoDetect to false.	206.174.193.115
Proxy Port	The TCP port the ProxyServer proxy is running on.	80
Proxy SSL Type	The SSL type to use when	AUTO

Setting	Description	Example
	connecting to the ProxyServer proxy: ◦ **AUTO** - If the URL is an HTTPS URL, the connector will use the TUNNEL option. If the URL is an HTTP URL, the connector will use the NEVER option (default) ◦ **ALWAYS** - the connection is always SSL enabled ◦ **NEVER** - the connection is not SSL enabled ◦ **TUNNEL** - the connection is through a tunneling proxy: The proxy server opens a connection to the remote host and traffic flows back and forth through the proxy	

Data connector updates

When you upgrade Analytics, the Robots Agent, or AX Server, you should test any of your scripts that import data using one of the Analytics data connectors (ACCESSDATA command).

The possibility exists that changes made by third-party data sources or ODBC driver vendors required updates to one or more of the data connectors. Scripted data connections may need to be updated in order to continue working correctly.

- **Re-run the import** – The easiest way to update a connection is to manually perform an import using the Data Access window in the upgraded version of Analytics. Copy the ACCESSDATA command from the log and use it to update your script.

  > **Note**
  > Before connecting to a data source and re-running the import, clear the connector cache to flush the existing set of table names.
  > In the **Existing Connections** tab in the Data Access window, beside the connector name, select ··· **> Clear cache**.

- **Update field specifications** – You may also need to update field specifications in the script body to align with table schema changes in the data source or ODBC driver. Possible changes include field names, field data types, and field and record lengths.
- **Check the results of any filtering** – You should also check the results of any filtering that you apply as part of the data import. Confirm that the import filtering is including and excluding records correctly.

Jira data connector changes

Specific changes made to the Jira data connector are listed below.

Analytics version	Jira table	Fields added	Fields removed
14.2	Attachments	AuthorAccountId	AuthorKey ContentEncoded FilePath Name
	Boards		FilterId (Existing scripted Jira imports that reference this field continue to work, but the field no longer exists.)
	Comments	AuthorAccountId UpdateAuthorAccountId	AuthorKey AuthorName UpdateAuthorKey UpdateAuthorName
	IssueChangeLogs	AuthorAccountId AuthorDisplayName	AuthorKey
	Projects	LeadAccountId	LeadKey LeadName Recent
	SecurityLevels	SecuritySchemeId	Link
	Users	AccountId	Key Name IncludeInactive
	Watchers	AccountId	Key Name
	Worklogs	AuthorAccountId UpdateAuthorAccountId	AuthorName UpdateAuthorName

Connecting to JSON Services

JavaScript Object Notation (JSON) is a standard file format used to transmit data objects consisting of attribute-value pairs and array data types. JSON is a common format for communication with Web services.

> **Note**
> You cannot use this connector independently of Analytics. You can configure a DSN for the connector driver using the Windows **ODBC Data Source Administrator**, however you must test the DSN connection from within Analytics and not from the connector's Windows DSN configuration dialog.

> **Note**
> You cannot use the JSON connector with the Professional Edition of ACL Robotics if the connection requires authentication.

Before you start

To connect to a JSON service, you must determine the structure of the file you intend to query and develop the JSONPath or XPath syntax required to split your file into a tabular structure containing rows. You can import one table per connection.

For help gathering the connection prerequisites, contact the administrator of your JSON service in your company. If your administrator cannot help you, you or your administrator should contact the Support services of the system you are trying to connect to.

JSON connections that require authentication

If the JSON service that you want to connect to requires authenticating with a token or user credentials, you must create a data connection (DSN) in the Windows **ODBC Data Source Administrator**. Once you have created the DSN, including the authentication information, you can select it in the Data Access window.

Create a JSON connection

1. From the Analytics main menu, select **Import > Database and application**.
2. From the **New Connections** tab, in the **ACL Connectors** section, select **JSON.**

> **Tip**
> You can filter the list of available connectors by entering a search string in the **Filter connections** box. Connectors are listed alphabetically.

3. In the **Data Connection Settings** panel, enter the connection settings and at the bottom of the panel, click **Save and Connect**.

   You can accept the default **Connection Name**, or enter a new one.

The connection for JSON is saved to the **Existing Connections** tab. In the future, you can reconnect to JSON from the saved connection.

Once the connection is established, the Data Access window opens to the **Staging Area** and you can begin importing data. For help importing data from JSON, see "Working with the Data Access window" on page 357.

Connection settings

Basic settings

Setting	Description	Example
Table	The JSONPath of an array element within the JSON document (used to split the document into multiple rows). This specifies the JSONPath (or XPath syntax) of an array element and is used to split the document into multiple rows. This property will be used to generate the schema definition when a schema file (RSD) file is not present.	$.value

Setting	Description	Example
	Note JSONPath and XPath are standardized query formats. For more information about the syntax and keywords, consult an online resource.	
JSON Location	The Uniform Resource Identifier (URI) or absolute file path of the JSON resource.	http://sample.example.net/examples.json

Advanced settings

Setting	Description	Example
Limit Key Size	The maximum length of a primary key column. Setting the size to 0 will make the key length revert to the original length. This property makes the connector override the reported length of all the primary key columns.	255
Map to Long Varchar	Controls whether or not a column is returned as SQL_LONGVARCHAR. Use this setting to map any column larger than the specified size so they are reported as SQL_LONGVARCHAR instead of SQL_VARCHAR.	-1
Map To WVarchar	Controls whether or not string types map to SQL_WVARCHAR instead of SQL_VARCHAR. It is set by default. String columns must be mapped to SQL_WVARCHAR to accommodate various international character sets, so MapToWVarchar is set to true by default. You may set it to false to use SQL_VARCHAR instead.	true

Setting	Description	Example
Pseudo Columns	Indicates whether or not to include pseudo columns as columns to the table. This setting is particularly helpful in Entity Framework, which does not allow you to set a value for a pseudo column unless it is a table column. The value of this connection setting is of the format "Table1=Column1, Table1=Column2, Table2=Column3". You can use the "*" character to include all tables and all columns.	MyTable=*
SSL Server Cert	The certificate to be accepted from the server when connecting using TLS/SSL. You can specify any of the following: ◦ full PEM certificate ◦ path to a local file containing the certificate ◦ the public key ◦ the MD5 Thumbprint (hex values can also be either space or colon separated) ◦ the SHA1 Thumbprint (hex values can also be either space or colon separated) If using a TLS/SSL connection, this property can be used to specify the TLS/SSL certificate to be accepted from the server. Any other certificate that is not trusted by the machine will be rejected.	C:\cert.cer
Convert Datetime to GMT	Converts datetime fields to GMT time zone during import. If false, the datetime value is converted to the operating system time zone of the machine running Analytics.	true
Proxy Authentication Scheme	The authentication type to use to authenticate to the ProxyServer proxy. This value specifies the authentication type to use to authenticate to the HTTP proxy specified by ProxyServer and ProxyPort.	BASIC

Setting	Description	Example
	Note The connector will use the system proxy settings by default, without further configuration needed; if you want to connect to another proxy, you will need to set ProxyAutoDetect to false, in addition to ProxyServer and ProxyPort. To authenticate, set ProxyAuthScheme and set ProxyUser and ProxyPassword, if needed. The authentication type can be one of the following: ◦ **BASIC** - The driver performs HTTP BASIC authentication ◦ **DIGEST** - The driver performs HTTP DIGEST authentication. ◦ **NEGOTIATE** - The driver retrieves an NTLM or Kerberos token based on the applicable protocol for authentication ◦ **PROPRIETARY** - The driver does not generate an NTLM or Kerberos token. You must supply this token in the Authorization header of the HTTP request	
Proxy Auto Detect	Indicates whether to use the system proxy settings or not. Set ProxyAutoDetect to FALSE to use custom proxy settings. This takes precedence over other proxy settings.	true
Proxy User	A user name to be used to authenticate to the ProxyServer proxy. The ProxyUser and ProxyPassword options are used to connect and authenticate against the HTTP proxy specified in ProxyServer. You can select one of the available	john_doe@example.com

Setting	Description	Example
	authentication types in ProxyAuthScheme. If you are using HTTP authentication, set this to the username of a user recognized by the HTTP proxy. If you are using Windows or Kerberos authentication, set this property to a username in one of the following formats: ◦ user@domain ◦ domain\user	
Proxy Password	A password to be used to authenticate to the ProxyServer proxy. This property is used to authenticate to an HTTP proxy server that supports NTLM (Windows), Kerberos, or HTTP authentication. To specify the HTTP proxy, you can set ProxyServer and ProxyPort. To specify the authentication type, set ProxyAuthScheme. If you are using HTTP authentication, additionally set ProxyUser and ProxyPassword to HTTP proxy. If you are using NTLM authentication, set ProxyUser and ProxyPassword to your Windows password. You may also need these to complete Kerberos authentication	
Proxy Server	The hostname or IP address of a proxy to route HTTP traffic through. The hostname or IP address of a proxy to route HTTP traffic through. The driver can use the HTTP, Windows (NTLM), or Kerberos authentication types to authenticate to an HTTP proxy. By default, the driver uses the system proxy. If you need to use another proxy, set ProxyAutoDetect to false.	206.174.193.115
Proxy Port	The TCP port the ProxyServer proxy is running on.	80
Proxy SSL Type	The SSL type to use when	AUTO

Setting	Description	Example
	connecting to the ProxyServer proxy: ◦ **AUTO** - If the URL is an HTTPS URL, the connector will use the TUNNEL option. If the URL is an HTTP URL, the connector will use the NEVER option (default) ◦ **ALWAYS** - the connection is always SSL enabled ◦ **NEVER** - the connection is not SSL enabled ◦ **TUNNEL** - the connection is through a tunneling proxy: The proxy server opens a connection to the remote host and traffic flows back and forth through the proxy	

Connecting to LDAP

The *Lightweight Directory Access Protocol* (LDAP) is an industry-standard application protocol for accessing and maintaining distributed directory information services over an Internet Protocol (IP) network. Use the LDAP connector for real-time access to LDAP data, directly from any applications that support ODBC connectivity.

> **Note**
> You cannot use this connector independently of Analytics. You can configure a DSN for the connector driver using the Windows **ODBC Data Source Administrator**, however you must test the DSN connection from within Analytics and not from the connector's Windows DSN configuration dialog.

Before you start

To connect to LDAP data, you must gather the following:

- the domain name or IP address of the LDAP server
- the correct connection port
- the connecting user account, including the distinguished name of the user and the password

For help gathering the connection prerequisites, contact the LDAP administrator in your company. If your administrator cannot help you, you or your administrator should contact the application Support for your LDAP system.

Create an LDAP connection

1. From the Analytics main menu, select **Import > Database and application**.
2. From the **New Connections** tab, in the **ACL Connectors** section, select **LDAP.**

   > **Tip**
   > You can filter the list of available connectors by entering a search string in the **Filter connections** box. Connectors are listed alphabetically.

3. In the **Data Connection Settings** panel, enter the connection settings and at the bottom of the panel, click **Save and Connect**.

   You can accept the default **Connection Name**, or enter a new one.

The connection for LDAP is saved to the **Existing Connections** tab. In the future, you can reconnect to LDAP from the saved connection.

Once the connection is established, the Data Access window opens to the **Staging Area** and you can begin importing data. For help importing data from LDAP, see "Working with the Data Access window" on page 357.

Connection settings

Basic settings

Setting	Description	Example
User	The distinguished name of a user. Together with Password, this field is used to authenticate against the LDAP server.	MYDOMAIN\test
Password	The password for the distinguished name of the specified user. Together with User, this field is used to authenticate against the LDAP server. **Note** You may connect without providing a password if your LDAP server permits anonymous connections. Based on your server's security configuration, anonymous connections may be able to list available tables. However, such connections may not be able to select data from some or all of the tables listed. For more information about your LDAP security configuration, contact your company's administrator.	
Server	The domain name or IP of the LDAP server. This does not need to include the LDAP:\\ portion, only the server domain name or IP.	10.120.1.110

Setting	Description	Example
Port	The port the LDAP server is running on. The default value is 389. Together with Server, this property is used to specify the LDAP server.	389
Base DN	The base portion of the distinguished name, used for limiting results to specific subtrees. Specifying a base DN may greatly improve performance when returning entries for large servers by limiting the number of entries that need to be examined.	DC=myConnection,DC=com
LDAP Version	The LDAP version used to connect to and communicate with the server. Valid options are 2 and 3 for LDAP versions 2 and 3. The connector is a standard LDAP client as specified in RFC 1777, 2251, and other LDAP RFCs.	2
Authentication Mechanism	The authentication mechanism to be used when connecting to the LDAP server: ◦ **SIMPLE (default)** - default plaintext authentication is used to log in to the server ◦ **DIGESTMD5** - the more secure DIGEST-MD5 authentication is used ◦ **NEGOTIATE** - NTLM/Negotiate authentication will be used	SIMPLE
Scope	Whether to limit the scope of the search to: ◦ **WholeSubtree** - the whole subtree (BaseDN and all of its descendants) ◦ **SingleLevel** - a single level (BaseDN and its direct descendants) ◦ **BaseObject** - the base object (BaseDN only) **Tip** Limiting scope can greatly improve the search performance.	BaseObject

Advanced settings

Setting	Description	Example
Convert Datetime to GMT	Converts datetime fields to GMT time zone during import. If false, the datetime value is converted to the operating system time zone of the machine running Analytics.	true
Limit Key Size	The maximum length of a primary key column. Setting the size to 0 will make the key length revert to the original length. This property makes the connector override the reported length of all the primary key columns.	255
Map to Long Varchar	Controls whether or not a column is returned as SQL_LONGVARCHAR. Use this setting to map any column larger than the specified size so they are reported as SQL_LONGVARCHAR instead of SQL_VARCHAR.	-1
Map to WVarchar	Controls whether or not string types map to SQL_WVARCHAR instead of SQL_VARCHAR. It is set by default. String columns must be mapped to SQL_WVARCHAR to accommodate various international character sets, so MapToWVarchar is set to true by default. You may set it to false to use SQL_VARCHAR instead.	true
Pseudo Columns	Indicates whether or not to include pseudo columns as columns to the table. This setting is particularly helpful in Entity Framework, which does not allow you to set a value for a pseudo column unless it is a table column. The value of this connection setting is of the format "Table1=Column1, Table1=Column2, Table2=Column3". You can use the "*" character to include all tables	MyTable=*

Setting	Description	Example
	and all columns.	
Upper Case Identifiers	Report all identifiers in uppercase, including table and column names.	false
SSL Server Certificate	The certificate to be accepted from the server when connecting using TLS/SSL. You can specify any of the following: ◦ full PEM certificate ◦ path to a local file containing the certificate ◦ the public key ◦ the MD5 Thumbprint (hex values can also be either space or colon separated) ◦ the SHA1 Thumbprint (hex values can also be either space or colon separated) If using a TLS/SSL connection, this property can be used to specify the TLS/SSL certificate to be accepted from the server. Any other certificate that is not trusted by the machine will be rejected.	C:\cert.cer
Support Enhanced SQL	Enhances SQL functionality beyond what can be supported through the API directly, by enabling in-memory client-side processing: ◦ **true** - the connector offloads as much of the SELECT statement processing as possible to the LDAP server and then processes the rest of the query in memory. In this way the driver can execute unsupported predicates, joins, and aggregation ◦ **false** - the connector limits SQL execution to what is supported by the LDAP API **Note** This setting must be false to support filtering using the where clause syntax.	false

Setting	Description	Example
	Execution of predicates The connector determines which of the clauses are supported by the data source and then pushes them to the source to get the smallest superset of rows that would satisfy the query. It then filters the rest of the rows locally. The filter operation is streamed, which enables the driver to filter effectively for even very large datasets. **Execution of Joins** The connector uses various techniques to join in memory. The driver trades off memory utilization against the requirement of reading the same table more than once. **Execution of Aggregates** The connector retrieves all rows necessary to process the aggregation in memory.	

Filtering the rows returned

The connector uses an SQL filtering syntax that aligns closely with the LDAP search syntax. Some fields contain delimited data that represents multiple object attributes. Your WHERE clause must account for each value in these delimited fields as if they are distinct values, rather than a single string.

Filtering User on ObjectCategory and ObjectClass

Scenario

You are working with the **User** table and you want to import records where the **ObjectClass** has the following attributes:

- person
- user

You also want to limit the records to those where the **ObjectCategory** has the Computer attribute, and not Person.

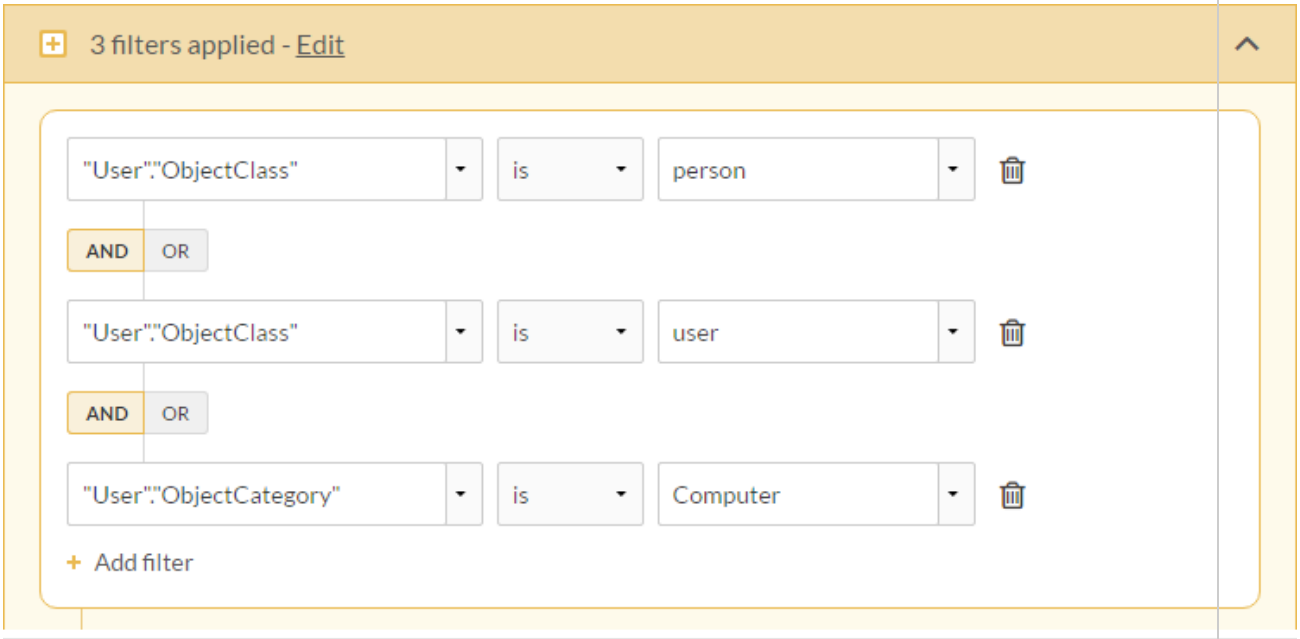
Connecting to the table

First, you connect to the LDAP server, and select the **User** table (subset of fields shown).

ObjectCategory *(ASCII)*	ObjectClass *(ASCII)*	
CN=Person,CN=Schema,CN=Configuration,DC...	top;person;organizationalPerson;user	
CN=Person,CN=Schema,CN=Configuration,DC...	top;person;organizationalPerson;user	
CN=Person,CN=Schema,CN=Configuration,DC...	top;person;organizationalPerson;user	
CN=Computer,CN=Schema,CN=Configuration,...	top;person;organizationalPerson;user;computer	
CN=Person,CN=Schema,CN=Configuration,DC...	top;person;organizationalPerson;user	
CN=Person,CN=Schema,CN=Configuration,DC...	top;person;organizationalPerson;user	
CN=Computer,CN=Schema,CN=Configuration,...	top;person;organizationalPerson;user;computer	
CN=Person,CN=Schema,CN=Configuration,DC...	top;person;organizationalPerson;user	
CN=Computer,CN=Schema,CN=Configuration,...	top;person;organizationalPerson;user;computer	
CN=Person,CN=Schema,CN=Configuration,DC...	top;person;organizationalPerson;user	

Filtering the records

To limit the records to those you want to import, you apply a filter that treats each delimited value as a discrete field.



You then use **SQL Mode** to verify the WHERE clause that the filter constructs:

```
WHERE
  (
    "User"."ObjectClass" = N'person' AND
    "User"."ObjectClass" = N'user' AND
    "User"."ObjectCategory" = N'Computer'
  )
```

Filter results

Once the filter is applied, the table includes records that match the WHERE clause and you import the table.

ObjectCategory *(ASCII)*	ObjectClass *(ASCII)*	
CN=Computer,CN=Schema,CN=Configuration,	top;person;organizationalPerson;user;computer	
CN=Computer,CN=Schema,CN=Configuration,	top;person;organizationalPerson;user;computer	
CN=Computer,CN=Schema,CN=Configuration,	top;person;organizationalPerson;user;computer	
CN=Computer,CN=Schema,CN=Configuration,	top;person;organizationalPerson;user;computer	
CN=Computer,CN=Schema,CN=Configuration,	top;person;organizationalPerson;user;computer	
CN=Computer,CN=Schema,CN=Configuration,	top;person;organizationalPerson;user;computer	
CN=Computer,CN=Schema,CN=Configuration,	top;person;organizationalPerson;user;computer	

Joining LDAP tables

Due to the data model used in LDAP-compliant databases, SQL joins are not recommended. Joins may yield unexpected results.

If you need to join one or more tables from an LDAP data source, you can import multiple tables using the Data Access window and then join them in Analytics. Use filters to limit the number of records and increase efficiency.

Connecting to LinkedIn

LinkedIn is a professional networking website for business professionals that allow its members to create business connections, search jobs, and find potential business and career opportunities. You can use the LinkedIn data connector to import your organization's LinkedIn data.

> **Note**
> The LinkedIn data connector is provided by our data partner, CData. For information on any of the connection fields, refer the documentation available at the CData website.

Before you start

To connect to LinkedIn, you must gather the following information:

- the company ID of the currently logged on user
- specific scope that the user requires for the access token

For help gathering the connection prerequisites, contact the LinkedIn administrator in your organization. If your administrator cannot help you, you or your administrator should contact LinkedIn Support.

Create a LinkedIn connection

1. From the Analytics main menu, select **Import > Database and application**.
2. From the **New Connections** tab, in the **ACL DSN Connectors (Bundled)** section, select **LinkedIn.**

   > **Tip**
   > You can filter the list of available connectors by entering a search string in the **Filter connections** box. Connectors are listed alphabetically.

   The DSN Configuration dialog box appears.
3. In the **DSN Configuration** dialog box, click **Show Required**.
4. Provide values for the required fields, if any.

5. Click **Test Connection**.

> **Note**
> When you click the **Test Connection** button, it validates whether the connection and authentication details provided are correct. If you click **OK** without clicking Test Connection, the connection details are saved without any validation and may not work later if the values provided are incorrect.

   The log in page for LinkedIn appears.

6. Provide the connection details and authenticate your login.

   When the connection is successful, a dialog box opens displaying the success message.

7. Click **OK** in the connection successful dialog box that appears.

   The connection to LinkedIn is established successfully.

8. In the **DSN Configuration** dialog box, click **OK**.

The connection for LinkedIn is saved to the **Existing Connections** tab. In the future, you can reconnect to LinkedIn from the saved connection.

Once the connection is established, the Data Access window opens to the **Staging Area** and you can begin importing data. For help importing data from LinkedIn, see "Working with the Data Access window" on page 357.

Rename the connection

When you create the LinkedIn data connector, you cannot provide a Connection name of your choice and the connection is saved to the **Existing Connections** tab with the same name as your data source. If you want to rename it, navigate to the **Existing Connections** tab, expand the **ACL Connectors (DSN)** section, click the ellipsis icon corresponding to LinkedIn and select **Rename connection**.

Connecting to Marketo

Marketo is a marketing automation platform. Specify a Marketo REST API endpoint to import data from your Marketo system using the connector.

> **Note**
> You cannot use this connector independently of Analytics. You can configure a DSN for the connector driver using the Windows **ODBC Data Source Administrator**, however you must test the DSN connection from within Analytics and not from the connector's Windows DSN configuration dialog.

Before you start

To connect to Marketo, you must configure and obtain OAuth credentials for the endpoint you are trying to access. For information about obtaining these OAuth credentials, see the Marketo Authentication documentation.

> **Note**
> Marketo's API services are subject to limitations for the number of requests per day as well as the number of simultaneous requests. If you are experiencing issues with these limitations, contact your Marketo administrator or Marketo Support.

For help gathering the connection prerequisites, contact the Marketo administrator in your organization. If your administrator cannot help you, you or your administrator should contact Marketo Support.

Create a Marketo connection

1. From the Analytics main menu, select **Import > Database and application**.
2. From the **New Connections** tab, in the **ACL Connectors** section, select **Marketo.**

   > **Tip**
   > You can filter the list of available connectors by entering a search string in the **Filter connections** box. Connectors are listed alphabetically.

3. In the **Data Connection Settings** panel, enter the connection settings and at the bottom of the panel, click **Save and Connect**.

   You can accept the default **Connection Name**, or enter a new one.

The connection for Marketo is saved to the **Existing Connections** tab. In the future, you can reconnect to Marketo from the saved connection.

Once the connection is established, the Data Access window opens to the **Staging Area** and you can begin importing data. For help importing data from Marketo, see "Working with the Data Access window" on page 357.

Connection settings

Basic settings

Setting	Description	Example
OAuth Client Id	The client Id assigned when you register your application with Marketo's OAuth authorization server.	xvz1evFS4wEEPTGEFPHBog
OAuth Client Secret	The client secret assigned when you register your application with Marketo's OAuth authorization server.	L8qq9PZyRg6ieKGEKhZolGC0vJWLw8iEJ88DRdyOg
Rest Endpoint	The URL of the REST Web service endpoint is provided by Marketo on the Admin page of the Marketo website.	https://064-CCJ-768.mktorest.com/rest

Advanced settings

Setting	Description	Example
Limit Key Size	The maximum length of a primary key column. Setting the size to 0 will make the key length revert to the original length. This property makes the connector	255

Setting	Description	Example
	override the reported length of all the primary key columns.	
Map to Long Varchar	Controls whether or not a column is returned as SQL_LONGVARCHAR. Use this setting to map any column larger than the specified size so they are reported as SQL_LONGVARCHAR instead of SQL_VARCHAR.	-1
Map To WVarchar	Controls whether or not string types map to SQL_WVARCHAR instead of SQL_VARCHAR. It is set by default. String columns must be mapped to SQL_WVARCHAR to accommodate various international character sets, so MapToWVarchar is set to true by default. You may set it to false to use SQL_VARCHAR instead.	true
Pseudo Columns	Indicates whether or not to include pseudo columns as columns to the table. This setting is particularly helpful in Entity Framework, which does not allow you to set a value for a pseudo column unless it is a table column. The value of this connection setting is of the format "Table1=Column1, Table1=Column2, Table2=Column3". You can use the "*" character to include all tables and all columns.	MyTable=*
SSL Server Certificate	The certificate to be accepted from the server when connecting using TLS/SSL. You can specify any of the following: ◦ full PEM certificate ◦ path to a local file containing the certificate ◦ the public key ◦ the MD5 Thumbprint (hex values can also be either space or colon separated) ◦ the SHA1 Thumbprint (hex values can also be either space	C:\cert.cer

Setting	Description	Example
	or colon separated) If using a TLS/SSL connection, this property can be used to specify the TLS/SSL certificate to be accepted from the server. Any other certificate that is not trusted by the machine will be rejected.	
Proxy Authentication Scheme	The authentication type to use to authenticate to the ProxyServer proxy. This value specifies the authentication type to use to authenticate to the HTTP proxy specified by ProxyServer and ProxyPort. **Note** The connector will use the system proxy settings by default, without further configuration needed; if you want to connect to another proxy, you will need to set ProxyAutoDetect to false, in addition to ProxyServer and ProxyPort. To authenticate, set ProxyAuthScheme and set ProxyUser and ProxyPassword, if needed. The authentication type can be one of the following: ◦ **BASIC** - The driver performs HTTP BASIC authentication ◦ **DIGEST** - The driver performs HTTP DIGEST authentication. ◦ **NEGOTIATE** - The driver retrieves an NTLM or Kerberos token based on the applicable protocol for authentication ◦ **PROPRIETARY** - The driver does not generate an NTLM or Kerberos token. You must supply this token in the Authorization header of the HTTP request	BASIC

Setting	Description	Example
Proxy Auto Detect	Indicates whether to use the system proxy settings or not. Set ProxyAutoDetect to FALSE to use custom proxy settings. This takes precedence over other proxy settings.	true
Proxy User	A user name to be used to authenticate to the ProxyServer proxy. The ProxyUser and ProxyPassword options are used to connect and authenticate against the HTTP proxy specified in ProxyServer. You can select one of the available authentication types in ProxyAuthScheme. If you are using HTTP authentication, set this to the username of a user recognized by the HTTP proxy. If you are using Windows or Kerberos authentication, set this property to a username in one of the following formats: ◦ user@domain ◦ domain\user	john_doe@example.com
Proxy Password	A password to be used to authenticate to the ProxyServer proxy. This property is used to authenticate to an HTTP proxy server that supports NTLM (Windows), Kerberos, or HTTP authentication. To specify the HTTP proxy, you can set ProxyServer and ProxyPort. To specify the authentication type, set ProxyAuthScheme. If you are using HTTP authentication, additionally set ProxyUser and ProxyPassword to HTTP proxy. If you are using NTLM authentication, set ProxyUser and ProxyPassword to your Windows password. You may also need these to complete Kerberos authentication	
Proxy Server	The hostname or IP address of a proxy to route HTTP traffic through.	206.174.193.115

Setting	Description	Example
	The hostname or IP address of a proxy to route HTTP traffic through. The driver can use the HTTP, Windows (NTLM), or Kerberos authentication types to authenticate to an HTTP proxy. By default, the driver uses the system proxy. If you need to use another proxy, set ProxyAutoDetect to false.	
Proxy Port	The TCP port the ProxyServer proxy is running on.	80
Proxy SSL Type	The SSL type to use when connecting to the ProxyServer proxy: ◦ **AUTO** - If the URL is an HTTPS URL, the connector will use the TUNNEL option. If the URL is an HTTP URL, the connector will use the NEVER option (default) ◦ **ALWAYS** - the connection is always SSL enabled ◦ **NEVER** - the connection is not SSL enabled ◦ **TUNNEL** - the connection is through a tunneling proxy: The proxy server opens a connection to the remote host and traffic flows back and forth through the proxy	AUTO

Connecting to Microsoft SQL Server

Microsoft SQL Server is a widely-used relational database management system. You can use the SQL Server data connector to import your company's SQL Server data.

> **Note**
> Analytics provides SQL Server as an optional connector and if it is not available in your Data Access window, it is likely that the connector was not selected during installation. For more information, see "Install optional Analytics data connectors and Python engine" on page 2567.

Before you start

To connect to SQL Server, you must gather the following:

- the database server's host name
- the correct connection port
- the *Service Principal Name* (SPN) if using Integrated Windows authentication
- your username and password if using standard authentication
- read access for the schema and tables you want to connect to

For help gathering the connection prerequisites, contact the SQL Server administrator in your organization. If your administrator cannot help you, you or your administrator should contact SQL Server Support.

Create a SQL Server connection

1. From the Analytics main menu, select **Import > Database and application**.
2. From the **New Connections** tab, in the **ACL Connectors** section, select **SQL Server.**

   > **Tip**
   > You can filter the list of available connectors by entering a search string in the **Filter connections** box. Connectors are listed alphabetically.

3. In the **Data Connection Settings** panel, enter the connection settings and at the bottom of the panel, click **Save and Connect**.

   You can accept the default **Connection Name**, or enter a new one.

The connection for SQL Server is saved to the **Existing Connections** tab. In the future, you can reconnect to SQL Server from the saved connection.

Once the connection is established, the Data Access window opens to the **Staging Area** and you can begin importing data. For help importing data from SQL Server, see "Working with the Data Access window" on page 357.

Connection settings

Setting	Description	Example
Server	The name or network address of the SQL server instance.	
Port	The port of the server hosting the SQL server database.	1433
Database	The name of the SQL server database running on the specified server.	
Encrypt	To set whether the driver will attempt to negotiate TLS/SSL connections to the server. By default, the driver checks the server certificate against the trusted certificate store of the system. To specify another certificate, set SSLServerCert.	false
CA Certificate	The certificate to be accepted from the server when connecting using TLS/SSL. You can provide any of the following: ◦ a full PEM certificate ◦ path to a local file containing the certificate ◦ public key ◦ MD5 or SHA1 thumbprint (hex values can also be either space or colon separated) Any other certificate that is not trusted by the machine is rejected.	cacerts.pem
Trust Server Certificate	Specifies whether to authenticate to SQL server with Windows Integrated Security. When this is set to true, a Windows identity will be used to perform Windows authentication.	false

Setting	Description	Example
	If this value is false, SQL Server authentication will be used.	
Use Trusted Connection	Specifies whether the driver uses Kerberos protocol to authenticate connections to SQL Server.	true
Server SPN	The service principal name of the SQL Server instance.	
User	The username provided for authentication with SQL Server.	
Password	The password to authenticate the specified user with SQL Server.	

Connecting to MongoDB

MongoDB is a cloud data service. You can use the MongoDB data connector to import your company's MongoDB data.

> **Note**
> Analytics provides MongoDB as an optional connector and if it is not available in your Data Access window, it is likely that the connector was not selected during installation. For more information, see "Install optional Analytics data connectors and Python engine" on page 2567.

Before you start

To connect to MongoDB, you must gather the following:

- server name or IP address of the server hosting the MongoDB database
- correct connection port
- name of MongoDB database

For help gathering the connection prerequisites, contact the MongoDB administrator in your organization. If your administrator cannot help you, you or your administrator should contact MongoDB Support.

Create a MongoDB connection

1. From the Analytics main menu, select **Import > Database and application**.
2. From the **New Connections** tab, in the **ACL Connectors** section, select **MongoDB.**

   > **Tip**
   > You can filter the list of available connectors by entering a search string in the **Filter connections** box. Connectors are listed alphabetically.

3. In the **Data Connection Settings** panel, enter the connection settings and at the bottom of the panel, click **Save and Connect**.

   You can accept the default **Connection Name**, or enter a new one.

The connection for MongoDB is saved to the **Existing Connections** tab. In the future, you can reconnect to MongoDB from the saved connection.

Once the connection is established, the Data Access window opens to the **Staging Area** and you can begin importing data. For help importing data from MongoDB, see "Working with the Data Access window" on page 357.

Connection settings

Basic settings

Setting	Description	Example
Server	The host name or IP address of the server hosting the MongoDB database.	
Port	The number of the TCP port that the MongoDB server uses to listen for client connections. The default value is 27017.	27017
Database	The name of the MongoDB database.	
Connect to Replica Set	Specifies whether the driver can access replica sets in your MongoDB implementation: ◦ Disabled - The driver cannot access replica sets. ◦ Enabled - The driver can access replica sets in your MongoDB implementation.	Disabled
Replica Set Name	The name of the replica set for the driver to access.	
Secondary Servers	A comma-separated list of the servers to use when connecting to a replica set. To indicate the TCP port that the server is using to listen for client connections, append a colon (:) and the port number to the server name or IP address.	
Authentication Mechanism	The authentication mechanism that MongoDB will use to authenticate the connection: ◦ **No Authentication** - The driver	No Authentication

Setting	Description	Example
	does not authenticate the connection. ◦ **MongoDB User Name and Password** - The driver authenticates using the SCRAM-SHA-1 protocol, which is the default authentication protocol used by MongoDB. ◦ **Kerberos** - The driver authenticates using the Kerberos protocol. ◦ **LDAP** - The driver authenticates using the LDAP protocol.	
Service name	The Kerberos service principal name of the MongoDB server.	mongodb
Authentication Source	The name of the MongoDB database for authentication. This value is needed only if the authentication database is different from the database to retrieve data.	admin
Username	The username used to authenticate with MongoDB.	
Password	The password used to authenticate with MongoDB.	

Advanced settings

Setting	Description	Example
Enable SSL	Specifies whether the driver uses SSL to connect to the server.	Disabled
Allow Self-signed Certificate	Specifies whether the driver allows self-signed SSL certificates from the server.	Disabled
PEM Key File	The full path of the .pem file containing the certificate for verifying the client.	
PEM Key Password	The password of the client certificate file that is specified in the PEM Key File field.	

Setting	Description	Example
Certificate Authority File	The full path of the .pem file that you use to verify the server.	
Certificate Authority Directory	The full path of the directory containing the .pem files to verify the server. This setting enables the driver to access multiple .pem files for SSL verification.	
Certificate Revocation List File	The full path of the .pem file containing the list of revoked certificates.	
Enable Double-Buffering	Specifies whether the driver retrieves the data using double-buffering. MongoDB driver is capable of using double-buffering to improve driver performance during SELECT operations.	Enabled
Expose strings as SQL_WVARCHAR	Specifies whether the string data type is mapped to SQL_WVARCHAR or SQL_VARCHAR.	Enabled
Expose binary as SQL_LONGVARBINARY	Specifies whether the driver returns binary columns as data of type SQL_LONGVARBINARY or SQL_VARBINARY.	Enabled
Enable Passdown	Specifies whether the driver optimizes joins between virtual tables, and passes filtering and aggregation optimizations to the MongoDB database for handling.	Enabled
Documents to fetch per block	The maximum number of documents that a query returns at a time. This setting also determines the buffer size used when double-buffering is enabled. The default value is 4096.	4096
String Column Size	The maximum number of characters that can be contained in STRING columns. The default value is 255.	255
Binary Column Size	The maximum data length for binary columns.	32767

Setting	Description	Example
	The default value is 32767.	
Metadata Mechanism	Specifies where the driver looks for the schema definition: ◦ **Database** - The driver loads the schema definition from the MongoDB database. ◦ **Local File** - The driver loads the schema definition from the JSON file specified in the Local File field.	Database
Local file	The full path of a local JSON file containing the schema definition that you want the driver to use when connecting to MongoDB.	
Sampling Method	Specifies how the driver samples data when generating a temporary schema definition: ◦ **Forward** - The driver samples data sequentially, starting from the first record in the database. ◦ **Backwards**: The driver samples data backwards starting from the last record in the database.	Forward
Documents to sample (0 for all documents)	Maximum number of records that the driver can sample to generate a temporary schema definition. When this option is set to 0, the driver samples every document in the database. The default value is 100.	100
Sampling Step Size	The interval at which the driver samples records when scanning through the database to generate a temporary schema definition. For example, if you set this option to 2, then the driver samples every second record in the database. The default value is 1.	1
Writeback Batch Size (1-1000)	The maximum number of documents that the driver can handle at one time during a write operation. The default value is 500.	500

Setting	Description	Example
Write Concern	Total number of primary and secondary servers that must acknowledge a write operation for the driver to report a successful write operation. When this option is set to 0, the driver does not require write operations to be acknowledged. The default value is 1.	1
Write Concern Timeout	Maximum number of seconds that the driver waits for a secondary server to acknowledge a write operation before reporting that the operation has failed. When this option is set to 0, the driver does not time out. Instead, the driver waits until all secondary servers acknowledge the write operation, and then reports that the operation has succeeded. The default value is 0.	0
Write Concern Journaled Writes	Specifies whether the driver requires the data from a write operation to be committed to the journal before the write operation can be acknowledged.	Disabled

Data connector updates

When you upgrade Analytics, the Robots Agent, or AX Server, you should test any of your scripts that import data using one of the Analytics data connectors (ACCESSDATA command).

The possibility exists that changes made by third-party data sources or ODBC driver vendors required updates to one or more of the data connectors. Scripted data connections may need to be updated in order to continue working correctly.

- **Re-run the import** – The easiest way to update a connection is to manually perform an import using the Data Access window in the upgraded version of Analytics. Copy the ACCESSDATA command from the log and use it to update your script.

> **Note**
> Before connecting to a data source and re-running the import, clear the connector cache to flush the existing set of table names.
> In the **Existing Connections** tab in the Data Access window, beside the connector name, select ··· **> Clear cache**.

- **Update field specifications** – You may also need to update field specifications in the script body to align with table schema changes in the data source or ODBC driver. Possible changes include field names, field data types, and field and record lengths.
- **Check the results of any filtering** – You should also check the results of any filtering that you apply as part of the data import. Confirm that the import filtering is including and excluding records correctly.

MongoDB data connector changes

Specific changes made to the MongoDB data connector are listed below.

Analytics version	Change
14.2	The connector no longer supports connecting to MongoDB 3.0 and 3.2. Connections can be made to MongoDB 3.4, 3.6, and 4.0.

Connecting to MySQL

MySQL is a popular open source relational database management system. Use the MySQL data connector to import your company's MySQL data.

> **Note**
>
> You cannot use this connector independently of Analytics. You can configure a DSN for the connector driver using the Windows **ODBC Data Source Administrator**, however you must test the DSN connection from within Analytics and not from the connector's Windows DSN configuration dialog.

Before you start

To connect to MySQL, you must gather the following:

- the database server's host name
- the correct connection port
- your username and password if using standard authentication
- read access for the schema and tables you want to connect to

For help gathering the connection prerequisites, contact the MySQL administrator in your organization. If your administrator cannot help you, you or your administrator should contact MySQL Support.

Create a MySQL connection

1. From the Analytics main menu, select **Import > Database and application**.
2. From the **New Connections** tab, in the **ACL Connectors** section, select **MySQL.**

   > **Tip**
   >
   > You can filter the list of available connectors by entering a search string in the **Filter connections** box. Connectors are listed alphabetically.

3. In the **Data Connection Settings** panel, enter the connection settings and at the bottom of the panel, click **Save and Connect**.

   You can accept the default **Connection Name**, or enter a new one.

The connection for MySQL is saved to the **Existing Connections** tab. In the future, you can reconnect to MySQL from the saved connection.

Once the connection is established, the Data Access window opens to the **Staging Area** and you can begin importing data. For help importing data from MySQL, see "Working with the Data Access window" on page 357.

Connection settings

Basic settings

Setting	Description	Example
Server	The host name or IP of the server hosting the MySQL database.	192.168.0.1
Port	The port of the server hosting the MySQL database.	3306
Database	The name of the default database to connect to when connecting to the MySQL Server.	
User	The user to authenticate when connecting to MySQL.	
Password	The password to authenticate the specified user with the MySQL server.	

Advanced settings

Setting	Description	Example
Integrated Security	Specifies whether to authenticate to MySQL server with Windows Integrated Security.	false
Limit Key Size	The maximum length of a primary key column. Setting the size to 0 will make the key length revert to the original length. This property makes the connector override the reported length of all the primary key columns.	255
Map to Long Varchar	Controls whether or not a column is returned as SQL_LONGVARCHAR.	-1

Setting	Description	Example
	Use this setting to map any column larger than the specified size so they are reported as SQL_LONGVARCHAR instead of SQL_VARCHAR.	
Map to WVarchar	Controls whether or not string types map to SQL_WVARCHAR instead of SQL_VARCHAR. It is set by default. String columns must be mapped to SQL_WVARCHAR to accommodate various international character sets, so MapToWVarchar is set to true by default. You may set it to false to use SQL_VARCHAR instead.	true
Pseudo Columns	Indicates whether or not to include pseudo columns as columns to the table. The value of this connection setting is of the format "Table1=Column1, Table1=Column2, Table2=Column3". You can use the "*" character to include all tables and all columns.	MyTable=*
Query Passthrough	This option passes the query to the MySQL server directly as is.	true
Upper Case Identifiers	This property reports all identifiers, including table and column names, in uppercase.	false

Connecting to NetSuite

NetSuite is a cloud-based enterprise resource planning (ERP) service from Oracle that enables organizations to manage business processes in a single system. You can use the NetSuite data connector to import your organization's NetSuite data.

> **Note**
> The NetSuite data connector is provided by our data partner, CData. For information on any of the connection fields, refer the documentation available at the CData website.

Before you start

To connect to NetSuite, you must gather the following information:

- the company account associated with NetSuite
- user credentials to log in
- OAuth access tokens and credentials
- the type of schema to use

For help gathering the connection prerequisites, contact the NetSuite administrator in your organization. If your administrator cannot help you, you or your administrator should contact NetSuite Support.

Create a NetSuite connection

1. From the Analytics main menu, select **Import > Database and application**.
2. From the **New Connections** tab, in the **ACL DSN Connectors (Bundled)** section, select **NetSuite.**

   > **Tip**
   > You can filter the list of available connectors by entering a search string in the **Filter connections** box. Connectors are listed alphabetically.

   The DSN Configuration dialog box appears.
3. In the **DSN Configuration** dialog box, click **Show Required**.
4. Provide values for the required fields, if any.

5. Click **Test Connection**.

> **Note**
> When you click the **Test Connection** button, it validates whether the connection and authentication details provided are correct. If you click **OK** without clicking Test Connection, the connection details are saved without any validation and may not work later if the values provided are incorrect.

   The log in page for NetSuite appears.

6. Provide the connection details and authenticate your login.

   When the connection is successful, a dialog box opens displaying the success message.

7. Click **OK** in the connection successful dialog box that appears.

   The connection to NetSuite is established successfully.

8. In the **DSN Configuration** dialog box, click **OK**.

The connection for NetSuite is saved to the **Existing Connections** tab. In the future, you can reconnect to NetSuite from the saved connection.

Once the connection is established, the Data Access window opens to the **Staging Area** and you can begin importing data. For help importing data from NetSuite, see "Working with the Data Access window" on page 357.

Rename the connection

When you create the NetSuite data connector, you cannot provide a Connection name of your choice and the connection is saved to the **Existing Connections** tab with the same name as your data source. If you want to rename it, navigate to the **Existing Connections** tab, expand the **ACL Connectors (DSN)** section, click the ellipsis icon  corresponding to NetSuite and select **Rename connection**.

Connecting to OData

OData is an open REST-based protocol querying and updating data. You can use the OData data connector to import data your organization's OData data.

> **Note**
> The OData data connector is provided by our data partner, CData. For information on any of the connection fields, refer the documentation available at the CData website.

Before you start

To connect to OData, you must gather the following information:

- root URL to the organization or the OData services file
- credentials to connect to the OData account
- authentication scheme used
- the Microsoft Online tenant used to access data
- the Azure Active resource to authenticate to

For help gathering the connection prerequisites, contact the OData administrator in your organization. If your administrator cannot help you, you or your administrator should contact OData Support.

Create an OData connection

1. From the Analytics main menu, select **Import > Database and application**.
2. From the **New Connections** tab, in the **ACL DSN Connectors (Bundled)** section, select **OData.**

   > **Tip**
   > You can filter the list of available connectors by entering a search string in the **Filter connections** box. Connectors are listed alphabetically.

   The DSN Configuration dialog box appears.
3. In the **DSN Configuration** dialog box, click **Show Required**.
4. Provide values for the required fields, if any.

5. Click **Test Connection**.

> **Note**
> When you click the **Test Connection** button, it validates whether the connection and authentication details provided are correct. If you click **OK** without clicking Test Connection, the connection details are saved without any validation and may not work later if the values provided are incorrect.

   The log in page for OData appears.

6. Provide the connection details and authenticate your login.

   When the connection is successful, a dialog box opens displaying the success message.

7. Click **OK** in the connection successful dialog box that appears.

   The connection to OData is established successfully.

8. In the **DSN Configuration** dialog box, click **OK**.

The connection for OData is saved to the **Existing Connections** tab. In the future, you can reconnect to OData from the saved connection.

Once the connection is established, the Data Access window opens to the **Staging Area** and you can begin importing data. For help importing data from OData, see "Working with the Data Access window" on page 357.

Rename the connection

When you create the OData data connector, you cannot provide a Connection name of your choice and the connection is saved to the **Existing Connections** tab with the same name as your data source. If you want to rename it, navigate to the **Existing Connections** tab, expand the **ACL Connectors (DSN)** section, click the ellipsis icon  corresponding to OData and select **Rename connection**.

Connecting to Open Exchange Rates

Open Exchange Rates is a live and historical foreign exchange (forex) rate service that provides data for over 200 worldwide and digital currencies. Data is tracked and blended algorithmically from multiple reliable sources, ensuring consistency.

> **Note**
> You cannot use this connector independently of Analytics. You can configure a DSN for the connector driver using the Windows **ODBC Data Source Administrator**, however you must test the DSN connection from within Analytics and not from the connector's Windows DSN configuration dialog.

Before you start

To connect to Open Exchange Rates data, you must register for an Open Exchange Rates App ID. To obtain an App ID, sign up at openexchangerates.org.

For help gathering the connection prerequisites, contact the Open Exchange Rates administrator in your organization. If your administrator cannot help you, you or your administrator should contact Open Exchange Rates Support.

Create an Open Exchange Rates connection

1. From the Analytics main menu, select **Import > Database and application**.
2. From the **New Connections** tab, in the **ACL Connectors** section, select **Open Exchange Rates.**

   > **Tip**
   > You can filter the list of available connectors by entering a search string in the **Filter connections** box. Connectors are listed alphabetically.

3. In the **Data Connection Settings** panel, enter the connection settings and at the bottom of the panel, click **Save and Connect**.

   You can accept the default **Connection Name**, or enter a new one.

The connection for Open Exchange Rates is saved to the **Existing Connections** tab. In the future, you can reconnect to Open Exchange Rates from the saved connection.

Once the connection is established, the Data Access window opens to the **Staging Area** and you can begin importing data. For help importing data from Open Exchange Rates, see "Working with the Data Access window" on page 357.

Connection settings

Basic settings

Setting	Description	Example
Application Id	The App ID you obtain when you register with Open Exchange Rates. The value will be displayed in your Open Exchange Rates dashboard.	881BDCBDB7FAABF047A6178DF4956172

Advanced settings

Setting	Description	Example
Limit Key Size	The maximum length of a primary key column. Setting the size to 0 will make the key length revert to the original length. This property makes the connector override the reported length of all the primary key columns.	255
Map to Long Varchar	Controls whether or not a column is returned as SQL_LONGVARCHAR. Use this setting to map any column larger than the specified size so they are reported as SQL_LONGVARCHAR instead of SQL_VARCHAR.	-1
Map to WVarchar	Controls whether or not string types map to SQL_WVARCHAR instead of SQL_VARCHAR. It is set by default.	true

Setting	Description	Example
	String columns must be mapped to SQL_WVARCHAR to accommodate various international character sets, so MapToWVarchar is set to true by default. You may set it to false to use SQL_ VARCHAR instead.	
Pseudo Columns	Indicates whether or not to include pseudo columns as columns to the table. This setting is particularly helpful in Entity Framework, which does not allow you to set a value for a pseudo column unless it is a table column. The value of this connection setting is of the format "Table1=Column1, Table1=Column2, Table2=Column3". You can use the "*" character to include all tables and all columns.	MyTable=*
SSL Server Cert	The certificate to be accepted from the server when connecting using TLS/SSL. You can specify any of the following: ◦ full PEM certificate ◦ path to a local file containing the certificate ◦ the public key ◦ the MD5 Thumbprint (hex values can also be either space or colon separated) ◦ the SHA1 Thumbprint (hex values can also be either space or colon separated) If using a TLS/SSL connection, this property can be used to specify the TLS/SSL certificate to be accepted from the server. Any other certificate that is not trusted by the machine will be rejected.	C:\cert.cer
Convert Datetime to GMT	Converts datetime fields to GMT time zone during import. If false, the datetime value is converted to the operating system time zone of the machine running Analytics.	true
Proxy Auth Scheme	The authentication type to use to	BASIC

Setting	Description	Example
	authenticate to the ProxyServer proxy. This value specifies the authentication type to use to authenticate to the HTTP proxy specified by ProxyServer and ProxyPort. **Note** The connector will use the system proxy settings by default, without further configuration needed; if you want to connect to another proxy, you will need to set ProxyAutoDetect to false, in addition to ProxyServer and ProxyPort. To authenticate, set ProxyAuthScheme and set ProxyUser and ProxyPassword, if needed. The authentication type can be one of the following: ◦ **BASIC** - The driver performs HTTP BASIC authentication ◦ **DIGEST** - The driver performs HTTP DIGEST authentication. ◦ **NEGOTIATE** - The driver retrieves an NTLM or Kerberos token based on the applicable protocol for authentication ◦ **PROPRIETARY** - The driver does not generate an NTLM or Kerberos token. You must supply this token in the Authorization header of the HTTP request	
Proxy Auto Detect	Indicates whether to use the system proxy settings or not. Set ProxyAutoDetect to FALSE to use custom proxy settings. This takes precedence over other proxy settings.	true
Proxy User	A user name to be used to authenticate to the ProxyServer	john_doe@example.com

Setting	Description	Example
	proxy. The ProxyUser and ProxyPassword options are used to connect and authenticate against the HTTP proxy specified in ProxyServer. You can select one of the available authentication types in ProxyAuthScheme. If you are using HTTP authentication, set this to the username of a user recognized by the HTTP proxy. If you are using Windows or Kerberos authentication, set this property to a username in one of the following formats: ◦ user@domain ◦ domain\user	
Proxy Password	A password to be used to authenticate to the ProxyServer proxy. This property is used to authenticate to an HTTP proxy server that supports NTLM (Windows), Kerberos, or HTTP authentication. To specify the HTTP proxy, you can set ProxyServer and ProxyPort. To specify the authentication type, set ProxyAuthScheme. If you are using HTTP authentication, additionally set ProxyUser and ProxyPassword to HTTP proxy. If you are using NTLM authentication, set ProxyUser and ProxyPassword to your Windows password. You may also need these to complete Kerberos authentication	
Proxy Server	The hostname or IP address of a proxy to route HTTP traffic through. The hostname or IP address of a proxy to route HTTP traffic through. The driver can use the HTTP, Windows (NTLM), or Kerberos authentication types to authenticate to an HTTP proxy. By default, the driver uses the system proxy. If you need to use another proxy, set ProxyAutoDetect	206.174.193.115

Setting	Description	Example
	to false.	
Proxy Port	The TCP port the ProxyServer proxy is running on.	80
Proxy SSL Type	The SSL type to use when connecting to the ProxyServer proxy: ◦ **AUTO** - If the URL is an HTTPS URL, the connector will use the TUNNEL option. If the URL is an HTTP URL, the connector will use the NEVER option (default) ◦ **ALWAYS** - the connection is always SSL enabled ◦ **NEVER** - the connection is not SSL enabled ◦ **TUNNEL** - the connection is through a tunneling proxy: The proxy server opens a connection to the remote host and traffic flows back and forth through the proxy	AUTO

Connecting to Oracle

Oracle is a widely used *relational database management system* (RDBMS). You can use the Oracle data connector to import data from your company's on-premise Oracle database.

> **Note**
>
> The Oracle data connector does not support importing data from Oracle Cloud or Oracle Fusion data sources.
>
> ACL for Windows provides Oracle as an optional connector and if it is not available in your Data Access window, it is likely that the connector was not selected during installation. For more information, see "Install optional Analytics data connectors and Python engine" on page 2567.

Before you start

Oracle credentials

To connect to Oracle, you must gather the following:

- your Oracle username and password
- the host name or IP address and service name of the database
- read access to the tables in the database

For help gathering the connection prerequisites, contact the Oracle administrator in your organization. If your administrator cannot help you, you or your administrator should contact Oracle Support.

Oracle Instant Client

The connector requires that Oracle Instant Client is installed on the same computer as Analytics. The bitness of the Instant Client must match your Operating System's bitness. If a 32-bit Oracle Instant Client is installed on a 64-bit machine, the connection fails.

Create an Oracle connection

1. From the Analytics main menu, select **Import > Database and application**.
2. From the **New Connections** tab, in the **ACL Connectors** section, select **Oracle.**

> **Tip**
> You can filter the list of available connectors by entering a search string in the **Filter connections** box. Connectors are listed alphabetically.

3. In the **Data Connection Settings** panel, enter the connection settings and at the bottom of the panel, click **Save and Connect**.

   You can accept the default **Connection Name**, or enter a new one.

The connection for Oracle is saved to the **Existing Connections** tab. In the future, you can reconnect to Oracle from the saved connection.

Once the connection is established, the Data Access window opens to the **Staging Area** and you can begin importing data. For help importing data from Oracle, see "Working with the Data Access window" on page 357.

Connection settings

Basic settings

Setting	Description	Example
Server	The host name or IP of the server hosting the Oracle database.	localhost
Port	The port to connect to the server hosting the Oracle database.	1521
Service Name	The service name of the Oracle database.	XE
User	The user Id for authentication with the Oracle database.	SYSTEM
Password	The password to authenticate the specified user with the Oracle database.	

Advanced settings

Setting	Description	Example
Browsable Schemas	To provide comma-separated list of schemas to restrict browse-able database object tree.	BrowsableSchemas=SYSTEM,SYS

Connecting to Oracle Eloqua

Oracle Eloqua is a marketing automation software as a service (SaaS) platform for that organizations manage marketing campaigns and generate sales leads. You can use the Oracle Eloqua data connector to import data your organization's Eloqua data.

> **Note**
> The Oracle Eloqua data connector is provided by our data partner, CData. For information on any of the connection fields, refer the documentation available at the CData website.

Before you start

To connect to Eloqua, you must gather the following information:

- the company using the Eloqua account
- credentials to connect to the Eloqua account

For help gathering the connection prerequisites, contact the Oracle Eloqua administrator in your organization. If your administrator cannot help you, you or your administrator should contact Oracle Eloqua Support.

Create an Oracle Eloqua connection

1. From the Analytics main menu, select **Import > Database and application**.
2. From the **New Connections** tab, in the **ACL DSN Connectors (Bundled)** section, select **Oracle Eloqua.**

   > **Tip**
   > You can filter the list of available connectors by entering a search string in the **Filter connections** box. Connectors are listed alphabetically.

   The DSN Configuration dialog box appears.
3. In the **DSN Configuration** dialog box, click **Show Required**.
4. Provide values for the required fields, if any.

5. Click **Test Connection**.

> **Note**
> When you click the **Test Connection** button, it validates whether the connection and authentication details provided are correct. If you click **OK** without clicking Test Connection, the connection details are saved without any validation and may not work later if the values provided are incorrect.

   The log in page for Oracle Eloqua appears.

6. Provide the connection details and authenticate your login.

   When the connection is successful, a dialog box opens displaying the success message.

7. Click **OK** in the connection successful dialog box that appears.

   The connection to Oracle Eloqua is established successfully.

8. In the **DSN Configuration** dialog box, click **OK**.

The connection for Oracle Eloqua is saved to the **Existing Connections** tab. In the future, you can reconnect to Oracle Eloqua from the saved connection.

Once the connection is established, the Data Access window opens to the **Staging Area** and you can begin importing data. For help importing data from Oracle Eloqua, see "Working with the Data Access window" on page 357.

Rename the connection

When you create the Oracle Eloqua data connector, you cannot provide a Connection name of your choice and the connection is saved to the **Existing Connections** tab with the same name as your data source. If you want to rename it, navigate to the **Existing Connections** tab, expand the **ACL Connectors (DSN)** section, click the ellipsis icon corresponding to Oracle Eloqua and select **Rename connection**.

Connecting to Oracle Sales Cloud

Oracle Sales Cloud is a customer relationship management (CRM) solution that enables organizations to accelerate sales and improve customer engagement with analytics and other collaborative features. You can use the Oracle Sales Cloud data connector to import data your organization's Oracle Sales Cloud data.

> **Note**
> The Oracle Sales Cloud data connector is provided by our data partner, CData. For information on any of the connection fields, refer the documentation available at the CData website.

Before you start

To connect to Oracle Sales Cloud, you must gather the following information:

- the URL to the Oracle Sales Cloud server
- credentials to connect to the Oracle Sales Cloud account on the server

For help gathering the connection prerequisites, contact the Oracle Sales Cloud administrator in your organization. If your administrator cannot help you, you or your administrator should contact Oracle Sales Cloud Support.

Create an Oracle Sales Cloud connection

1. From the Analytics main menu, select **Import > Database and application**.
2. From the **New Connections** tab, in the **ACL DSN Connectors (Bundled)** section, select **Oracle Sales Cloud.**

   > **Tip**
   > You can filter the list of available connectors by entering a search string in the **Filter connections** box. Connectors are listed alphabetically.

   The DSN Configuration dialog box appears.
3. In the **DSN Configuration** dialog box, click **Show Required**.
4. Provide values for the required fields, if any.

5. Click **Test Connection**.

> **Note**
> When you click the **Test Connection** button, it validates whether the connection and authentication details provided are correct. If you click **OK** without clicking Test Connection, the connection details are saved without any validation and may not work later if the values provided are incorrect.

   The log in page for Oracle Sales Cloud appears.

6. Provide the connection details and authenticate your login.

   When the connection is successful, a dialog box opens displaying the success message.

7. Click **OK** in the connection successful dialog box that appears.

   The connection to Oracle Sales Cloud is established successfully.

8. In the **DSN Configuration** dialog box, click **OK**.

The connection for Oracle Sales Cloud is saved to the **Existing Connections** tab. In the future, you can reconnect to Oracle Sales Cloud from the saved connection.

Once the connection is established, the Data Access window opens to the **Staging Area** and you can begin importing data. For help importing data from Oracle Sales Cloud, see "Working with the Data Access window" on page 357.

Rename the connection

When you create the Oracle Sales Cloud data connector, you cannot provide a Connection name of your choice and the connection is saved to the **Existing Connections** tab with the same name as your data source. If you want to rename it, navigate to the **Existing Connections** tab, expand the **ACL Connectors (DSN)** section, click the ellipsis icon corresponding to Oracle Sales Cloud and select **Rename connection**.

Connecting to Presto

Presto is an open source SQL query engine to run interactive analytic queries against different data sources. You can use the Presto data connector to import your company's Presto data.

Before you start

To connect to Presto, you must gather the following:

- user name and password
- the server's host name or IP address
- the correct connection port
- the correct authentication scheme

For help gathering the connection prerequisites, contact the Presto administrator in your organization. If your administrator cannot help you, you or your administrator should contact Presto Support.

Create a Presto connection

1. From the Analytics main menu, select **Import > Database and application**.
2. From the **New Connections** tab, in the **ACL Connectors** section, select **Presto.**

> **Tip**
> You can filter the list of available connectors by entering a search string in the **Filter connections** box. Connectors are listed alphabetically.

3. In the **Data Connection Settings** panel, enter the connection settings and at the bottom of the panel, click **Save and Connect**.

   You can accept the default **Connection Name**, or enter a new one.

The connection for Presto is saved to the **Existing Connections** tab. In the future, you can reconnect to Presto from the saved connection.

Once the connection is established, the Data Access window opens to the **Staging Area** and you can begin importing data. For help importing data from Presto, see "Working with the Data Access window" on page 357.

Connection settings

Basic settings

Setting	Description	Example
User	The Presto user account to authenticate against the Presto server.	
Password	The password used to authenticate the user with the server.	
Server	The host name or IP address of the Presto REST server.	
Port	The port for the Presto REST server.	8080
Catalog	The name of the catalog to use for all requests against the server. If this value is not set, the driver will retrieve the available catalogs from the Presto server.	
Auth Scheme	Specifies the authentication scheme that the driver uses. The options available are as follows: ◦ **None -** The driver does not authenticate the connection. ◦ **LDAP -** The driver uses LDAP to authenticate the connection. ◦ **Kerberos -** The driver uses Kerberos to authenticate the connection.	
Kerberos KDC	The Kerberos Key Distribution Center (KDC) service used to authenticate the user. If the Kerberos KDC is not specified, the driver will attempt to detect this property automatically from the KRB5 Config File or the Domain Name and Host.	
Kerberos Realm	The Kerberos Realm used to authenticate the user with the	

Setting	Description	Example
	Kerberos Key Distribution Service (KDC).	
Kerberos SPN	The Service Principal Name (SPN) for the Kerberos Domain Controller. If the SPN on the Kerberos Domain Controller is not same as the URL that you are authenticating to, you can use this property to set the SPN.	
Kerberos Keytab File	The Keytab file that contains your pairs of Kerberos principals and encrypted keys.	
Use SSL	Specifies whether SSL is enabled or not.	

Advanced settings

Setting	Description	Example
Limit Key Size	The maximum length of a primary key column.	255
Map To Long Varchar	This property controls whether a column is returned as SQL_LONGVARCHAR or not.	-1
Map To WVarchar	This property controls whether string types map to SQL_WVARCHAR instead of SQL_VARCHAR.	
Pseudo Columns	This property indicates whether to include pseudo columns as columns to the table. The value of this connection setting is of the format "Table1=Column1, Table1=Column2, Table2=Column3".	
SSL Server Certificate	The certificate that must be accepted from the server when connecting using SSL. You can provide any of the following:	

Setting	Description	Example
	◦ a full PEM certificate ◦ path to a local file containing the certificate ◦ public key ◦ MD5 or SHA1 thumbprint (hex values can also be either space or colon separated) Any other certificate that is not trusted by the machine is rejected.	
Query Passthrough	When you enable this option, all queries are passed directly to Presto.	
Convert Datetime To GMT	Converts datetime fields to GMT time zone during import. If this option is disabled, the datetime value is converted to the operating system time zone of the system running Analytics.	
Proxy Authentication Scheme	Specifies the authentication scheme to authenticate to the Proxy Server. The options available are as follows: ◦ **BASIC -** The driver performs HTTP BASIC authentication. ◦ **DIGEST -** The driver performs HTTP DIGEST authentication. ◦ **NEGOIATE -** The driver retrieves an NTLM or Kerberos token, based on the protocol set for authentication. ◦ **PROPRIETARY -** The driver does not generate an NTLM or Kerberos token. You must supply this token in the Authorization header of the HTTP request.	BASIC
Proxy Auto Detect	Specifies whether to use the system proxy settings or not. The value set for this option takes precedence over other proxy settings. So, if you want to use custom proxy settings, disable this option.	
Proxy User	The user name to authenticate to the Proxy Server.	
Proxy Password	The password to be used with the	

Setting	Description	Example
	Proxy User to authenticate to the Proxy Server.	
Proxy Server	The hostname or IP address of a proxy to route HTTP traffic through.	
Proxy Port	The TCP port the Proxy Server is running on.	
Proxy SSL Type	The SSL type to use when connecting to the ProxyServer proxy. The options available are as follows: ◦ **AUTO -** If the URL is an HTTPS URL, the driver will use the TUNNEL option. If the URL is an HTTP URL, the component will use the NEVER option. ◦ **ALWAYS -** The connection is always SSL enabled. ◦ **NEVER -** The connection is not SSL enabled. ◦ **TUNNEL -** The connection is through a tunneling proxy. This option is enabled only when you provide the value for the **Proxy Server**.	AUTO

Connecting to Qualys

Qualys is a cloud-based suite of security and compliance solutions that help organizations to simplify their security operations. Use the Qualys data connector to import your company's Qualys data.

Before you start

To connect to Qualys, you must gather the following:

- the Qualys server's host name or IP address
- user name and password
- the import mode to import data from Qualys
- the Asset(s) being scanned
- unique scan reference ID for a specific scan

For help gathering the connection prerequisites, contact the Qualys administrator in your organization. If your administrator cannot help you, you or your administrator should contact Qualys Support.

Create a Qualys connection

1. From the Analytics main menu, select **Import > Database and application**.
2. From the **New Connections** tab, in the **ACL Connectors** section, select **Qualys.**

   > **Tip**
   >
   > You can filter the list of available connectors by entering a search string in the **Filter connections** box. Connectors are listed alphabetically.

3. In the **Data Connection Settings** panel, enter the connection settings and at the bottom of the panel, click **Save and Connect**.

   You can accept the default **Connection Name**, or enter a new one.

The connection for Qualys is saved to the **Existing Connections** tab. In the future, you can reconnect to Qualys from the saved connection.

Once the connection is established, the Data Access window opens to the **Staging Area** and you can begin importing data. For help importing data from Qualys, see "Working with the Data Access window" on page 357.

Connection settings

Basic settings

Setting	Description	Example
Server	The host name or IP address of the Qualys server.	
User	The Qualys user account to authenticate against the Qualy server.	
Password	The password used to authenticate the user with the server.	
Mode	Specifies the import mode to use when importing data from Qualys. The options available are: ◦ **Import Vulnerabilities by Asset Group -** Allows you to import data from all possible Asset Groups linked to the Qualys account, along with vulnerab-ilities results. ◦ **Import KnowledgeBase -** Allows you to import KnowledgeBase data, filtered by Vulnerabilities that are modified and/or published within a given date range. ◦ **Import scan by scan reference id -** Allows you to specify a scan ID linked to a Qualys account and import that scan result into Analytics.	Import Vulnerabilities by Asset Group
Asset	Allows you to specify the Asset(s) to scan and import data from. To select an Asset, you must provide values for the **Server**, **User**, and **Password** fields and select **Import Vulnerabilities by Asset Group** in the **Mode** field. You must click the **Select Asset** button to get the list of assets and to select multiple assets, you can use the CTRL key.	

Setting	Description	Example
Filter	Allows you to select a default or custom date filter. The options available are: ◦ **Last 7 days** ◦ **Last 14 days** ◦ **Last 30 days** ◦ **Last 90 days** ◦ **Date Processed**	Last 7 days
Processed After	Allows you to import data processed after a specified date. This field is enabled only when **Date Processed** is selected in the Filter field.	2019-08-17
Processed Before	Allows you to import data processed before a specified date. This field is enabled when you select **Date Processed** in the Filter field.	
Filter By	To filter data by **Date Published** or **Date Modified**. This field is enabled when you select **Import KnowledgeBase** in the Mode field.	**Date Published**
Before	To filter the output to show vulnerabilities published or modified before a certain date.	
After	To filter the output to show vulnerabilities published or modified after a certain date.	
Scan Reference ID	Allows you to specify the scan to import. To select a scan, you must provide values for the **Server**, **User**, and **Password** fields and select **Import scan by scan reference id** in the **Mode** field. You must click the **Select Scan**button to get the list of scans available.	

Advanced settings

Setting	Description	Example
Enable Proxy	Specifies whether to use a proxy server.	
Proxy User	The user name to authenticate to the proxy server.	john_doe@example.com
Proxy Password	The password to be used with the Proxy User to authenticate to the proxy server.	
Proxy Server	The hostname or IP address of a proxy server.	206.174.193.116
Proxy Port	The port the proxy server is running on.	

Connecting to QuickBooks

QuickBooks is online accounting software that enables you to create invoices, manage expenses, and analyze your transaction details. You can use the QuickBooks data connector to import data your organization's QuickBooks data.

> **Note**
> The QuickBooks data connector is provided by our data partner, CData. For information on any of the connection fields, refer the documentation available at the CData website.

Before you start

To connect to QuickBooks, you must gather the credentials to connect to the QuickBooks account.

For help gathering the connection prerequisites, contact the QuickBooks administrator in your organization. If your administrator cannot help you, you or your administrator should contact QuickBooks Support.

Create a QuickBooks connection

1. From the Analytics main menu, select **Import > Database and application**.
2. From the **New Connections** tab, in the **ACL DSN Connectors (Bundled)** section, select **QuickBooks.**

   > **Tip**
   > You can filter the list of available connectors by entering a search string in the **Filter connections** box. Connectors are listed alphabetically.

   The DSN Configuration dialog box appears.
3. In the **DSN Configuration** dialog box, click **Show Required**.
4. Provide values for the required fields, if any.

5. Click **Test Connection**.

> **Note**
> When you click the **Test Connection** button, it validates whether the connection and authentication details provided are correct. If you click **OK** without clicking Test Connection, the connection details are saved without any validation and may not work later if the values provided are incorrect.

   The log in page for QuickBooks appears.

6. Provide the connection details and authenticate your login.

   When the connection is successful, a dialog box opens displaying the success message.

7. Click **OK** in the connection successful dialog box that appears.

   The connection to QuickBooks is established successfully.

8. In the **DSN Configuration** dialog box, click **OK**.

The connection for QuickBooks is saved to the **Existing Connections** tab. In the future, you can reconnect to QuickBooks from the saved connection.

Once the connection is established, the Data Access window opens to the **Staging Area** and you can begin importing data. For help importing data from QuickBooks, see "Working with the Data Access window" on page 357.

Rename the connection

When you create the QuickBooks data connector, you cannot provide a Connection name of your choice and the connection is saved to the **Existing Connections** tab with the same name as your data source. If you want to rename it, navigate to the **Existing Connections** tab, expand the **ACL Connectors (DSN)** section, click the ellipsis icon corresponding to QuickBooks and select **Rename connection**.

Connecting to QuickBooks Online

QuickBooks Online is a cloud-based accounting software solution for small and mid-sized businesses. You can use the QuickBooks Online data connector to import data your organization's QuickBooks Online data.

> **Note**
> The QuickBooks Online data connector is provided by our data partner, CData. For information on any of the connection fields, refer the documentation available at the CData website.

Before you start

To connect to QuickBooks Online, you must gather the following information:

- whether you are using a sandbox account
- the client ID and client secret assigned when you register with an OAuth authorization server

For help gathering the connection prerequisites, contact the QuickBooks Online administrator in your organization. If your administrator cannot help you, you or your administrator should contact QuickBooks Online Support.

Create a QuickBooks Online connection

1. From the Analytics main menu, select **Import > Database and application**.
2. From the **New Connections** tab, in the **ACL DSN Connectors (Bundled)** section, select **QuickBooks Online.**

   > **Tip**
   > You can filter the list of available connectors by entering a search string in the **Filter connections** box. Connectors are listed alphabetically.

   The DSN Configuration dialog box appears.
3. In the **DSN Configuration** dialog box, click **Show Required**.
4. Provide values for the required fields, if any.

5. Click **Test Connection**.

> **Note**
> When you click the **Test Connection** button, it validates whether the connection and authentication details provided are correct. If you click **OK** without clicking Test Connection, the connection details are saved without any validation and may not work later if the values provided are incorrect.

   The log in page for QuickBooks Online appears.

6. Provide the connection details and authenticate your login.

   When the connection is successful, a dialog box opens displaying the success message.

7. Click **OK** in the connection successful dialog box that appears.

   The connection to QuickBooks Online is established successfully.

8. In the **DSN Configuration** dialog box, click **OK**.

The connection for QuickBooks Online is saved to the **Existing Connections** tab. In the future, you can reconnect to QuickBooks Online from the saved connection.

Once the connection is established, the Data Access window opens to the **Staging Area** and you can begin importing data. For help importing data from QuickBooks Online, see "Working with the Data Access window" on page 357.

Rename the connection

When you create the QuickBooks Online data connector, you cannot provide a Connection name of your choice and the connection is saved to the **Existing Connections** tab with the same name as your data source. If you want to rename it, navigate to the **Existing Connections** tab, expand the **ACL Connectors (DSN)** section, click the ellipsis icon corresponding to QuickBooks Online and select **Rename connection**.

Connecting to QuickBooks POS

QuickBooks Point of Sale (POS) is a point of sale system for small to mid-sized businesses that enables users to track sales, customers and manage inventory. You can use the QuickBooks POS data connector to import data your organization's QuickBooks POS data.

> **Note**
> The QuickBooks POS data connector is provided by our data partner, CData. For information on any of the connection fields, refer the documentation available at the CData website.

Before you start

To connect to QuickBooks POS, you must gather the following information:

- the URL for the Remote Connector
- the credentials to connect to the Remote Connector

For help gathering the connection prerequisites, contact the QuickBooks POS administrator in your organization. If your administrator cannot help you, you or your administrator should contact QuickBooks POS Support.

Create a QuickBooks POS connection

1. From the Analytics main menu, select **Import > Database and application**.
2. From the **New Connections** tab, in the **ACL DSN Connectors (Bundled)** section, select **QuickBooks POS.**

   > **Tip**
   > You can filter the list of available connectors by entering a search string in the **Filter connections** box. Connectors are listed alphabetically.

   The DSN Configuration dialog box appears.
3. In the **DSN Configuration** dialog box, click **Show Required**.
4. Provide values for the required fields, if any.

5. Click **Test Connection**.

> **Note**
> When you click the **Test Connection** button, it validates whether the connection and authentication details provided are correct. If you click **OK** without clicking Test Connection, the connection details are saved without any validation and may not work later if the values provided are incorrect.

   The log in page for QuickBooks POS appears.

6. Provide the connection details and authenticate your login.

   When the connection is successful, a dialog box opens displaying the success message.

7. Click **OK** in the connection successful dialog box that appears.

   The connection to QuickBooks POS is established successfully.

8. In the **DSN Configuration** dialog box, click **OK**.

The connection for QuickBooks POS is saved to the **Existing Connections** tab. In the future, you can reconnect to QuickBooks POS from the saved connection.

Once the connection is established, the Data Access window opens to the **Staging Area** and you can begin importing data. For help importing data from QuickBooks POS, see "Working with the Data Access window" on page 357.

Rename the connection

When you create the QuickBooks POS data connector, you cannot provide a Connection name of your choice and the connection is saved to the **Existing Connections** tab with the same name as your data source. If you want to rename it, navigate to the **Existing Connections** tab, expand the **ACL Connectors (DSN)** section, click the ellipsis icon corresponding to QuickBooks POS and select **Rename connection**.

Connecting to REST Data Services

***RE**presentational **S**tate **T**ransfer* (REST) is an architectural style for standardizing communication between computer systems on the web. Rest-compliant, or RESTful, systems are able to communicate with each other easily.

You can load data from RESTful systems into Analytics by hitting endpoints that use GET methods. Other types of HTTP connections like POST and PUT, which are designed to send data to a resource, are not supported.

> **Caution**
>
> In Analytics 15, the REST connector is enhanced to support more authentication methods and corresponding connection fields are also updated. However, scripts from previous versions will not work in this version after upgrade. If you upgrade to the Analytics 15 version, you must re-configure the connector to connect to a RESTful system. If you have previous versions of REST connector scripts running in Robots or Analytics Exchange, you must upload the re-configured scripts after upgrading to Analytics 15.

Before you start

To connect Analytics to a RESTful system, you need the following:

- The URI endpoint of the RESTful system you want to connect to.
- Connection credentials for that system, if they are required. In some cases, that will simply be a username and password. Some systems use more substantial credentials like OAuth. If you are not sure what credentials you need, contact the administrator of your REST service in your company. If your administrator cannot help you, consult the support services or help content for the system you want to connect to.

Create a REST connection

1. From the Analytics main menu, select **Import > Database and application**.
2. From the **New Connections** tab, in the **ACL Connectors** section, select **REST.**

> **Tip**
>
> You can filter the list of available connectors by entering a search string in the **Filter connections** box. Connectors are listed alphabetically.

3. In the **Data Connection Settings** panel, enter the connection settings and at the bottom of the panel, click **Save and Connect**.

   You can accept the default **Connection Name**, or enter a new one.

The connection for REST is saved to the **Existing Connections** tab. In the future, you can reconnect to REST from the saved connection.

Once the connection is established, the Data Access window opens to the **Staging Area** and you can begin importing data. For help importing data from REST, see "Working with the Data Access window" on page 357.

Connection settings

Basic settings

Setting	Description	Example
URI	The Uniform Resource Identifier (URI) or absolute file path of the RESTful resource.	https://jsonplaceholder.typicode.com/users/
Format	Specifies whether the data is in XML or JSON format.	JSON
Authorization Type	The scheme used for authentication. Authentication types available are as follows: ◦ **No Auth** - The driver uses no authentication. ◦ **API Key** - The driver uses an API key for authentication. ◦ **Bearer Token** - The driver uses a bearer token for authentication. ◦ **Basic Auth** - The driver uses a basic authentication with a username and password. ◦ **OAuth 1.0** - The	No Auth

Setting	Description	Example
	driver uses OAuth 1.0 for authentication. ◦ **OAuth 2.0** – The driver uses OAuth 2.0 for authentication.	
Key	Specifies the API key to authenticate. This field is enabled when you select **API Key** in the **Authorization Type** field.	
Value	The value in this field is used with the API Key provided to authenticate to the server. This field is enabled when you select **API Key** in the **Authorization Type** field.	
Add To	Specifies whether to append the API key-value pair in the request header or query parameters.	Header
Token	Specifies the bearer token to authenticate to the server. The token can be an access key, such as a JSON Web Token (JWT) that is included in the request header. This field is enabled when you select **Bearer Token** in the **Authorization Type** field.	
User	The username that will be used to connect to a	jgibbons

Setting	Description	Example
	REST data source. This field is enabled when you select **Basic Auth** in the **Authorization Type** field.	
Password	The password for the username to connect to a REST data source. This field is enabled when you select **Basic Auth** in the **Authorization Type** field.	
OAuth Authorization URL	The authorization URL for the OAuth service.	https://login.example.com/services/oauth2/authorize
OAuth Access Token URL	The URL to retrieve the OAuth access token from. In OAuth 1.0, the authorized request token is exchanged for the access token at this URL.	https://login.example.com/services/oauth2/access
OAuth Request Token URL	The URL to retrieve request tokens from.	https://login.example.com/services/oauth2/token
OAuth Client Id	The client ID assigned when you register your REST data source with an OAuth authorization server.	ZYDPLLBWSK3MVQJSIYHB1OR2JXCY0X2C5UJ2QAR2MAAIT5Q
OAuth Client Secret	The client secret assigned when you register your REST data source with an OAuth authorization server.	
OAuth Refresh Token URL	The URL to refresh the OAuth token from. In OAuth 2.0, this URL is where	https://login.example.com/services/oauth2/refresh

Setting	Description	Example
	the refresh token is exchanged for a new access token when the old access token expires.	
OAuth Grant Type	The grant type for the OAuth flow. Options available are as follows: ◦ **CODE** ◦ **CLIENT** ◦ **PASSWORD**	CODE
Callback URL	The OAuth callback URL to return to when authenticating. This value must match the callback URL you specify in your app settings.	https://www.example.com/api/billing/123

Advanced settings

Setting	Description	Example
Access Key	Your AWS account access key. This value is accessible from your AWS security credentials page.	AKIAIOSFODNN7EXAMPLE
Secret Key	Your AWS account secret key. This value is accessible from your AWS security credentials page.	wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY
Region	The hosting region for your Amazon Web Services.	NORTHERNCALIFORNIA
Other	Hidden properties needed only in specific use cases. Normally, you do not need to enter anything here. Specify multiple properties in a semicolon-separated list. Caching Configuration ◦ **CachePartial=True** - Caches only a subset of columns, which you can specify in your query.	CachePartial=True; QueryPassthrough=True

Setting	Description	Example
	◦ **QueryPassthrough=True** - Passes the specified query to the cache database instead of using the SQL parser of the driver. Integration and Formatting ◦ **SupportAccessLinkedMode** - In Access' linked mode, it is generally a good idea to always use a cache as most data sources do not support multiple Id queries. However if you want to use the driver in Access but not in linked mode, this property must be set to False to avoid using a cache of a SELECT * query for the given table. ◦ **DefaultColumnSize** -Sets the default length of string fields when the data source does not provide column length in the metadata. The default value is 2000. ◦ **ConvertDateTimeToGMT** - Whether to convert date-time values to GMT (UTC), instead of the local time of the machine. ◦ **RecordToFile=filename** - Records the underlying socket data transfer to the specified file. OAuth Properties ◦ **InitiateOAuth** -Set this property to initiate the process to obtain or refresh the OAuth access token when you connect. The following options are available: • **OFF** - Indicates that the OAuth flow will be handled entirely by you. An OAuthAccessToken will be required to authenticate. • **GETANDREFRESH** - Indicates that the entire OAuth flow will be handled by the driver. If no token currently exists, it will be obtained by prompting you	

Setting	Description	Example
	via the browser. If a token exists, it will be refreshed when applicable. • **REFRESH** - Indicates that the driver will only handle refreshing the OAuthAccessToken. You will never be prompted by the driver to authenticate via the browser. You must handle obtaining the OAuthAccessToken and OAuthRefreshToken initially. ◦ **OAuthSettingsLocation** -The location of the settings file where OAuth values are saved when InitiateOAuth is set to true. When InitiateOAuth is enabled, the driver saves OAuth values to a settings file to avoid requiring the user to manually enter OAuth connection properties. When your OAuth session expires, the driver will automatically obtain a new access token if InitiateOAuth is set. If InitiateOAuth is enabled but OAuthSettingsLocation is not defined, the driver uses a default settings file, %AppData%\CData\REST Data Provider-\OAuthSettings.txt on Windows. On macOS, this file is located in ~/Library/Application Support/CData/REST Data Provider/OAuthSettings.txt. On Linux, ~/cdata/.config. ◦ **OAuthAcccessToken** -The OAuthAccessToken is retrieved from the OAuth server as part of the authentication process. It has a server-dependent timeout and can be reused between requests.The access token is used in place of your username and password. The access token protects your credentials by keeping them on the server.	

Setting	Description	Example
	◦ **OAuthAccessTokenSecret** - The OAuthAccessTokenSecret is retrieved from the OAuth server as part of the authentication process. It is used with the OAuthAccessToken and can be used for multiple requests until it times out. ◦ **OAuthRefreshToken** -The OAuthRefreshToken property is used to refresh the OAuthAccessToken when using OAuth authentication.	
Data Model	Specifies the data model to use when parsing JSON or XML documents and generating the metadata. ◦ **Document** - A single table representing a row per document will be returned. In this data model, any nested documents (object arrays) will not be flattened and will be returned as aggregates. Unless a XPath value is explicitly specified, the driver will identify and use the top-most document (object array) found as the XPath. ◦ **FlattenedDocuments** - A single table representing a JOIN of the available documents in the JSON or XML file will be returned. In this data model, nested XPath values will act in the same manner as a SQL JOIN. Additionally nested sibling XPath values (child paths at the same height), will be treated as a SQL CROSS JOIN. Unless explicitly specified, the driver will identify the XPath values available by parsing the JSON or XML file and identifying the available documents (including nested documents). ◦ **Relational** - Multiple tables will be returned, one for each XPath value specified. In this	Document

Setting	Description	Example
	data model, any nested documents (object arrays) will be returned as relational tables with primary and foreign keys. Unless explicitly specified, the driver will identify the XPath values available by parsing the JSON or XML file and identifying the available documents (including nested documents).	
Data Source	This property specifies a URI for the REST resource location.	s3://remotePath/file.json
Flatten Arrays	By default, nested arrays are returned as strings of JSON or XML. The FlattenArrays property can be used to flatten the elements of nested arrays into columns of their own. Set FlattenArrays to the number of elements you want to return from nested arrays. Setting FlattenArrays to -1 will flatten all the elements of nested arrays.	1
Flatten Objects	Set FlattenObjects to true to flatten object properties into columns of their own. Otherwise, objects nested in arrays are returned as strings of JSON or XML.	true
JSON Format	Specifies the format of the JSON document.	JSON
Generate Schema Files	Specifies whether to generate a schema file (RSD) from the parsed document. ◦ **Never** - A schema file (RSD) will never be generated. ◦ **OnUse** - Aschema file (RSD) will be generated the first time a table is referenced, provided the schema file (RSD) for the table does not already exist. ◦ **OnStart** - A schema file (RSD) will be generated at connect time for any tables that do not currently have a schema file (RSD).	Never

Setting	Description	Example
	This property is used in conjuction with Format, URI, XPath, and Location.	
XPath	The XPath of an element that repeats at the same height within the XML/JSON document (used to split the document into multiple rows). Multiple paths can be specified using a semi-colon separated list. The DataModel setting allows you to configure how the XPath values will be used to create tables and display data.	$.store.book[0].title
Kerberos Keytab File	The Keytab file containing your pairs of Kerberos principals and encrypted keys.	/path_to_keytab_file/filename.keytab
Kerberos SPN	If the service principle name (SPN) on the Kerberos Domain Controller is not the same as the URL that you are authenticating to, use this property to set the SPN.	HTTP/TimeOffWebPortal
Row Scan Depth	The number of rows to scan when dynamically determining columns for the table. Columns are dynamically determined when a schema (RSD) file is not available for the table, such as when using GenerateSchemaFiles. Higher values will result in a longer request, but will be more accurate. Setting this value to 0 (zero) will parse the entire document.	100
SSL Server Cert	If using a TLS/SSL connection, this property can be used to specify the TLS/SSL certificate to be accepted from the server. Any other certificate that is not trusted by the machine will be rejected. If not specified, any certificate trusted by the machine will be accepted. Use '*' to signify to accept all certificates (not recommended for	◦ -----BEGIN CERTIFICATE----- MIIChTCCAe4CAQAwDQYJKoZIhv.....Qw== ----- END CERTIFICATE----- ◦ C:\cert.cer ◦ -----BEGIN RSA PUBLIC KEY----- MIGfMA0GCSq.....AQAB -----END RSA PUBLIC KEY----- ◦ ecadbdda5a1529c58a1e9e09828d70e4 ◦ 34a929226ae0819f2ec14b4a3d904f801cbb150d

Setting	Description	Example
	security concerns).	
Limit Key Size	The maximum length of a primary key column. In some ODBC tools, the length of the primary key column cannot be larger than a specific value. This property makes the ODBC Driver override the reported length of all the primary key columns. It is especially useful when using the ODBC Driver as a Microsoft Access Linked Data Source. Setting the LimitKeySize to zero will make the key length revert to the original length.	255
Map To Long Varchar	This property controls whether or not a column is returned as SQL_LONGVARCHAR. Some applications require all text data larger than a certain number of characters to be reported as SQL_LONGVARCHAR. Use this property to map any column larger than the specified size so they are reported as SQL_LONGVARCHAR instead of SQL_VARCHAR.	-1
Map To WVarchar	This property controls whether or not string types map to SQL_WVARCHAR instead of SQL_VARCHAR. It is set by default.	true
Pseudo Columns	Indicates whether or not to include pseudo columns as columns to the table. This setting is particularly helpful in Entity Framework, which does not allow you to set a value for a pseudo column unless it is a table column. You can use an asterisk (*) character to include all tables and all columns. For example: *=*.	Table1=Column1, Table1=Column2, Table2=Column3
Upper Case Identifiers	Report all identifiers in uppercase. This is the default for Oracle	false

Setting	Description	Example
	databases and thus allows better integration with Oracle tools such as the Oracle Database Gateway.	
Proxy Auth Scheme	This value specifies the authentication type to use to authenticate to the HTTP proxy specified by ProxyServer and ProxyPort. Note that the driver will use the system proxy settings by default, without further configuration needed; if you want to connect to another proxy, you will need to set ProxyAutoDetect to false, in addition to ProxyServer and ProxyPort. To authenticate, set ProxyAuthScheme and set ProxyUser and ProxyPassword, if needed. The authentication type can be one of the following: ◦ **BASIC** - The driver performs HTTP BASIC authentication. ◦ **DIGEST** -The driver performs HTTP DIGEST authentication. ◦ **NONE** ◦ **NEGOTIATE** -The driver retrieves an NTLM or Kerberos token based on the applicable protocol for authentication. ◦ **NTLM** ◦ **PROPRIETARY** -The driver does not generate an NTLM or Kerberos token. You must supply this token in the Authorization header of the HTTP request.	BASIC
Proxy Auto Detect	This indicates whether to use the system proxy settings or not. Set ProxyAutoDetect to FALSE to use custom proxy settings. This takes precedence over other proxy settings.	true
Proxy User	A user name to be used to authenticate to the ProxyServer proxy.	jgibbons
Proxy Password	A password to be used to authenticate to the ProxyServer proxy.	UsaPhone897Batteries!Tokyo

Setting	Description	Example
Proxy Server	The hostname or IP address of a proxy to route HTTP traffic through.	192.168.1.100
Proxy Port	The TCP port the ProxyServer proxy is running on.	80
Proxy SSL Type	The SSL type to use when connecting to the ProxyServer proxy. ◦ **AUTO** -Default setting. If the URL is an HTTPS URL, the driver will use the TUNNEL option. If the URL is an HTTP URL, the component will use the NEVER option. ◦ **ALWAYS** -The connection is always SSL enabled. ◦ **NEVER** -The connection is not SSL enabled. ◦ **TUNNEL** -The connection is through a tunneling proxy: The proxy server opens a connection to the remote host and traffic flows back and forth through the proxy.	AUTO
Proxy Exception	A semicolon separated list of hosts or IPs that will be exempt from connecting through the ProxyServer . The driver will use the system proxy settings by default, without further configuration needed. If you want to explicitly configure proxy exceptions for this connection, you will need to set ProxyAutoDetect to false, and configure ProxyServer and ProxyPort. To authenticate, set ProxyAuthScheme and set ProxyUser and ProxyPassword, if needed.	172.16.254.1;192.0.2.1

Connecting to Rsam

In Rsam, you can create saved searches that allow you to store results and retrieve them when you need them. You can import those saved searches into Analytics. Once you bring this data into Analytics, you can prepare it, analyze it, and move it into Results to create and share storyboards and visualizations about your data.

Before you import

Note the following requirements and considerations before you bring data into Analytics.

First, integrate Rsam

An Rsam administrator must complete the Rsam integration process before anyone can use the Rsam connector. For more information, see Rsam integration with HighBond and ACL Robotics.

Rsam version requirement

Your company needs to be on Rsam Cloud version 9.2.2210 or later. If you are on an earlier version of Rsam, or an on-premise edition, Analytics cannot access data from Rsam.

Permissions

Whether you authenticate with a user name and password or an API key, the saved searches available to you in Analytics are the same as the saved searches available to you in Rsam. Your access is based on your permissions and roles in Rsam. If you do not have access to the data you need to import, contact your company's Rsam administrator, as only he or she can adjust what you can access.

Set up your saved searches in Rsam

Before proceeding, ensure the saved searches you want to bring into Analytics are set up correctly. For help with these tasks, see Rsam's end-user help and admin help.

- Create or ensure you have access to the saved searches you need. Saved searches being brought into Analytics or HighBond should be created for that purpose alone. If you want to use an existing saved search, save a copy of it and use that copy. This ensures that another user cannot modify your search and unintentionally break any scripts you create to automate the

data transfer between platforms.
- Ensure the "HighBond Platform" option is checked for each search you want to bring into Analytics.
- Remove groupings in your saved search so your import is "flat". If you want to group your records together, do so when creating visualizations and storyboards after you bring the data into Analytics and/or HighBond.
- You do not need to perfect your Rsam data before importing. Analytics gives you the ability to filter, clean, combine, and standardize the data you import. However, it is generally good practice to work with the best data available to you and minimize transformations. You may want to take some time and ensure your saved searches in Rsam yield all the data you want and omit data that is irrelevant to your analysis. Doing this can make the data preparation process easier.

How Analytics determines column names

Objects and records in Rsam have a display name and an admin name. Typically, Rsam users see the display name throughout that platform, and the admin name is a hidden field only used by Rsam admins to maintain the Rsam environment.

By default, Analytics takes Rsam's display names as its field names since this is what most people are used to seeing. However, Analytics has strict field name requirements. Spaces and special characters are converted to underscores in Analytics, meaning the column names in Analytics probably will not be an exact match to the column names in the Rsam saved search. Generally, this will not cause issues, but it is possible to change what you see in Analytics.

- If you want to use different physical field names, you can change the SQL query Analytics makes to Rsam and specify your own field names. However, you are still subject to Analytics field name requirements, and you will need to keep track of this query if you import updated data in the future.
- If you want to keep the default field names, but change the column headers that actually appear in Analytics, you can change the column view names to something more meaningful to you without affecting Analytics' actual field name.

Create an Rsam connection

1. From the Analytics main menu, select **Import > Database and application**.
2. From the **New Connections** tab, in the **ACL Connectors** section, select **Rsam.**

   > **Tip**
   > You can filter the list of available connectors by entering a search string in the **Filter connections** box. Connectors are listed alphabetically.

3. In the **Data Connection Settings** panel, enter the connection settings and at the bottom of the panel, click **Save and Connect**.

   You can accept the default **Connection Name**, or enter a new one.

The connection for Rsam is saved to the **Existing Connections** tab. In the future, you can reconnect to Rsam from the saved connection.

Once the connection is established, the Data Access window opens to the **Staging Area** and you can begin importing data. For help importing data from Rsam, see "Working with the Data Access window" on page 357.

Connection settings

Basic settings

Setting	Description	Example
Host	The Rsam instance you are connecting to.	https://training02.rsam.com/
API App Name	Your Rsam API app name. By default, this is "rsam_api", but it is possible to change this in Rsam.	rsam_api
Authentication Method	Whether you are connecting to Rsam with a user name and password, or an API key.	Password
User	If you are authenticating to Rsam with a username and password, enter your username here.	jgibbons
Password	If you are authenticating to Rsam with a username and password, enter your password here.	UsaPhone897Batteries!Tokyo
API Key	If you are authenticating to Rsam with an API key, enter it here. Your API key is specific to you, but if you do not already have one, your Rsam administrator must generate it for you in Rsam.	99141fae-4c41-4abd-ade2-469f7d0151a4

Advanced settings

Setting	Description	Example
Convert Datetime to GMT	Converts datetime fields to GMT time zone during import. If false, the datetime value is converted to the operating system time zone of the machine running Analytics.	True

What comes next?

Once your Rsam data is in Analytics, you have many options to prepare and then analyze that data.

- **Prepare** – Use the tools in Analytics to clean and standardize your data, combine it with data from other sources, and prepare it for analysis. Think about the ways your data can tell a more comprehensive story.
  - Can you learn more about your company's IT risk posture?
  - Are there vendors in your ERP system that are not in your Rsam vendor management program?
  - Are there employees in your HR database that are not in Rsam's policy management campaigns?
- Collecting and preparing all your data can help you answer questions like these. For more information, see "Preparing data for analysis" on page 794.
- **Analyze** –Use Analytics commands and functions to analyze your data. For more information, see "Analyzing data" on page 1090.
- **Export** –You can export your data to other formats, including to HighBond's Results app. For more information, see "Exporting data" on page 203 and "Exporting exceptions to HighBond Results" on page 208.
- **Automate** –You can write scripts and use HighBond's Robots app to automate repetitive tasks like importing, aggregating, and exporting data to Results. As your Rsam (and other) data changes over time, your storyboards in HighBond can reflect this automatically.
- **Visualize** –Once you bring your data into Results, use Results to build meaningful visualizations. You can turn these into and storyboards that you can easily share with others in your company.

Connecting to RSS/ATOM

Really Simple Syndication(RSS) and Atom are XML-based format/feed language used for publishing news or articles on website in a computer-readable form. An RSS or Atom file is called a feed and is compatible with most consumer feed readers. You can use the RSS/ATOM data connector to import your company's RSS/ATOM data.

Before you start

To connect to RSS/ATOM, you must gather the following:

- whether the feed type is RSS or Atom
- the URI of the feed

For help gathering the connection prerequisites, contact the RSS/ATOM administrator in your organization. If your administrator cannot help you, you or your administrator should contact RSS/ATOM Support.

Create an RSS/ATOM connection

1. From the Analytics main menu, select **Import > Database and application**.
2. From the **New Connections** tab, in the **ACL Connectors** section, select **RSS/ATOM.**

> **Tip**
> You can filter the list of available connectors by entering a search string in the **Filter connections** box. Connectors are listed alphabetically.

3. In the **Data Connection Settings** panel, enter the connection settings and at the bottom of the panel, click **Save and Connect**.

   You can accept the default **Connection Name**, or enter a new one.

The connection for RSS/ATOM is saved to the **Existing Connections** tab. In the future, you can reconnect to RSS/ATOM from the saved connection.

Once the connection is established, the Data Access window opens to the **Staging Area** and you can begin importing data. For help importing data from RSS/ATOM, see "Working with the Data Access window" on page 357.

Connection settings

Basic settings

Setting	Description	Example
Type	Specifies the feed type. The options available are as follows: ◦ **RSS** ◦ **ATOM**	RSS
URI	The Uniform Resource Identifier (URI) for the feed resource location. This can be an http source or a file.	

Advanced settings

Setting	Description	Example
Auth Scheme	Specifies the scheme used for HTTP authentication. The options available are as follows: ◦ **NTLM -** To use your Windows credentials for authentication. ◦ **BASIC -** To use HTTP BASIC authentication. ◦ **DIGEST -** To use HTTP DIGEST authentication. ◦ **NONE -** To use anonymous authentication; for example, to access a public site. ◦ **KERBEROSDELEGATION -** To use delegation through the Kerberos protocol. Set the User and Password of the account you want to impersonate.	NONE
Auth Token	The token used for authentication to identify the validity of an HTTP request. The auth token set in the connection will be posted with the HTTP server as the request variable '@authtoken'.	
User	The user name to authenticate the	

Setting	Description	Example
	connection.	
Password	The password for the username to authenticate the connection.	
Limit Key Size	The maximum length of a primary key column.	255
Map To Long Varchar	This property controls whether a column is returned as SQL_ LONGVARCHAR or not.	-1
Map To WVarchar	This property controls whether string types map to SQL_ WVARCHAR instead of SQL_ VARCHAR.	
Pseudo Columns	This property indicates whether to include pseudo columns as columns to the table. The value of this connection setting is of the format "Table1=Column1, Table1=Column2, Table2=Column3".	
SSL Server Certificate	The certificate that must be accepted from the server when connecting using SSL. You can provide any of the following: ◦ a full PEM certificate ◦ path to a local file containing the certificate ◦ public key ◦ MD5 or SHA1 thumbprint (hex values can also be either space or colon separated) Any other certificate that is not trusted by the machine is rejected.	
Convert Datetime To GMT	Converts datetime fields to GMT time zone during import. If this option is disabled, the datetime value is converted to the operating system time zone of the system running Analytics.	
Proxy Authentication Scheme	Specifies the authentication scheme to authenticate to the Proxy Server. The options available are	BASIC

Setting	Description	Example
	as follows: ◦ **BASIC -** The driver performs HTTP BASIC authentication. ◦ **NONE -** The driver does not perform any authentication. ◦ **DIGEST -** The driver performs HTTP DIGEST authentication. ◦ **NEGOIATE -** The driver retrieves an NTLM or Kerberos token, based on the protocol set for authentication. ◦ **PROPRIETARY -** The driver does not generate an NTLM or Kerberos token. You must supply this token in the Authorization header of the HTTP request.	
Proxy Auto Detect	Specifies whether to use the system proxy settings or not. The value set for this option takes precedence over other proxy settings. So, if you want to use custom proxy settings, disable this option.	
Proxy User	The user name to authenticate to the Proxy Server.	
Proxy Password	The password to be used with the Proxy User to authenticate to the Proxy Server.	
Proxy Server	The hostname or IP address of a proxy to route HTTP traffic through.	
Proxy Port	The TCP port the Proxy Server is running on.	
Proxy SSL Type	The SSL type to use when connecting to the ProxyServer proxy. The options available are as follows: ◦ **AUTO -** If the URL is an HTTPS URL, the driver will use the TUNNEL option. If the URL is an HTTP URL, the component will use the NEVER option. ◦ **ALWAYS -** The connection is always SSL enabled. ◦ **NEVER -** The connection is not SSL enabled.	AUTO

Setting	Description	Example
	◦ **TUNNEL -** The connection is through a tunneling proxy. This option is enabled only when you provide the value for the **Proxy Server**.	

Connecting to Sage 50 UK

Sage 50 UK is an account management software that helps process financial data, trade in foreign currencies, and manage invoices and customers. You can use the Sage 50 UK data connector to import data your organization's Sage 50 UK data.

> **Note**
> The Sage 50 UK data connector is provided by our data partner, CData. For information on any of the connection fields, refer the documentation available at the CData website.

Before you start

To connect to Sage 50 UK, you must gather the following information:

- the URL to the Sage 50 UK SData service
- the credentials to connect to the Sage 50 UK account
- the authentication scheme used

For help gathering the connection prerequisites, contact the Sage 50 UK administrator in your organization. If your administrator cannot help you, you or your administrator should contact Sage 50 UK Support.

Create a Sage 50 UK connection

1. From the Analytics main menu, select **Import > Database and application**.
2. From the **New Connections** tab, in the **ACL DSN Connectors (Bundled)** section, select **Sage 50 UK.**

   > **Tip**
   > You can filter the list of available connectors by entering a search string in the **Filter connections** box. Connectors are listed alphabetically.

   The DSN Configuration dialog box appears.
3. In the **DSN Configuration** dialog box, click **Show Required**.
4. Provide values for the required fields, if any.

5. Click **Test Connection**.

> **Note**
> When you click the **Test Connection** button, it validates whether the connection and authentication details provided are correct. If you click **OK** without clicking Test Connection, the connection details are saved without any validation and may not work later if the values provided are incorrect.

   The log in page for Sage 50 UK appears.

6. Provide the connection details and authenticate your login.

   When the connection is successful, a dialog box opens displaying the success message.

7. Click **OK** in the connection successful dialog box that appears.

   The connection to Sage 50 UK is established successfully.

8. In the **DSN Configuration** dialog box, click **OK**.

The connection for Sage 50 UK is saved to the **Existing Connections** tab. In the future, you can reconnect to Sage 50 UK from the saved connection.

Once the connection is established, the Data Access window opens to the **Staging Area** and you can begin importing data. For help importing data from Sage 50 UK, see "Working with the Data Access window" on page 357.

Rename the connection

When you create the Sage 50 UK data connector, you cannot provide a Connection name of your choice and the connection is saved to the **Existing Connections** tab with the same name as your data source. If you want to rename it, navigate to the **Existing Connections** tab, expand the **ACL Connectors (DSN)** section, click the ellipsis icon corresponding to Sage 50 UK and select **Rename connection**.

Connecting to Sage Cloud Accounting

Sage Cloud Accounting is a cloud-based accounting software that helps organizations manage payments and invoices, payroll, and file taxes. You can use the Sage Cloud Accounting data connector to import data your organization's Sage Cloud Accounting data.

> **Note**
> The Sage Cloud Accounting data connector is provided by our data partner, CData. For information on any of the connection fields, refer the documentation available at the CData website.

Before you start

To connect to Sage Cloud Accounting, you must gather the connection details to the Sage Cloud Accounting server.

For help gathering the connection prerequisites, contact the Sage Cloud Accounting administrator in your organization. If your administrator cannot help you, you or your administrator should contact Sage Cloud Accounting Support.

Create a Sage Cloud Accounting connection

1. From the Analytics main menu, select **Import > Database and application**.
2. From the **New Connections** tab, in the **ACL DSN Connectors (Bundled)** section, select **Sage Cloud Accounting.**

   > **Tip**
   > You can filter the list of available connectors by entering a search string in the **Filter connections** box. Connectors are listed alphabetically.

   The DSN Configuration dialog box appears.
3. In the **DSN Configuration** dialog box, click **Show Required**.
4. Provide values for the required fields, if any.

5. Click **Test Connection**.

> **Note**
> When you click the **Test Connection** button, it validates whether the connection and authentication details provided are correct. If you click **OK** without clicking Test Connection, the connection details are saved without any validation and may not work later if the values provided are incorrect.

   The log in page for Sage Cloud Accounting appears.

6. Provide the connection details and authenticate your login.

   When the connection is successful, a dialog box opens displaying the success message.

7. Click **OK** in the connection successful dialog box that appears.

   The connection to Sage Cloud Accounting is established successfully.

8. In the **DSN Configuration** dialog box, click **OK**.

The connection for Sage Cloud Accounting is saved to the **Existing Connections** tab. In the future, you can reconnect to Sage Cloud Accounting from the saved connection.

Once the connection is established, the Data Access window opens to the **Staging Area** and you can begin importing data. For help importing data from Sage Cloud Accounting, see "Working with the Data Access window" on page 357.

Rename the connection

When you create the Sage Cloud Accounting data connector, you cannot provide a Connection name of your choice and the connection is saved to the **Existing Connections** tab with the same name as your data source. If you want to rename it, navigate to the **Existing Connections** tab, expand the **ACL Connectors (DSN)** section, click the ellipsis icon corresponding to Sage Cloud Accounting and select **Rename connection**.

Connecting to Sage Intacct

Sage Intacct is a cloud-based financial management and accounting software that helps automate accounting operations for small to mid-sized businesses. You can use the Sage Intacct data connector to import data your organization's Sage Intacct data.

> **Note**
> The Sage Intacct data connector is provided by our data partner, CData. For information on any of the connection fields, refer the documentation available at the CData website.

Before you start

To connect to Sage Intacct, you must gather the following information:

- the company ID of the user authenticating to Sage Intacct
- the URL to connect to Sage Intacct
- the credentials to connect to the Sage Intacct account
- the Web Services SenderID and password assigned to you by Sage Intacct

For help gathering the connection prerequisites, contact the Sage Intacct administrator in your organization. If your administrator cannot help you, you or your administrator should contact Sage Intacct Support.

Create a Sage Intacct connection

1. From the Analytics main menu, select **Import > Database and application**.
2. From the **New Connections** tab, in the **ACL DSN Connectors (Bundled)** section, select **Sage Intacct.**

   > **Tip**
   > You can filter the list of available connectors by entering a search string in the **Filter connections** box. Connectors are listed alphabetically.

   The DSN Configuration dialog box appears.
3. In the **DSN Configuration** dialog box, click **Show Required**.
4. Provide values for the required fields, if any.

5. Click **Test Connection**.

> **Note**
> When you click the **Test Connection** button, it validates whether the connection and authentication details provided are correct. If you click **OK** without clicking Test Connection, the connection details are saved without any validation and may not work later if the values provided are incorrect.

   The log in page for Sage Intacct appears.

6. Provide the connection details and authenticate your login.

   When the connection is successful, a dialog box opens displaying the success message.

7. Click **OK** in the connection successful dialog box that appears.

   The connection to Sage Intacct is established successfully.

8. In the **DSN Configuration** dialog box, click **OK**.

The connection for Sage Intacct is saved to the **Existing Connections** tab. In the future, you can reconnect to Sage Intacct from the saved connection.

Once the connection is established, the Data Access window opens to the **Staging Area** and you can begin importing data. For help importing data from Sage Intacct, see "Working with the Data Access window" on page 357.

Rename the connection

When you create the Sage Intacct data connector, you cannot provide a Connection name of your choice and the connection is saved to the **Existing Connections** tab with the same name as your data source. If you want to rename it, navigate to the **Existing Connections** tab, expand the **ACL Connectors (DSN)** section, click the ellipsis icon corresponding to Sage Intacct and select **Rename connection**.

Connecting to Salesforce

Salesforce.com is a cloud *Customer Relationship Management* (CRM) platform. You can use the Salesforce data connector to import your company's Salesforce data.

> **Note**
> Analytics provides Salesforce as an optional connector and if it is not available in your Data Access window, it is likely that the connector was not selected during installation. For more information, see "Install optional Analytics data connectors and Python engine" on page 2567.

Before you start

Salesforce credentials

To connect to Salesforce, you must gather the following:

- Salesforce username
- Salesforce password
- Security token

> **Note**
> Some connections require a security token, while others do not. You only require a security token if your connection fails without it.

For help gathering the connection prerequisites, contact the Salesforce administrator in your organization. If your administrator cannot help you, you or your administrator should contact Salesforce Support.

Salesforce API and editions

The connector uses the Salesforce API to access data, therefore you or your Salesforce administrator must enable API access for your company and user account in Salesforce before you connect. You must also have one of the following Salesforce editions:

- Developer Edition
- Professional Edition
- Enterprise Edition
- Unlimited Edition

> **Note**
> If you are an existing Salesforce customer and want to upgrade your account to one of these editions, contact your Salesforce account representative.

Create a Salesforce connection

1. From the Analytics main menu, select **Import > Database and application**.
2. From the **New Connections** tab, in the **ACL Connectors** section, select **Salesforce.**

   > **Tip**
   > You can filter the list of available connectors by entering a search string in the **Filter connections** box. Connectors are listed alphabetically.

3. In the **Data Connection Settings** panel, enter the connection settings and at the bottom of the panel, click **Save and Connect**.

   You can accept the default **Connection Name**, or enter a new one.

The connection for Salesforce is saved to the **Existing Connections** tab. In the future, you can reconnect to Salesforce from the saved connection.

Once the connection is established, the Data Access window opens to the **Staging Area** and you can begin importing data. For help importing data from Salesforce, see "Working with the Data Access window" on page 357.

Connection settings

Basic settings

Setting	Description	Example
User	The user name of the Salesforce account to authenticate to the server.	
Password	The password of the Salesforce account used to authenticate to the Salesforce server.	
Security Token	The security token to authenticate access to the Salesforce account. A security token can be obtained by going to your profile information and resetting the security token. If your password is reset, you will	

Setting	Description	Example
	also need to reset the security token.	
Proxy Server	The hostname or IP address of a proxy to route HTTP traffic through. The driver can use the HTTP, Windows (NTLM), or Kerberos authentication types to authenticate to an HTTP proxy.	
Proxy Port	The port the HTTP proxy is running on that you want to redirect HTTP traffic through.	80
Proxy User	The user name to be used to authenticate to the ProxyServer proxy.	
Proxy Password	The password to authenticate the Proxy User to the ProxyServer proxy.	

Advanced settings

Setting	Description	Example
Enable connection to sandbox URL	To specify if the connection should be made to a Salesforce sandbox account.	false
Sandbox URL		
SSL Server Certificate	The certificate to be accepted from the server when connecting using TLS/SSL. You can provide any of the following: ◦ a full PEM certificate ◦ path to a local file containing the certificate ◦ public key ◦ MD5 or SHA1 thumbprint (hex values can also be either space or colon separated) Any other certificate that is not trusted by the machine is rejected.	

Setting	Description	Example
Map To WVarchar	To specify whether string types map to SQL_WVARCHAR instead of SQL_VARCHAR. It is set by default. String columns must be mapped to SQL_WVARCHAR to accommodate various international character sets, so MapToWVarchar is set to true by default. You may set it to false to use SQL_VARCHAR instead.	true

Connecting to SAP

SAP is an enterprise-wide business suite for managing a broad range of business processes. Use the ACL Connector for SAP to import your company's SAP data.

> **Note**
>
> Setting up the SAP connector, and if applicable SNC (Secure Network Communications) and SSO (Single Sign-on), requires personnel with the appropriate level of technical expertise.
>
> The SAP connector requires a separate subscription entitlement beyond the basic Analytics subscription. If you do not have the required subscription entitlement, you are not able to connect to SAP.
>
> Contact your account representative for information about an SAP connector subscription.

Before you start

Compatible SAP systems

The ACL Connector for SAP can be used with the following SAP systems:

- S/4HANA
- all enhancement levels of the following systems, running on all supported database platforms including SAP HANA:
  - SAP ERP 6.0 (ECC 6.0)
  - SAP CRM 7.0
  - SAP SRM 7.0
  - SAP SCM 7.0
  - SAP EWM 7.0

Compatible SAP SPAM/SAINT version

For installing the SAP add-on for the SAP system, the required version of SAP SPAM/SAINT is 0053 or higher.

SAP SPAM is the Support Package Manager. SAP SAINT is the Add-On Installation Tool.

Install the SAP connector add-on for the SAP system

Your SAP Basis Administrator must install the SAP connector add-on for the SAP system. The add-on is required for the SAP connector to communicate with the SAP system. Users with an SAP connector subscription can download the add-on file from Launchpad (www.highbond.com).

Use the SAP Front End to install the SAP connector add-on. The compatible SAP Basis release for installing the SAP add-on is 700 or higher.

1. Sign in to HighBond (www.highbond.com).
2. In Launchpad, under **Resources**, click **Downloads**.
3. From the **ACL for Windows** top tab, under **Data Connector for SAP ERP**, click **Download Version 1.1**.

   The downloaded .zip file contains the following `.sar` files:

   - `DABEXP_SAPCONN_AOI_110.sar` - Installer file to install the SAP connector add-on.
   - `DABEXP_SAPCONN_AOU_110.sar` - Upgrade file to upgrade the existing SAP connector add-on from version 1.0.
4. On the SAP system, use the **SAINT** transaction to access the **Add-On Installation Tool**.
5. Use the **Add-On Installation Tool** to install or upgrade the SAP Add-on Installation Package.

SAP authorizations

> **Note**
> SAP authorizations must be assigned by your SAP Security Administrator.

Users of the SAP connector require the following SAP access and authorizations in order to connect to an SAP system and extract data:

- An SAP user ID and password that allows them to connect to the SAP system
- Specific SAP authorization objects and authorizations, including SAP table authorizations

SAP authorization objects

Users of the SAP connector require the specific SAP authorizations listed below.

> **Note**
> Consult your SAP security documentation for detailed information about assigning SAP authorizations to users.

Authorization class	Authorization object	Field	Values	Details
AAAB Cross-	**S_RFC** Authorization	**ACTVT**	16 (authorizes Execute)	Controls a user's ability to execute function modules on the SAP

Authorization class	Authorization object	Field	Values	Details
Application Authorization Objects	check for RFC access			system from a remote location, such as a desktop computer.
		RFC_NAME	/SDF/RI_CRM CMON RFC1 SYST /DABEXP/DAB_FUGR	
		RFC_TYPE	FUGR (Function Group)	
		Alternatively, as of **SAP NW 7.0 EHP 2** (SAP Basis 702), the RFC_TYPE can be set to FUNC (Function Module). With FUNC, function modules must be authorized directly and not via the function group.		
		ACTVT	16 (authorizes Execute)	
		RFC_NAME	/SDF/CMO_GET_INSTNO CMO_GET_INSTNO RFC_GET_FUNCTION_INTERFACE RFC_GET_NAMETAB RFCPING /DABEXP/RFC_SAPCONNECTOR	
		RFC_TYPE	FUNC (Function Module)	

Authorization class	Authorization object	Field	Values	Details
BC_A Basis: Administration	**S_TABU_DIS**	**ACTVT**	03 (Display)	Controls a user's access to specific groups of SAP tables.
		DICBERCLS (*Authorization Group*)	*	
	S_TABU_NAM	**ACTVT**	03 (Display)	Controls a user's access to individual SAP tables.
		TABLE (*Table Name*)	*	

> **Note**
> Users of the SAP connector should be assigned authorizations for those SAP tables they need to access in order to perform their analysis.
>
> For example, a user performing a General Ledger audit needs authorizations for the general ledger tables.
>
> Your company's own business processes dictate which users require table authorizations, and what authorizations they require. Work with your SAP Security Administrator to determine the appropriate level of access that your users require.

SAP port configuration

On each SAP server that you will connect to with the SAP connector, you need to open the following TCP/IP ports for inbound and outbound communication:

Port name	Port number	Comment
Dispatcher	32<*NN*>	Used by SAP GUI
Gateway	33<*NN*>	Used for RFC communication
Message Server	36<*NN*>	

> **Note**
> <*NN*> is the instance number of your SAP system. So, if the SAP system number is 10, then ports 3210, 3310 and 3610 need to be open.

Install SAP GUI for Windows

SAP GUI for Windows, version 7.60, or higher, must be installed on the local computer or the server where you intend to use Analytics's SAP connector. The SAP GUI allows the SAP connector to remotely access your SAP system.

Analytics users	The SAP GUI must be installed on the same computer as Analytics.
Robots users	The SAP GUI must be installed on the server that houses the Robots Agent.
Analytics Exchange users	The SAP GUI must be installed on the server that houses AX Server, and on any AX Engine Node servers that will be used to access your SAP system.

SAP connection information and credentials

Typically, an SAP Basis Administrator maintains connection information for an SAP system. Ask your SAP Basis Administrator to provide the necessary connection information, or have them configure the required connection information in the SAP Logon window.

If your administrator cannot help you, you or your administrator should contact SAP Support.

To connect to SAP, you or your administrator must gather the following:

- the correct server type (Normal, or Load Balanced)
- instance number (for Normal server type)
- logon group and system ID (for Load Balanced server type)
- the host name or IP address of the SAP system
- your SAP username and password
- client number
- read access to the tables in the SAP system

Create an SAP connection

1. From the Analytics main menu, select **Import > Database and application**.
2. From the **New Connections** tab, in the **ACL Connectors** section, select **SAP.**

> **Tip**
> You can filter the list of available connectors by entering a search string in the **Filter connections** box. Connectors are listed alphabetically.

3. In the **Data Connection Settings** panel, enter the connection settings and at the bottom of the panel, click **Save and Connect**.

   You can accept the default **Connection Name**, or enter a new one.

The connection for SAP is saved to the **Existing Connections** tab. In the future, you can reconnect to SAP from the saved connection.

Once the connection is established, the Data Access window opens to the **Staging Area** and you can begin importing data. For help importing data from SAP, see "Working with the Data Access window" on page 357.

Connection settings

> **Note**
> Consult your organization's SAP Basis Administrator for the required settings for the SAP connector.

SAP systems with Secure Network Communications (SNC)

Your organization's SAP system may have Secure Network Communications (SNC) enabled, which allows encrypted communication between the different components of an SAP system, and user authentication with Single Sign-on.

Additional settings are required in the ACL Connector for SAP if you are using SNC, with or without Single Sign-on.

SNC is a software layer in the SAP system that allows you to extend basic SAP security by integrating with an external security product.

Basic settings

Setting	Description	Example
Preloaded SAP Systems optional	The name of an SAP system with connection settings specified in one of the following SAP GUI configuration files: ◦ `SAPUILandscape.xml` ◦ `saplogon.ini` If you select a preloaded SAP system, values from the SAP GUI configuration file automatically populate a number of the other SAP connection settings for you. Default location of the SAP GUI configuration file: `C:\Users\<username>\AppData\Roaming\SAP\Common` If both configuration files are present in the `..\SAP\Common` folder, `SAPUILandscape.xml` takes precedence.	PHR - Production Human Resources
Server	The host name or IP address of the	◦ phr-1.example.com

Setting	Description	Example
	SAP system.	◦ 52.202.133.148
Client	The three-digit code identifying a client within the SAP system. A client is a sub-partition of an SAP system.	800
Language	The language of the SAP system.	EN
Server Type	The configuration of servers in the SAP system: ◦ **NORMAL** - a single server ◦ **LOAD BALANCE** - a load-balanced group of servers	◦ NORMAL ◦ LOAD BALANCE
Instance Number [Server Type = NORMAL]	The two-digit instance number of the SAP system. The instance number forms the last two digits of the port numbers used for inbound and outbound communication between the SAP connector and the SAP system. For more information, see "SAP port configuration" on page 598.	01
Logon Group [Server Type = LOAD BALANCE]	The name of a load-balanced group of servers in the SAP system.	ACL Production
System Id [Server Type = LOAD BALANCE]	The three-character identifier of the SAP system.	PHR
Authentication Method	The method of user authentication used by the SAP system: ◦ **Username and Password** • Standard SAP authentication method • Secure Network Communications (SNC) not enabled ◦ **Single Sign On** • SNC with single sign-on ◦ **SNC without Single Sign On** • SNC without single sign-on	
Authentication Method = Username and Password		
User	User's SAP user ID.	

Setting	Description	Example
Password	User's SAP password.	
Authentication Method = SNC without Single Sign On		
SNC User Name	User's SAP SNC user ID.	
SNC Password	User's SAP SNC password.	
Authentication Method = Single Sign On or **Authentication Method = SNC without Single Sign On**		

Partner Name	The SNC name of the communication partner (the target name).	`"p:CN=sap01.host1, OU=Admi-inistration, O=myCompany, C=US"`
		`p:CN=SAPService@myCompany.com`
		Note
		Enclose the SNC name in quotation marks if it contains any spaces.

Quality of Protection	The level of security protection applied by Secure Network Communications (SNC): ◦ **Authentication** - Verification of the identity of communication partners. No additional data protection. ◦ **Integrity** - Detection of any changes or manipulation of data in transit. (Includes Authentication) ◦ **Privacy** -Encryption of messages in transit. The maximum level of security protection provided by SNC. (Includes Integrity and Authentication) ◦ **Maximum Available** - When connecting to the SAP system, the highest level of security protection available is used.	
Mechanism	The authentication protocol used by the SAP system: ◦ **Kerberos** ◦ **NTLM** ◦ **Individual**	
Mechanism = Individual		
Library Path	Network path to the external security	`C:\SNC_Cryptolib\sapcrypto.dll`

Setting	Description	Example
	product's cryptographic library.	"C:\Security Product\library.dll" **Note** Enclose the network path in quotation marks if it contains any spaces.

Advanced settings

Setting	Description	Example
Maximum Number of Records Per Request optional	Specifies the maximum number of records in each portion of a query. You can specify a number from 5000 to 1,000,000. In most cases, you should leave the default of 15,000 records. When the connector queries the SAP system, it makes multiple requests and returns records in batches until all records are returned. It then assembles the batches into a complete table. On slower networks, you may be able to improve the performance of the SAP connector by increasing the number of records per request, which reduces the number of times the connector must reconnect.	100000
Variable String Length optional	The number of characters that are imported from the start of SAP fields with a data type of STRING or RAWSTRING. Large amounts of system information can	6 You know that only the first 6 characters in a STRING field contain information of interest.

Setting	Description	Example
	sometimes be present as trailing characters in fields of these types. You can use this setting to limit the number of characters that are imported. If you omit this setting, the default of 256 characters is used.	
Temporary Working Directory	The path for a directory that temporarily stores SAP data. When the SAP connector retrieves data it is temporarily stored locally. Large tables can cause an overflow error if they cannot be accommodated by the Analytics project folder, which is the default storage location. This setting allows you to specify a different temporary storage location, one with enough space for large files. If you leave the prefilled setting of **Default**, the data is temporary stored in the Analytics project folder.	`D:\SAP_temp_storage`

SAP Router
optional
	If the SAP system is using an SAProuter, the connection string for the router.
	The general form of the connection string is:
	/H/host name/S/port number/W/password (if used)
		`/H/saprouter.mycompany.com/S/3299`
		`/H/saprouter.mycompany.com/S/3299/W/password`
		Note
		`H`, `S`, and `W` must be uppercase.
		The specific form of your SAProuter connection string may differ.

RFC Function Module optional	The name of the RFC Function Module that forms the basis of the SAP connector.	

Setting	Description	Example
	Leave this field blank. The value is automatically provided in the background. The field exists in the event that the module is renamed in the future, and the new name must be manually specified.	

Searching for SAP tables

After you create a new connection to an SAP system, Analytics lists 200 common SAP tables in the **Available Tables** list in the Data Access window. You can scroll through the list and click any table name to add the table to the **Staging Area**.

Search the Available Tables list

Instead of scrolling through the list, you may find it easier to search for a table name or table description. You can enter a search string in the **Search tables** box above the list. The search string can contain both literal characters and wildcard characters.

The search extends beyond the list to also search any undisplayed names stored in the local cache of table names.

For detailed information about searching the **Available Tables** list, see "Add one or more tables to the Staging Area" on page 359.

What if the table I searched for is not found?

If the table you searched for is not in the **Available Tables** list, or in the table name cache, you can search and retrieve additional table names from the SAP system.

1. Enter a search string in the **Search tables** box.

   If the table is not found, the **Search table name in database** button appears.

2. Click **Search table name in database**.

   Any table name in the SAP system that matches the search string is added to the **Available Tables** list. The table name or names are also added to the table name cache so that they are more immediately available in the future.

> **Note**
> Searches in the SAP database are on the short SAP table name only (ANAT, BNKA, and so on). Table descriptions are not searched.

> **Tip**
> You cannot selectively remove table names from the **Available Tables** list or the table name cache. To clean up the list of tables, delete and recreate the SAP connection. The list reverts to containing only the common SAP tables.

Joining SAP tables

You can join SAP tables in the **Staging Area** in the Data Access window. Joining SAP tables has the following limitations:

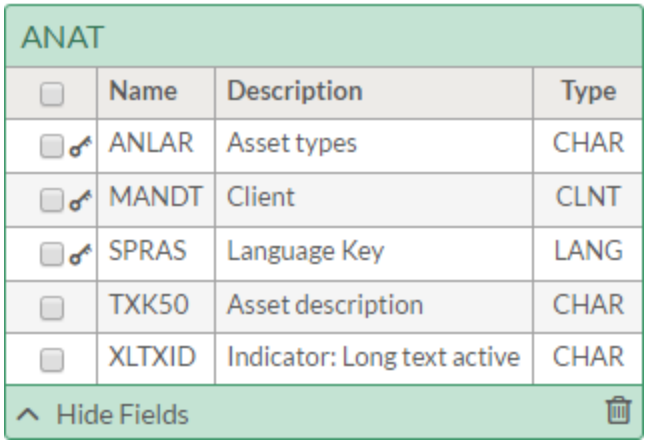
- Eight tables can be joined, unlike other Analytics connectors, which allow joining up to ten tables.
- Inner joins, and left joins, are supported. Outer joins, and right joins, are not supported.

For more information, see "Joining tables in the Data Access window" on page 368.

Key field icon

The key field icon identifies key fields in SAP tables added to the **Staging Area**. The purpose of the key field icon is to identify fields that you can use for joining SAP tables.

> **Note**
> Once you import SAP data into Analytics, the key field designation is lost, and key fields are not treated any differently from non-key fields.



SAP data type

The **Type** displayed with field names in SAP tables added to the **Staging Area** is the SAP data type. Other Analytics connectors display the Analytics data type. SAP data types are converted to Analytics data types upon import.

Language of the SAP system

In the **Data Connection Settings** panel, you can specify the language of the SAP system that you are connecting to.

Languages supported by Analytics

If the language of the SAP system is one of the languages supported by Analytics, the **Available Tables** list displays table descriptions in the supported language.

The languages supported by Analytics:

English	German	Spanish	French
Chinese	Japanese	Portuguese	

Languages not supported by Analytics

If the language of the SAP system is not one of the languages supported by Analytics, the **Available Tables** list displays the initial list of table descriptions in English. Any tables that you subsequently search and retrieve from the SAP system appear in the language of the system. The result is that table descriptions in the **Available Tables** list are a mix of English and the language of the SAP system.

> **Note**
> The short SAP table name (ANAT, BNKA, and so on) is standard across all languages.

SAP connection errors

Error number	Error code	Description
0	Unknown	Connecting to SAP, or importing

Error number	Error code	Description
		data, cannot be completed. An unknown error occurred.
1	SapDriverInitializationError	Cannot connect to SAP. Required driver configur-ation files may be missing.
2	SapDriverLicenseInvalidError	Cannot connect to SAP. There is a problem with the driver license.
3	ConnectionPropertiesParserValueMustNotBeNullOrWhitespaceError	Cannot connect to SAP. A required connection parameter is missing.
4	ConnectionPropertiesParserLibraryTypeOutOfRangeError	The value for the library parameter is not valid. Valid values are: ClassicRfc, NetweaverRfc
5	ConnectionPropertiesParserRfcFunctionTypeOutOfRangeError	The value for the RFC function parameter is not valid. Valid values are: Standard, Extended, Three
6	ConnectionPropertiesParserServerDetailInfoAmbiguousError	Cannot connect to SAP. The values specified for one or more of the following properties is incorrect: InstanceNumber, LogonGroup, SID.
7	ConnectionPropertiesParserVariableStringLengthNotAnIntegerError	The variable

Error number	Error code	Description
		string length for the connection parameter must be a whole number.
8	SapConnectionInfoValidationClientIsNullOrWhitespaceError	The value for the client connection parameter is missing.
9	SapConnectionInfoValidationClientLongerThanThreeCharactersError	The value for the client connection parameter cannot exceed 3 characters.
10	SapConnectionInfoValidationLanguageIsNullOrWhitespaceError	The value for the language connection parameter is missing.
11	SapConnectionInfoValidationLanguageIsLongerThanTwoCharactersError	The value for the language connection parameter cannot exceed 2 characters.
12	SapConnectionInfoValidationUsernameIsNullOrWhitespaceError	The value for the username connection parameter is missing.
13	SapConnectionInfoValidationPasswordIsNullOrWhitespaceError	The value for the password connection parameter is missing.
14	SapConnectionInfoValidationTestDownloadTableIsNullOrWhitespaceError	The value for the Test Download Table connection parameter is missing.
15	SapConnectionInfoValidationRfcFunctionModuleIsNullOrWhitespaceError	The value for the RFC Function Module

Error number	Error code	Description
		connection parameter is missing.
16	SapConnectionInfoValidationSapConnectionServerInfoIsNullError	Cannot connect to SAP. SAP server properties are missing (internal driver error).
17	SapConnectionInfoValidationVariableStringLengthMustBeGreaterZeroError	Cannot connect to SAP. Variable string length must be greater than 0 (internal driver error).
18	SapConnectionInfoValidationServerIsNullError	The value for the server connection parameter is missing.
19	SapConnectionInfoValidationServerDetailsIsNullError	Cannot connect to SAP. SAP server detail properties are missing (internal driver error).
20	SapConnectionInfoValidationInstanceNumberNullOrWhitespaceError	The value for the Instance Number connection parameter is missing.
21	SapConnectionInfoValidationInstanceNumberMustHaveTwoDigitsError	The value for the Instance Number connection parameter must be two digits.
22	SapConnectionInfoValidationLogonGroupNullOrWhitespaceError	The value for the Logon Group connection parameter is missing.
23	SapConnectionInfoValidationSidNullOrWhitespaceError	The value for the SID connection

Error number	Error code	Description
		parameter is missing.
24	SapConnectionInfoValidationSidLongerThanThreeCharactersError	The value for the SID connection parameter cannot exceed 3 characters.
25	SapDriverErpError	Connection to the SAP server was lost. An unknown error occurred.
26	SapDriverTableNotFoundError	The table cannot be found.
27	SapDriverTableStructureNotExportableError	The table cannot be imported because it has no data.
28	SapDriverAppendStructureNotExportableError	The table cannot be imported because it has no data.
29	SapDriverGetTablesDownloadError	The download of SAP tables could not be completed.
30	SapDriverSqlParsingError	The SQL syntax is probably not valid and the import of data cannot be performed. A specific cause of the error cannot be identified.
31	SapDriverSqlParsingNoSqlScriptFragmentError	SQL statements must be specified.
32	SapDriverSqlParsingExactlyOneBatchAllowedError	Only 1 SQL batch is allowed.
33	SapDriverSqlParsingExactlyOneSelectStatementAllowedError	Only 1 SELECT

Error number	Error code	Description
		statement is allowed.
34	SapDriverSqlParsingFragmentNotAllowedError	A portion of the SQL syntax is not allowed, or is not allowed in its current position. For example, UPDATE or DELETE statements are not allowed.
36	SapDriverSqlParsingJoinTypeNotAllowedError	The specified JOIN type is not allowed. Specify an INNER JOIN, or a LEFT JOIN.
37	SapDriverSqlParsingFirstJoinValueNotATableError	The first value specified by the JOIN is not a table.
38	SapDriverSqlParsingOnlyOneJoinAllowedError	Only 1 JOIN is allowed.
39	SapDriverSqlParsingSecondJoinValueNotATableError	The second value specified by the JOIN is not a table.
40	SapDriverSqlParsingNoJoinConditionError	The JOIN condition must not be empty.
41	SapDriverSqlParsingJoinConditionOringNotAllowedError	Using OR with JOIN conditions is not allowed.
42	SapDriverSqlParsingJoinConditionNotAllowedError	The type of JOIN condition specified is not allowed.
43	SapDriverSqlParsingJoinOperatorMustBeEqualsOnlyError	The JOIN operator can only be Equals.
44	SapDriverSqlParsingJoinSameFieldError	You cannot JOIN

Error number	Error code	Description
		a field to itself.
45	SapDriverSqlParsingOnlyExplicitJoinsAllowedError	Only explicit JOINS are allowed.
46	SapDriverSqlParsingHasNoSelectStatementError	A SELECT statement must be specified.
47	SapDriverSqlParsingHasNoQuerySpecificationError	SQL syntax is probably invalid. A query specification may be required.
48	SapDriverSqlParsingSqlFieldMustBeQualifiedWithTableError	The field must be qualified with a table name or a table alias.
49	SapDriver-SqlParsingSqlSelectStarFieldMustOnlyBeQualifiedWithTableNameError	SELECT * must be qualified with a table name or a table alias.
50	SapDriverSqlParsingSqlFieldMustOnlyBeQualifiedWithTableNameError	The field can only be qualified with a table name or a table alias.
51	SapDriverSqlParsingSqlTableMustHaveExactlyOneIdentifierError	The table name must not be qualified.
52	SapDriverSqlParsingSqlQueryeMustHaveAtLeastOneFieldError	The SELECT statement must contain at least one field.
53	SapDriverSqlParsingSqlQueryeMustHaveAtLeastOneTableError	The SELECT statement must contain at least one table.
54	SapDriver-SqlParsingSqlEnhancedQuerySqlTableNotMatchingAnySapTableError	The table qualifying a field does not match any table received from the

Error number	Error code	Description
		SAP system.
55	SapDriverSqlParsingSqlFieldQualifierNotMatchingAnyTableError	The table qualifying a field does not match any table specified in the SELECT statement.
56	SapDriverSqlParsingSqlSelectStarFieldQualifierNotMatchingAnyTableError	The table qualifying a * field does not match any table specified in the SELECT statement.
57	SapDriverSqlParsingSqlFieldNotInSapTableError	The field specified in the SELECT statement does not exist in the SAP data.
58	SapDriverSqlParsingSqlFieldNotExportableError	The field specified in the SELECT statement cannot be imported from the SAP system.
59	SapDriverSqlParsingSqlFieldNotFilterableError	The field specified in the WHERE clause cannot be filtered in the SAP system.
60	SapDriverSqlParsingTopRowFilterMustHaveAbsolutValueError	The SELECT TOP statement must specify an absolute value.
61	SapDriverSqlParsingTopRowFilterWithTiesNotAllowedError	The SELECT TOP statement must be used without ties.
63	SapDriverSqlParsingTopRowFilterMustHaveIntegerValueError	The SELECT TOP statement

Error number	Error code	Description
		must specify an integer value.
64	SapDriverSqlParsingTopRowFilterIntegerValueMustBeGreaterZeroError	The SELECT TOP statement must specify a value greater than zero (0).
68	SapDriver-SqlParsingTopRowFilterFromClauseMustHaveInnerQuerySpecificationError	The FROM clause in the SELECT TOP statement must contain a query specification.
69	SapDriverSqlParsingOringFilterValuesOnlyForSameFieldAndClauseError	Different fields cannot be specified in an OR condition. The same field must be used throughout the OR condition, and the same field operator must be used.
70	SapDriverSqlParsingFilterExpressionNotAllowedError	The expression specified in the WHERE clause for filtering the field is not allowed.
71	SapDriverSqlParsingFilterValueLiteralNotAllowedError	The literal specified in the WHERE clause for filtering the field is not allowed.
72	SapDriverSqlParsingFilterComparisonTypeNotAllowedError	The comparison operator specified in the WHERE clause for filtering the field is not allowed.
73	SapDriverPackageCreationFilterTypeNotSupportedError	The filter type is

Error number	Error code	Description
		not supported.
74	SapDriverTableCreationMaximumRecordLimitReachedError	The number of selected fields exceeds the maximum supported by the SAP RFC (SAP Remote Function Call).
75	SapDriverExecuteDownloadError	An unknown error occurred while importing SAP table data.
76	SapDriverSearchTablesDownloadError	An unknown error occurred while searching for SAP tables.
77	SapDriverSqlParsingNoFromClauseError	The SELECT statement must contain a FROM clause.
78	SapDriverSqlParsingNoTableReferenceInFromClauseError	The FROM clause in the SELECT statement must specify a table.
79	SapDriverSqlParsingExpectedTopRowForPreviewError	The outer SELECT statement must have a TOP clause.
80	SapDriverSqlParsingPreviewStatmentMustNotHaveWhereClauseError	The outer SELECT statement must not have a WHERE clause.
81	SapDriver-SqlParsingPreviewStatmentMustHaveUnqualifiedStarQualifierOnlyError	The outer SELECT statement must only have a * qualifier.
82	SapDriverProjectPathNotFoundError	The specified

Error number	Error code	Description
		Temporary Working Directory does not exist.

Connecting to SAP ByDesign

SAP Business ByDesign is a cloud enterprise resource planning (ERP) software that enables customers to match supply-demand better and manage inventory. You can use the SAP ByDesign data connector to import your organization's SAP ByDesign data.

> **Note**
> The SAP ByDesign data connector is provided by our data partner, CData. For information on any of the connection fields, refer the documentation available at the CData website.

Before you start

To connect to SAP ByDesign, you must gather the following:

- the URL to your system host name
- credentials to connect to the SAP ByDesign account

For help gathering the connection prerequisites, contact the SAP ByDesign administrator in your organization. If your administrator cannot help you, you or your administrator should contact SAP ByDesign Support.

Create a SAP ByDesign connection

1. From the Analytics main menu, select **Import > Database and application**.
2. From the **New Connections** tab, in the **ACL DSN Connectors (Bundled)** section, select **SAP ByDesign.**

   > **Tip**
   > You can filter the list of available connectors by entering a search string in the **Filter connections** box. Connectors are listed alphabetically.

   The DSN Configuration dialog box appears.
3. In the **DSN Configuration** dialog box, click **Show Required**.
4. Provide values for the required fields, if any.

5. Click **Test Connection**.

> **Note**
> When you click the **Test Connection** button, it validates whether the connection and authentication details provided are correct. If you click **OK** without clicking Test Connection, the connection details are saved without any validation and may not work later if the values provided are incorrect.

   The log in page for SAP ByDesign appears.

6. Provide the connection details and authenticate your login.

   When the connection is successful, a dialog box opens displaying the success message.

7. Click **OK** in the connection successful dialog box that appears.

   The connection to SAP ByDesign is established successfully.

8. In the **DSN Configuration** dialog box, click **OK**.

The connection for SAP ByDesign is saved to the **Existing Connections** tab. In the future, you can reconnect to SAP ByDesign from the saved connection.

Once the connection is established, the Data Access window opens to the **Staging Area** and you can begin importing data. For help importing data from SAP ByDesign, see "Working with the Data Access window" on page 357.

Rename the connection

When you create the SAP ByDesign data connector, you cannot provide a Connection name of your choice and the connection is saved to the **Existing Connections** tab with the same name as your data source. If you want to rename it, navigate to the **Existing Connections** tab, expand the **ACL Connectors (DSN)** section, click the ellipsis icon corresponding to SAP ByDesign and select **Rename connection**.

Connecting to SAP Hybris Cloud for Customer

SAP Hybris Cloud for Customer (C4C) is an on-demand cloud-based Customer Relationship Management solution to manage customer sales, customer service, and marketing activities. You can use the SAP Hybris C4C data connector to import your organization's SAP Hybris C4C data.

> **Note**
> The SAP Hybris C4C data connector is provided by our data partner, CData. For information on any of the connection fields, refer the documentation available at the CData website.

Before you start

To connect to SAP Hybris C4C, you must gather the following:

- the credentials to connect to the SAP Hybris C4C account
- the tenant to establish the connection to
- the SAP Hybris C4C URL to connect to

For help gathering the connection prerequisites, contact the SAP Hybris C4C administrator in your organization. If your administrator cannot help you, you or your administrator should contact SAP Hybris C4C Support.

Create a SAP Hybris C4C connection

1. From the Analytics main menu, select **Import > Database and application**.
2. From the **New Connections** tab, in the **ACL DSN Connectors (Bundled)** section, select **SAP Hybris C4C.**

   > **Tip**
   > You can filter the list of available connectors by entering a search string in the **Filter connections** box. Connectors are listed alphabetically.

   The DSN Configuration dialog box appears.
3. In the **DSN Configuration** dialog box, click **Show Required**.
4. Provide values for the required fields, if any.

5. Click **Test Connection**.

> **Note**
> When you click the **Test Connection** button, it validates whether the connection and authentication details provided are correct. If you click **OK** without clicking Test Connection, the connection details are saved without any validation and may not work later if the values provided are incorrect.

   The log in page for SAP Hybris C4C appears.

6. Provide the connection details and authenticate your login.

   When the connection is successful, a dialog box opens displaying the success message.

7. Click **OK** in the connection successful dialog box that appears.

   The connection to SAP Hybris C4C is established successfully.

8. In the **DSN Configuration** dialog box, click **OK**.

The connection for SAP Hybris C4C is saved to the **Existing Connections** tab. In the future, you can reconnect to SAP Hybris C4C from the saved connection.

Once the connection is established, the Data Access window opens to the **Staging Area** and you can begin importing data. For help importing data from SAP Hybris C4C, see "Working with the Data Access window" on page 357.

Rename the connection

When you create the SAP Hybris C4C data connector, you cannot provide a Connection name of your choice and the connection is saved to the **Existing Connections** tab with the same name as your data source. If you want to rename it, navigate to the **Existing Connections** tab, expand the **ACL Connectors (DSN)** section, click the ellipsis icon corresponding to SAP Hybris C4C and select **Rename connection**.

Connecting to SAP SuccessFactors

SAP SuccessFactors is a cloud-based human resources (HR) solution to manage HR functions, such as business alignment, people performance, and recruitment. You can use the SAP SuccessFactors data connector to import your organization's SAP SuccessFactors data.

> **Note**
> The SAP SuccessFactors data connector is provided by our data partner, CData. For information on any of the connection fields, refer the documentation available at the CData website.

Before you start

To connect to SAP SuccessFactors, you must gather the following:

- the URL to the SuccessFactors server
- the credentials to connect to the SAP SuccessFactors account
- the unique identifier of your company

For help gathering the connection prerequisites, contact the SAP SuccessFactors administrator in your organization. If your administrator cannot help you, you or your administrator should contact SAP SuccessFactors Support.

Create a SAP SuccessFactors connection

1. From the Analytics main menu, select **Import > Database and application**.
2. From the **New Connections** tab, in the **ACL DSN Connectors (Bundled)** section, select **SAP SuccessFactors.**

   > **Tip**
   > You can filter the list of available connectors by entering a search string in the **Filter connections** box. Connectors are listed alphabetically.

   The DSN Configuration dialog box appears.
3. In the **DSN Configuration** dialog box, click **Show Required**.
4. Provide values for the required fields, if any.

5. Click **Test Connection**.

> **Note**
> When you click the **Test Connection** button, it validates whether the connection and authentication details provided are correct. If you click **OK** without clicking Test Connection, the connection details are saved without any validation and may not work later if the values provided are incorrect.

   The log in page for SAP SuccessFactors appears.

6. Provide the connection details and authenticate your login.

   When the connection is successful, a dialog box opens displaying the success message.

7. Click **OK** in the connection successful dialog box that appears.

   The connection to SAP SuccessFactors is established successfully.

8. In the **DSN Configuration** dialog box, click **OK**.

The connection for SAP SuccessFactors is saved to the **Existing Connections** tab. In the future, you can reconnect to SAP SuccessFactors from the saved connection.

Once the connection is established, the Data Access window opens to the **Staging Area** and you can begin importing data. For help importing data from SAP SuccessFactors, see "Working with the Data Access window" on page 357.

Rename the connection

When you create the SAP SuccessFactors data connector, you cannot provide a Connection name of your choice and the connection is saved to the **Existing Connections** tab with the same name as your data source. If you want to rename it, navigate to the **Existing Connections** tab, expand the **ACL Connectors (DSN)** section, click the ellipsis icon corresponding to SAP SuccessFactors and select **Rename connection**.

Connecting to ServiceNow

ServiceNow is a cloud provider of IT services management, IT operations management, and IT business management solutions. Use the ServiceNow data connector to import your company's ServiceNow data.

> **Note**
> You cannot use this connector independently of Analytics. You can configure a DSN for the connector driver using the Windows **ODBC Data Source Administrator**, however you must test the DSN connection from within Analytics and not from the connector's Windows DSN configuration dialog.

Before you start

To connect to ServiceNow, you must gather the following:

- the connecting user account, including the user name and password
- the instance to retrieve tables from
- the Client Id and Client Secret OAuth credentials assigned when you register an application with ServieNow's OAuth authorization server

> **Tip**
> For more information about registering an application and obtaining OAuth credentials, search "Set up OAuth" in the ServiceNow documentation.

For help gathering the connection prerequisites, contact the ServiceNow administrator in your organization. If your administrator cannot help you, you or your administrator should contact ServiceNow Support.

Create a ServiceNow connection

1. From the Analytics main menu, select **Import > Database and application**.
2. From the **New Connections** tab, in the **ACL Connectors** section, select **ServiceNow.**

> **Tip**
> You can filter the list of available connectors by entering a search string in the **Filter connections** box. Connectors are listed alphabetically.

3. In the **Data Connection Settings** panel, enter the connection settings and at the bottom of the panel, click **Save and Connect**.

   You can accept the default **Connection Name**, or enter a new one.

The connection for ServiceNow is saved to the **Existing Connections** tab. In the future, you can reconnect to ServiceNow from the saved connection.

Once the connection is established, the Data Access window opens to the **Staging Area** and you can begin importing data. For help importing data from ServiceNow, see "Working with the Data Access window" on page 357.

Connection settings

Basic settings

Setting	Description	Example
User	The user account used to authenticate to ServiceNow.	admin_1
Password	The password used to authenticate the user.	
Instance	The ServiceNow instance to retrieve tables from.	staging16387
OAuth Client Id	The client Id assigned when you register your application with ServiceNow's OAuth authorization server. OAuth requires you to register your application. As part of the registration, you will receive a client Id, sometimes also called a consumer key, and a client secret. You must specify both to connect.	356a825803610300ef0662490d237522
OAuth Client Secret	The client secret assigned when you register your application with ServiceNow's OAuth authorization server. OAuth requires you to register your application.	secretTest

Setting	Description	Example
	As part of the registration you will receive a client Id and a client secret, sometimes also called a consumer secret. You must specify both to connect.	

Advanced settings

Setting	Description	Example
Convert Datetime to GMT	Converts datetime fields to GMT time zone during import. If false, the datetime value is converted to the operating system time zone of the machine running Analytics. **Note** If you select this option, any filters you apply against datetime fields using the **ON** (=) operator require that you use the 'Z' format specifier when entering the datetime value for the filter condition: 2017-01-01 12:38:54Z.	true
Limit Key Size	The maximum length of a primary key column. Setting the size to 0 will make the key length revert to the original length. This property makes the connector override the reported length of all the primary key columns.	255
Map to Long Varchar	Controls whether or not a column is returned as SQL_LONGVARCHAR. Use this setting to map any column larger than the specified size so they are reported as SQL_LONGVARCHAR instead of SQL_VARCHAR.	-1
Map To WVarchar	Controls whether or not string types map to SQL_WVARCHAR instead	true

Setting	Description	Example
	of SQL_VARCHAR. It is set by default. String columns must be mapped to SQL_WVARCHAR to accommodate various international character sets, so MapToWVarchar is set to true by default. You may set it to false to use SQL_VARCHAR instead.	
Pseudo Columns	Indicates whether or not to include pseudo columns as columns to the table. This setting is particularly helpful in Entity Framework, which does not allow you to set a value for a pseudo column unless it is a table column. The value of this connection setting is of the format "Table1=Column1, Table1=Column2, Table2=Column3". You can use the "*" character to include all tables and all columns.	MyTable=*
SSL Server Cert	The certificate to be accepted from the server when connecting using TLS/SSL. You can specify any of the following: ◦ full PEM certificate ◦ path to a local file containing the certificate ◦ the public key ◦ the MD5 Thumbprint (hex values can also be either space or colon separated) ◦ the SHA1 Thumbprint (hex values can also be either space or colon separated) If using a TLS/SSL connection, this property can be used to specify the TLS/SSL certificate to be accepted from the server. Any other certificate that is not trusted by the machine will be rejected.	C:\cert.cer
Proxy Auth Scheme	The authentication type to use to authenticate to the ProxyServer proxy. This value specifies the authentic-	BASIC

Setting	Description	Example
	ation type to use to authenticate to the HTTP proxy specified by ProxyServer and ProxyPort. **Note** The connector will use the system proxy settings by default, without further configuration needed; if you want to connect to another proxy, you will need to set ProxyAutoDetect to false, in addition to ProxyServer and ProxyPort. To authenticate, set ProxyAuthScheme and set ProxyUser and ProxyPassword, if needed. The authentication type can be one of the following: ◦ **BASIC** - The driver performs HTTP BASIC authentication ◦ **DIGEST** - The driver performs HTTP DIGEST authentication. ◦ **NEGOTIATE** - The driver retrieves an NTLM or Kerberos token based on the applicable protocol for authentication ◦ **PROPRIETARY** - The driver does not generate an NTLM or Kerberos token. You must supply this token in the Authorization header of the HTTP request	
Proxy Auto Detect	Indicates whether to use the system proxy settings or not. Set ProxyAutoDetect to FALSE to use custom proxy settings. This takes precedence over other proxy settings.	true
Proxy User	A user name to be used to authenticate to the ProxyServer proxy. The ProxyUser and ProxyPassword options are used to connect and	john_doe@example.com

Setting	Description	Example
	authenticate against the HTTP proxy specified in ProxyServer. You can select one of the available authentication types in ProxyAuthScheme. If you are using HTTP authentication, set this to the username of a user recognized by the HTTP proxy. If you are using Windows or Kerberos authentication, set this property to a username in one of the following formats: ◦ user@domain ◦ domain\user	
Proxy Password	A password to be used to authenticate to the ProxyServer proxy. This property is used to authenticate to an HTTP proxy server that supports NTLM (Windows), Kerberos, or HTTP authentication. To specify the HTTP proxy, you can set ProxyServer and ProxyPort. To specify the authentication type, set ProxyAuthScheme. If you are using HTTP authentication, additionally set ProxyUser and ProxyPassword to HTTP proxy. If you are using NTLM authentication, set ProxyUser and ProxyPassword to your Windows password. You may also need these to complete Kerberos authentication	
Proxy Server	The hostname or IP address of a proxy to route HTTP traffic through. The hostname or IP address of a proxy to route HTTP traffic through. The driver can use the HTTP, Windows (NTLM), or Kerberos authentication types to authenticate to an HTTP proxy. By default, the driver uses the system proxy. If you need to use another proxy, set ProxyAutoDetect to false.	206.174.193.115
Proxy Port	The TCP port the ProxyServer	80

Setting	Description	Example
	proxy is running on.	
Proxy SSL Type	The SSL type to use when connecting to the ProxyServer proxy: ◦ **AUTO** - If the URL is an HTTPS URL, the connector will use the TUNNEL option. If the URL is an HTTP URL, the connector will use the NEVER option (default) ◦ **ALWAYS** - the connection is always SSL enabled ◦ **NEVER** - the connection is not SSL enabled ◦ **TUNNEL** - the connection is through a tunneling proxy: The proxy server opens a connection to the remote host and traffic flows back and forth through the proxy	AUTO

Data connector updates

When you upgrade Analytics, the Robots Agent, or AX Server, you should test any of your scripts that import data using one of the Analytics data connectors (ACCESSDATA command).

The possibility exists that changes made by third-party data sources or ODBC driver vendors required updates to one or more of the data connectors. Scripted data connections may need to be updated in order to continue working correctly.

- **Re-run the import** – The easiest way to update a connection is to manually perform an import using the Data Access window in the upgraded version of Analytics. Copy the ACCESSDATA command from the log and use it to update your script.

  > **Note**
  > Before connecting to a data source and re-running the import, clear the connector cache to flush the existing set of table names.
  > In the **Existing Connections** tab in the Data Access window, beside the connector name, select ··· **> Clear cache**.

- **Update field specifications** – You may also need to update field specifications in the script body to align with table schema changes in the data source or ODBC driver. Possible changes include field names, field data types, and field and record lengths.
- **Check the results of any filtering** – You should also check the results of any filtering that you apply as part of the data import. Confirm that the import filtering is including and excluding records correctly.

ServiceNow data connector changes

Specific changes made to the ServiceNow data connector are listed below.

Analytics version	Change
14.2	ServiceNow date fields are now mapped to a date data type in the ServiceNow data connector, with a format of YYYY-MM-DD and a length of 10 characters. Previously, ServiceNow date fields were mapped to a datetime data type in the ServiceNow data connector, with a format of YYYY-MM-DD hh:mm:ss and a length of 19 characters. The time portion of the imported data was empty, and represented as 00:00:00.

Connecting to SFTP

SFTP is a secure file transfer protocol that runs over the SSH protocol. Organizations use SFTP to send files and data with an added layer of security. You can use the SFTP data connector to import your company's SFTP data.

> **Note**
> The SFTP data connector can be used to import file types with extensions .xlsx and csv only. Script support for this connector may also be limited.

Before you start

To connect to SFTP, you must gather the following:

- the credentials to connect to the SFTP server
- the host name or IP address of the SFTP server
- the correct connection port
- the path in the SFTP server

For help gathering the connection prerequisites, contact the SFTP administrator in your organization. If your administrator cannot help you, you or your administrator should contact SFTP Support.

Create an SFTP connection

1. From the Analytics main menu, select **Import > Database and application**.
2. From the **New Connections** tab, in the **ACL Connectors** section, select **SFTP.**

   > **Tip**
   > You can filter the list of available connectors by entering a search string in the **Filter connections** box. Connectors are listed alphabetically.

3. In the **Data Connection Settings** panel, enter the connection settings and at the bottom of the panel, click **Save and Connect**.

   You can accept the default **Connection Name**, or enter a new one.

The connection for SFTP is saved to the **Existing Connections** tab. In the future, you can reconnect to SFTP from the saved connection.

Once the connection is established, the Data Access window opens to the **Staging Area** and you can begin importing data. For help importing data from SFTP, see "Working with the Data Access window" on page 357.

Connection settings

Basic settings

Setting	Description	Example
User	The user account to authenticate against the SFTP server.	
Password	The password used to authenticate the user with the SFTP server.	
Host	The host name or IP address of the instance running the SFTP server.	https://sftp.abc.com/
Port	The port for the SFTP server.	443
Path	The current working directory on the SFTP server. You can change the working directory by setting it to an absolute directory path or a relative path with respect to the existing value of this property.	

Advanced settings

Setting	Description	Example
Limit Key Size	The maximum length of a primary key column.	255
Map To Long Varchar	This property controls whether a	-1

Setting	Description	Example
	column is returned as SQL_LONGVARCHAR or not.	
Map To WVarchar	This property controls whether string types map to SQL_WVARCHAR instead of SQL_VARCHAR.	
Pseudo Columns	This property indicates whether to include pseudo columns as columns to the table. The value of this connection setting is of the format "Table1=Column1, Table1=Column2, Table2=Column3".	
SSL Server Certificate	The certificate that must be accepted from the server when connecting using SSL. You can provide any of the following: ◦ a full PEM certificate ◦ path to a local file containing the certificate ◦ public key ◦ MD5 or SHA1 thumbprint (hex values can also be either space or colon separated) Any other certificate that is not trusted by the machine is rejected.	

Connecting to SharePoint

Microsoft SharePoint is a web-based collaborative platform for sharing and managing organizational content and applications. Use the SharePoint data connector to import your company's SharePoint data.

Before you import

Only tabular data

Analytics can only query SharePoint data that is in tabular form. The following items cannot be queried from Analytics because they are not in tabular form.

- Attachments
- FileVersions
- GetValidTerms
- Permissions
- Views

Multi-factor authentication not supported

The SharePoint data connector cannot access data from a SharePoint account that requires multi-factor authentication.

Known issue: filtering on datetime fields

If you import your data with the **Convert Datetime to GMT** option checked, you cannot filter on datetime fields. If you need to filter based on a datetime field, clear this option before importing.

Create a SharePoint connection

1. From the Analytics main menu, select **Import > Database and application**.
2. From the **New Connections** tab, in the **ACL Connectors** section, select **Sharepoint.**

> **Tip**
> You can filter the list of available connectors by entering a search string in the **Filter connections** box. Connectors are listed alphabetically.

3. In the **Data Connection Settings** panel, enter the connection settings and at the bottom of the panel, click **Save and Connect**.

   You can accept the default **Connection Name**, or enter a new one.

The connection for Sharepoint is saved to the **Existing Connections** tab. In the future, you can reconnect to Sharepoint from the saved connection.

Once the connection is established, the Data Access window opens to the **Staging Area** and you can begin importing data. For help importing data from Sharepoint, see "Working with the Data Access window" on page 357.

Connection settings

Basic settings

Setting	Description	Example
Connection Name	The name you want to give this connection in Analytics.	Sharepoint
Host	The SharePoint site you are connecting to. Set the URL to a Site Collection to work with all Lists and Documents in all nested Subsites. Set the URL to a specific Site to work with Lists and Documents in that Site only.	https://teams.example.com/teamA
User	Your SharePoint user name. > Note > If your company accesses SharePoint through SSO tools like AD FS, OneLogin, or OKTA, you can enter your SSO credentials to connect through Analytics rather than your SharePoint credentials.	jgibbons

Setting	Description	Example
Password	Your SharePoint password. **Note** If your company accesses SharePoint through SSO tools like AD FS, OneLogin, or OKTA, you can enter your SSO credentials to connect through Analytics rather than your SharePoint credentials.	MyStrongPasswordExample
Share Point Edition	The edition of SharePoint being used. It is either SharePoint Online or SharePoint On-Premise.	Sharepoint Online
Use SSO	When set to true, single sign-on (SSO) will be used to authenticate to SharePoint Online using the account specified via User and Password. Active Directory Federation Services (AD FS), OneLogin, and OKTA SSO are supported. SSO Domain may be required if the domain configured on the SSO domain is different than the domain for User. SSO is only applicable when using SharePoint Online. SSO is not supported for On-Premise versions of SharePoint.	false
Auth Scheme	Together with Password and User, this field is used to authenticate against the server. NTLM is the default option. Use the following options to select your authentication scheme: ◦ **NTLM** - Set this to use your Windows credentials for authentication. ◦ **NEGOTIATE** - If Auth Scheme is set to NEGOTIATE, the driver will negotiate an authentication mechanism with the server. Set Auth Scheme to NEGOTIATE if you want to use Kerberos authentication. ◦ **KERBEROSDELEGATION** - Set this to use delegation through the Kerberos protocol. Set the User and Password of the account you want to impersonate. ◦ **NONE** - Set this to use anonymous authentication; for example, to access a public site. ◦ **FORMS** - Set this if your SharePoint instance uses a custom authentication method through a Web form. ◦ **DIGEST** - Set this to use HTTP Digest authentication. ◦ **BASIC** - Set this to use HTTP Basic authentication.	NTLM

Advanced settings

Setting	Description	Example
SSO Domain	This property is only applicable when using single sign-on (Use SSO is set to true) and if the domain for User (e.g. user@mydo-main.com) is different than the domain configured within the SSO service (e.g. user@myssodomain.com). This property may be required when using AD FS, OneLogin, or OKTA SSO.	myssodomain.com
Convert Datetime to GMT	Converts datetime fields to GMT time zone during import. If false, the datetime value is converted to the operating system time zone of the machine running Analytics.	true
Limit Key Size	In some ODBC tools, (Microsoft Access, for example) the length of the primary key column cannot be larger than a specific value. This property makes the ODBC Driver override the reported length of all the primary key columns. It is especially useful when using the ODBC Driver as a Microsoft Access Linked Data Source. Setting Limit Key Size to zero will make the key length revert to the original length.	255
Map to Long Varchar	This property controls whether or not a column is returned as SQL_ LONGVARCHAR. Some applications require all text data larger than a certain number of characters to be reported as SQL_LONGVARCHAR. Use this to map any column larger than the specified size so they are reported as SQL_ LONGVARCHAR instead of SQL_ VARCHAR.	-1
Map to WVarchar	This property controls whether or not string types map to SQL_WVARCHAR instead of SQL_VARCHAR. It is set by default. String columns must be mapped to SQL_ WVARCHAR to accommodate various international character sets, so Map to WVarchar is set to true by default. You may set it to false to use SQL_VARCHAR instead.	true
Pseudo Columns	Indicates whether or not to include pseudo	◦ Table1=Column1, Table2=Column3

Setting	Description	Example
	columns as columns to the table. This setting is particularly helpful in Entity Framework, which does not allow you to set a value for a pseudo column unless it is a table column. You can use an asterisk (*) to include all tables and all columns.	◦ *=*
Upper Case Identifiers	Set this property to report all identifiers, including table and column names, in uppercase. This is the default for Oracle databases and thus allows better integration with Oracle tools such as the Oracle Database Gateway. For example, you can use this property to avoid quoting identifiers.	false
Proxy Authentication Scheme	The authentication type to use to authenticate to the Proxy Server. This value specifies the authentication type to use to authenticate to the HTTP proxy specified by Proxy Server and Proxy Port. Note that the driver will use the system proxy settings by default, without further configuration needed. If you want to connect to another proxy, you will need to set Proxy Auto Detect to false, in addition to Proxy Server and Proxy Port. To authenticate, set Proxy Authentication Scheme and set Proxy User and Proxy Password, if needed. The authentication type can be one of the following: ◦ **BASIC** - The driver performs HTTP BASIC authentication. ◦ **DIGEST** - The driver performs HTTP DIGEST authentication. ◦ **NEGOTIATE** - The driver retrieves an NTLM or Kerberos token based on the applicable protocol for authentication. ◦ **PROPRIETARY** - The driver does not generate an NTLM or Kerberos token. You must supply this token in the Authorization header of the HTTP request.	BASIC
Proxy Auto Detect	This indicates whether to use the system proxy settings or not. Set ProxyAutoDetect to false to use custom proxy settings. This takes precedence over other proxy settings.	false

Setting	Description	Example
	By default, the driver uses the system HTTP proxy. Set this to false if you want to connect to another proxy. To connect to an HTTP proxy, see Proxy Server.	
Proxy User	A user name to be used to authenticate to the Proxy Server. The Proxy User and Proxy Password options are used to connect and authenticate against the HTTP proxy specified in ProxyServer. You can select one of the available authentication types in Proxy Authentication Scheme. If you are using HTTP authentication, set this to the username of a user recognized by the HTTP proxy. If you are using Windows or Kerberos authentication, set this to a username in one of the following formats: ◦ user@domain ◦ domain\user	jgibbons@example.com
Proxy Password	A password to be used to authenticate to the Proxy Server. This property is used to authenticate to an HTTP proxy server that supports NTLM (Windows), Kerberos, or HTTP authentication. To specify the HTTP proxy, you can set Proxy Server and Proxy Port. To specify the authentication type, set Proxy Authentication Scheme. If you are using HTTP authentication, additionally set Proxy User and Proxy Password to HTTP proxy. If you are using NTLM authentication, set Proxy User and Proxy Password to your Windows password. You may also need these to complete Kerberos authentication. By default, the driver uses the system proxy. If you want to connect to another proxy, set Proxy Auto Detect to false.	MyStrongPasswordExample
Proxy Server	The hostname or IP address of a proxy to route HTTP traffic through. The driver can use the HTTP, Windows (NTLM), or Kerberos authentication types to	127.168.192.10

Setting	Description	Example
	authenticate to an HTTP proxy. By default, the driver uses the system proxy. If you need to use another proxy, set Proxy Auto Detect to false.	
Proxy Port	The port the HTTP proxy is running on that you want to redirect HTTP traffic through. Specify the HTTP proxy in Proxy Server.	80
Proxy SSL Type	This property determines when to use SSL for the connection to an HTTP proxy specified by ProxyServer. This value can be AUTO, ALWAYS, NEVER, or TUNNEL. **Auto** -Default setting. If the URL is an HTTPS URL, the driver will use the TUNNEL option. If the URL is an HTTP URL, the component will use the NEVER option. **Always** - The connection is always SSL enabled. **Never** - The connection is not SSL enabled. **Tunnel** - The connection is through a tunneling proxy. The proxy server opens a connection to the remote host and traffic flows back and forth through the proxy.	Auto
Proxy Exception	A semicolon separated list of hosts or IPs that will be exempt from connecting through the ProxyServer. The ProxyServer will be used for all addresses, except for addresses defined in this property.	127.168.189.10; 127.168.188.11

Connecting to Slack

Slack is a messaging app for teams and workplaces that can be used across multiple devices and platforms. You can use the Slack data connector to import your organization's Slack data .

> **Note**
> The Slack data connector is provided by our data partner, CData. For information on any of the connection fields, refer the documentation available at the CData website.

Before you start

To connect to Slack, you must gather the credentials to connect to the Slack server.

For help gathering the connection prerequisites, contact the Slack administrator in your organization. If your administrator cannot help you, you or your administrator should contact Slack Support.

Create a Slack connection

1. From the Analytics main menu, select **Import > Database and application**.
2. From the **New Connections** tab, in the **ACL DSN Connectors (Bundled)** section, select **Slack.**

   > **Tip**
   > You can filter the list of available connectors by entering a search string in the **Filter connections** box. Connectors are listed alphabetically.

   The DSN Configuration dialog box appears.
3. In the **DSN Configuration** dialog box, click **Test Connection**.

   The Slack Log in page appears.
4. To log in:
   - If you have the Slack account credentials, provide the credentials.
   - If your organization uses SSO, click the button to sign in with SSO.
5. Click **Continue**.

   If you are using SSO, provide the credentials and log in.
6. Click **OK** in the connection successful dialog box that appears.

   The connection to Slack is established successfully.
7. In the **DSN Configuration** dialog box, click **OK**.

The connection for Slack is saved to the **Existing Connections** tab with the connection name as **Slack** . In future, you can reconnect to Slack using the saved connection.

Once the connection is established, the Data Access window opens to the **Staging Area** and you can begin importing data. For help importing data from Slack, see "Working with the Data Access window" on page 357.

Rename the connection

When you create the Slack data connector, you cannot provide a Connection name of your choice and the connection is saved to the **Existing Connections** tab with the same name as your data source. If you want to rename it, navigate to the **Existing Connections** tab, expand the **ACL Connectors (DSN)** section, click the ellipsis icon corresponding to Slack and select **Rename connection**.

Connecting to Snowflake

Snowflake is a cloud-based data warehouse platform that delivers high performance, scalability, elasticity, and concurrency and features global services layers that are physically separated but logically integrated. You can use the Snowflake data connector to import your organization's Snowflake data.

> **Note**
> The Snowflake data connector is provided by our data partner, CData. For information on any of the connection fields, refer the documentation available at the CData website.

Before you start

To connect to Snowflake, you must gather the following:

- credentials to connect to the Snowflake database
- name of the Snowflake warehouse
- the URL of Snowflake database

For help gathering the connection prerequisites, contact the Snowflake administrator in your organization. If your administrator cannot help you, you or your administrator should contact Snowflake Support.

Create a Snowflake connection

1. From the Analytics main menu, select **Import > Database and application**.
2. From the **New Connections** tab, in the **ACL DSN Connectors (Bundled)** section, select **Snowflake.**

   > **Tip**
   > You can filter the list of available connectors by entering a search string in the **Filter connections** box. Connectors are listed alphabetically.

   The DSN Configuration dialog box appears.
3. In the **DSN Configuration** dialog box, click **Show Required**.
4. Provide values for the required fields, if any.

5. Click **Test Connection**.

> **Note**
> When you click the **Test Connection** button, it validates whether the connection and authentication details provided are correct. If you click **OK** without clicking Test Connection, the connection details are saved without any validation and may not work later if the values provided are incorrect.

   The log in page for Snowflake appears.

6. Provide the connection details and authenticate your login.

   When the connection is successful, a dialog box opens displaying the success message.

7. Click **OK** in the connection successful dialog box that appears.

   The connection to Snowflake is established successfully.

8. In the **DSN Configuration** dialog box, click **OK**.

The connection for Snowflake is saved to the **Existing Connections** tab. In the future, you can reconnect to Snowflake from the saved connection.

Once the connection is established, the Data Access window opens to the **Staging Area** and you can begin importing data. For help importing data from Snowflake, see "Working with the Data Access window" on page 357.

Rename the connection

When you create the Snowflake data connector, you cannot provide a Connection name of your choice and the connection is saved to the **Existing Connections** tab with the same name as your data source. If you want to rename it, navigate to the **Existing Connections** tab, expand the **ACL Connectors (DSN)** section, click the ellipsis icon corresponding to Snowflake and select **Rename connection**.

Connecting to Splunk

Splunk is a security information and event management (SIEM) solution that enables organizations to detect and respond to security attacks, to simplify threat management, and safeguard your business. You can use the Splunk data connector to import your organization's Splunk data.

> **Note**
> The Splunk data connector is provided by our data partner, CData. For information on any of the connection fields, refer the documentation available at the CData website.

Before you start

To connect to Splunk, you must gather the following:

- credentials to connect to the Splunk account
- URL to access the Splunk account

For help gathering the connection prerequisites, contact the Splunk administrator in your organization. If your administrator cannot help you, you or your administrator should contact Splunk Support.

Create a Splunk connection

1. From the Analytics main menu, select **Import > Database and application**.
2. From the **New Connections** tab, in the **ACL DSN Connectors (Bundled)** section, select **Splunk.**

   > **Tip**
   > You can filter the list of available connectors by entering a search string in the **Filter connections** box. Connectors are listed alphabetically.

   The DSN Configuration dialog box appears.
3. In the **DSN Configuration** dialog box, click **Show Required**.
4. Provide values for the required fields, if any.

5. Click **Test Connection**.

> **Note**
> When you click the **Test Connection** button, it validates whether the connection and authentication details provided are correct. If you click **OK** without clicking Test Connection, the connection details are saved without any validation and may not work later if the values provided are incorrect.

   The log in page for Splunk appears.

6. Provide the connection details and authenticate your login.

   When the connection is successful, a dialog box opens displaying the success message.

7. Click **OK** in the connection successful dialog box that appears.

   The connection to Splunk is established successfully.

8. In the **DSN Configuration** dialog box, click **OK**.

The connection for Splunk is saved to the **Existing Connections** tab. In the future, you can reconnect to Splunk from the saved connection.

Once the connection is established, the Data Access window opens to the **Staging Area** and you can begin importing data. For help importing data from Splunk, see "Working with the Data Access window" on page 357.

Rename the connection

When you create the Splunk data connector, you cannot provide a Connection name of your choice and the connection is saved to the **Existing Connections** tab with the same name as your data source. If you want to rename it, navigate to the **Existing Connections** tab, expand the **ACL Connectors (DSN)** section, click the ellipsis icon corresponding to Splunk and select **Rename connection**.

Connecting to Square

Square is a financial and merchant services aggregator that enables to run businesses. You can use the Square data connector to import your organization's Square data.

> **Note**
> The Square data connector is provided by our data partner, CData. For information on any of the connection fields, refer the documentation available at the CData website.

Before you start

To connect to Square, you must gather the credentials to connect to the Square server.

For help gathering the connection prerequisites, contact the Square administrator in your organization. If your administrator cannot help you, you or your administrator should contact Square Support.

Create a Square connection

1. From the Analytics main menu, select **Import > Database and application**.
2. From the **New Connections** tab, in the **ACL DSN Connectors (Bundled)** section, select **Square.**

   > **Tip**
   > You can filter the list of available connectors by entering a search string in the **Filter connections** box. Connectors are listed alphabetically.

   The DSN Configuration dialog box appears.
3. In the **DSN Configuration** dialog box, click **Show Required**.
4. Provide values for the required fields, if any.
5. Click **Test Connection**.

   > **Note**
   > When you click the **Test Connection** button, it validates whether the connection and authentication details provided are correct. If you click **OK** without clicking Test Connection, the connection details are saved without any validation and may not work later if the values provided are incorrect.

The log in page for Square appears.

6. Provide the connection details and authenticate your login.

   When the connection is successful, a dialog box opens displaying the success message.

7. Click **OK** in the connection successful dialog box that appears.

   The connection to Square is established successfully.

8. In the **DSN Configuration** dialog box, click **OK**.

The connection for Square is saved to the **Existing Connections** tab. In the future, you can reconnect to Square from the saved connection.

Once the connection is established, the Data Access window opens to the **Staging Area** and you can begin importing data. For help importing data from Square, see "Working with the Data Access window" on page 357.

Rename the connection

When you create the Square data connector, you cannot provide a Connection name of your choice and the connection is saved to the **Existing Connections** tab with the same name as your data source. If you want to rename it, navigate to the **Existing Connections** tab, expand the **ACL Connectors (DSN)** section, click the ellipsis icon  corresponding to Square and select **Rename connection**.

Connecting to Stripe

Stripe provides online payment infrastructure services for e-commerce websites and mobile applications. You can use the Stripe data connector to import your organization's Stripe data.

> **Note**
> The Stripe data connector is provided by our data partner, CData. For information on any of the connection fields, refer the documentation available at the CData website.

Before you start

To connect to Stripe, you must gather the credentials to connect to the Stripe server.

For help gathering the connection prerequisites, contact the Stripe administrator in your organization. If your administrator cannot help you, you or your administrator should contact Stripe Support.

Create a Stripe connection

1. From the Analytics main menu, select **Import > Database and application**.
2. From the **New Connections** tab, in the **ACL DSN Connectors (Bundled)** section, select **Stripe.**

   > **Tip**
   > You can filter the list of available connectors by entering a search string in the **Filter connections** box. Connectors are listed alphabetically.

   The DSN Configuration dialog box appears.
3. In the **DSN Configuration** dialog box, click **Show Required**.
4. Provide values for the required fields, if any.
5. Click **Test Connection**.

   > **Note**
   > When you click the **Test Connection** button, it validates whether the connection and authentication details provided are correct. If you click **OK** without clicking Test Connection, the connection details are saved without any validation and may not work later if the values provided are incorrect.

   The log in page for Stripe appears.
6. Provide the connection details and authenticate your login.

When the connection is successful, a dialog box opens displaying the success message.

7. Click **OK** in the connection successful dialog box that appears.

   The connection to Stripe is established successfully.

8. In the **DSN Configuration** dialog box, click **OK**.

The connection for Stripe is saved to the **Existing Connections** tab. In the future, you can reconnect to Stripe from the saved connection.

Once the connection is established, the Data Access window opens to the **Staging Area** and you can begin importing data. For help importing data from Stripe, see "Working with the Data Access window" on page 357.

Rename the connection

When you create the Stripe data connector, you cannot provide a Connection name of your choice and the connection is saved to the **Existing Connections** tab with the same name as your data source. If you want to rename it, navigate to the **Existing Connections** tab, expand the **ACL Connectors (DSN)** section, click the ellipsis icon  corresponding to Stripe and select **Rename connection**.

Connecting to SugarCRM

SugarCRM is an open source Customer Relationship Management (CRM) software. You can use the SugarCRM data connector to import your organization's SugarCRM data.

> **Note**
> The SugarCRM data connector is provided by our data partner, CData. For information on any of the connection fields, refer the documentation available at the CData website.

Before you start

To connect to SugarCRM, you must gather the following:

- API version of SugarCRM installed
- URL of the SugarCRM account
- credentials to connect to the SugarCRM account

For help gathering the connection prerequisites, contact the SugarCRM administrator in your organization. If your administrator cannot help you, you or your administrator should contact SugarCRM Support.

Create a SugarCRM connection

1. From the Analytics main menu, select **Import > Database and application**.
2. From the **New Connections** tab, in the **ACL DSN Connectors (Bundled)** section, select **SugarCRM.**

   > **Tip**
   > You can filter the list of available connectors by entering a search string in the **Filter connections** box. Connectors are listed alphabetically.

   The DSN Configuration dialog box appears.
3. In the **DSN Configuration** dialog box, click **Show Required**.
4. Provide values for the required fields, if any.

5. Click **Test Connection**.

> **Note**
> When you click the **Test Connection** button, it validates whether the connection and authentication details provided are correct. If you click **OK** without clicking Test Connection, the connection details are saved without any validation and may not work later if the values provided are incorrect.

   The log in page for SugarCRM appears.

6. Provide the connection details and authenticate your login.

   When the connection is successful, a dialog box opens displaying the success message.

7. Click **OK** in the connection successful dialog box that appears.

   The connection to SugarCRM is established successfully.

8. In the **DSN Configuration** dialog box, click **OK**.

The connection for SugarCRM is saved to the **Existing Connections** tab. In the future, you can reconnect to SugarCRM from the saved connection.

Once the connection is established, the Data Access window opens to the **Staging Area** and you can begin importing data. For help importing data from SugarCRM, see "Working with the Data Access window" on page 357.

Rename the connection

When you create the SugarCRM data connector, you cannot provide a Connection name of your choice and the connection is saved to the **Existing Connections** tab with the same name as your data source. If you want to rename it, navigate to the **Existing Connections** tab, expand the **ACL Connectors (DSN)** section, click the ellipsis icon corresponding to SugarCRM and select **Rename connection**.

Connecting to SurveyMonkey

SurveyMonkey is a cloud-based free online survey tool to gather responses from people. You can use the SurveyMonkey data connector to import data your organization's SurveyMonkey data.

> **Note**
> The SurveyMonkey data connector is provided by our data partner, CData. For information on any of the connection fields, refer the documentation available at the CData website.

Before you start

To connect to SurveyMonkey, you must gather the credentials to connect to the SurveyMonkey account.

For help gathering the connection prerequisites, contact the SurveyMonkey administrator in your organization. If your administrator cannot help you, you or your administrator should contact SurveyMonkey Support.

Create a SurveyMonkey connection

1. From the Analytics main menu, select **Import > Database and application**.
2. From the **New Connections** tab, in the **ACL DSN Connectors (Bundled)** section, select **SurveyMonkey.**

   > **Tip**
   > You can filter the list of available connectors by entering a search string in the **Filter connections** box. Connectors are listed alphabetically.

   The DSN Configuration dialog box appears.
3. In the **DSN Configuration** dialog box, click **Show Required**.
4. Provide values for the required fields, if any.

5. Click **Test Connection**.

> **Note**
> When you click the **Test Connection** button, it validates whether the connection and authentication details provided are correct. If you click **OK** without clicking Test Connection, the connection details are saved without any validation and may not work later if the values provided are incorrect.

   The log in page for SurveyMonkey appears.

6. Provide the connection details and authenticate your login.

   When the connection is successful, a dialog box opens displaying the success message.

7. Click **OK** in the connection successful dialog box that appears.

   The connection to SurveyMonkey is established successfully.

8. In the **DSN Configuration** dialog box, click **OK**.

The connection for SurveyMonkey is saved to the **Existing Connections** tab. In the future, you can reconnect to SurveyMonkey from the saved connection.

Once the connection is established, the Data Access window opens to the **Staging Area** and you can begin importing data. For help importing data from SurveyMonkey, see "Working with the Data Access window" on page 357.

Rename the connection

When you create the SurveyMonkey data connector, you cannot provide a Connection name of your choice and the connection is saved to the **Existing Connections** tab with the same name as your data source. If you want to rename it, navigate to the **Existing Connections** tab, expand the **ACL Connectors (DSN)** section, click the ellipsis icon corresponding to SurveyMonkey and select **Rename connection**.

Connecting to Sybase

Sybase is a database management system (DBMS) software to manage and analyze information in relational databases. You can use the Sybase data connector to import your organization's Sybase data.

> **Note**
> The Sybase data connector is provided by our data partner, CData. For information on any of the connection fields, refer the documentation available at the CData website.

Before you start

To connect to Sybase, you must gather the following:

- credentials to connect to the Sybase database
- port for the Sybase database
- name of the server running the Sybase database
- name of the Sybase database

For help gathering the connection prerequisites, contact the Sybase administrator in your organization. If your administrator cannot help you, you or your administrator should contact Sybase Support.

Create a Sybase connection

1. From the Analytics main menu, select **Import > Database and application**.
2. From the **New Connections** tab, in the **ACL DSN Connectors (Bundled)** section, select **Sybase.**

   > **Tip**
   > You can filter the list of available connectors by entering a search string in the **Filter connections** box. Connectors are listed alphabetically.

   The DSN Configuration dialog box appears.
3. In the **DSN Configuration** dialog box, click **Show Required**.
4. Provide values for the required fields, if any.

5. Click **Test Connection**.

> **Note**
> When you click the **Test Connection** button, it validates whether the connection and authentication details provided are correct. If you click **OK** without clicking Test Connection, the connection details are saved without any validation and may not work later if the values provided are incorrect.

   The log in page for Sybase appears.

6. Provide the connection details and authenticate your login.

   When the connection is successful, a dialog box opens displaying the success message.

7. Click **OK** in the connection successful dialog box that appears.

   The connection to Sybase is established successfully.

8. In the **DSN Configuration** dialog box, click **OK**.

The connection for Sybase is saved to the **Existing Connections** tab. In the future, you can reconnect to Sybase from the saved connection.

Once the connection is established, the Data Access window opens to the **Staging Area** and you can begin importing data. For help importing data from Sybase, see "Working with the Data Access window" on page 357.

Rename the connection

When you create the Sybase data connector, you cannot provide a Connection name of your choice and the connection is saved to the **Existing Connections** tab with the same name as your data source. If you want to rename it, navigate to the **Existing Connections** tab, expand the **ACL Connectors (DSN)** section, click the ellipsis icon  corresponding to Sybase and select **Rename connection**.

Connecting to Sybase IQ

Sybase IQ is a high-performance relational database server designed for data warehousing that optimizes business intelligence, data warehousing, and big data analytics solutions. You can use the Sybase IQ data connector to import your organization's Sybase IQ data.

> **Note**
> The Sybase IQ data connector is provided by our data partner, CData. For information on any of the connection fields, refer the documentation available at the CData website.

Before you start

To connect to Sybase IQ, you must gather the following:

- credentials to connect to the Sybase IQ account
- port for the Sybase IQ database
- name of the server running the Sybase IQ or SAP SQL Anywhere database
- name of the Sybase IQ or SAP SQL Anywhere database

For help gathering the connection prerequisites, contact the Sybase IQ administrator in your organization. If your administrator cannot help you, you or your administrator should contact Sybase IQ Support.

Create a Sybase IQ connection

1. From the Analytics main menu, select **Import > Database and application**.
2. From the **New Connections** tab, in the **ACL DSN Connectors (Bundled)** section, select **Sybase IQ.**

   > **Tip**
   > You can filter the list of available connectors by entering a search string in the **Filter connections** box. Connectors are listed alphabetically.

   The DSN Configuration dialog box appears.
3. In the **DSN Configuration** dialog box, click **Show Required**.
4. Provide values for the required fields, if any.

5. Click **Test Connection**.

> **Note**
> When you click the **Test Connection** button, it validates whether the connection and authentication details provided are correct. If you click **OK** without clicking Test Connection, the connection details are saved without any validation and may not work later if the values provided are incorrect.

   The log in page for Sybase IQ appears.

6. Provide the connection details and authenticate your login.

   When the connection is successful, a dialog box opens displaying the success message.

7. Click **OK** in the connection successful dialog box that appears.

   The connection to Sybase IQ is established successfully.

8. In the **DSN Configuration** dialog box, click **OK**.

The connection for Sybase IQ is saved to the **Existing Connections** tab. In the future, you can reconnect to Sybase IQ from the saved connection.

Once the connection is established, the Data Access window opens to the **Staging Area** and you can begin importing data. For help importing data from Sybase IQ, see "Working with the Data Access window" on page 357.

Rename the connection

When you create the Sybase IQ data connector, you cannot provide a Connection name of your choice and the connection is saved to the **Existing Connections** tab with the same name as your data source. If you want to rename it, navigate to the **Existing Connections** tab, expand the **ACL Connectors (DSN)** section, click the ellipsis icon corresponding to Sybase IQ and select **Rename connection**.

Connecting to Tenable.sc

Tenable SecurityCenter is a vulnerability management solution that provides real-time visibility into the security posture of your IT setup. It also consolidates and evaluates vulnerability data across your organization, and prioritizes security risks. You can use the Tenable.sc data connector to import your company's Tenable SecurityCenter data.

Before you start

To connect to Tenable SecurityCenter, you must gather the following:

- the host name of the Tenable.sc instance
- the correct connection port for the Tenable.sc instance
- the correct authentication method to connect to the instance
- the user credentials to connect to the instance

For help gathering the connection prerequisites, contact the Tenable SecurityCenter administrator in your organization. If your administrator cannot help you, you or your administrator should contact Tenable SecurityCenter Support.

Create a Tenable SecurityCenter connection

1. From the Analytics main menu, select **Import > Database and application**.
2. From the **New Connections** tab, in the **ACL Connectors** section, select **Tenable.sc.**

   > **Tip**
   > You can filter the list of available connectors by entering a search string in the **Filter connections** box. Connectors are listed alphabetically.

3. In the **Data Connection Settings** panel, enter the connection settings and at the bottom of the panel, click **Save and Connect**.

   You can accept the default **Connection Name**, or enter a new one.

The connection for Tenable.sc is saved to the **Existing Connections** tab. In the future, you can reconnect to Tenable.sc from the saved connection.

Once the connection is established, the Data Access window opens to the **Staging Area** and you can begin importing data. For help importing data from Tenable.sc, see "Working with the Data Access window" on page 357.

Connection settings

Basic settings

Setting	Description	Example
Host	The host name or IP address of the instance running Tenable.sc.	https://sc.tenable.online/
Port	The port for the Tenable.sc instance.	443
Authentication Method	The authentication method to connect to the Tenable.sc instance. The options available are: ◦ **API Keys -** Use Access Key and Secret Key for authentication. ◦ **UserName/Password -** Provide username and password for authentication. **Note** To use API key authorization, your organization must have Tenable.sc 5.13.x or later vesion.	API Keys
User	The Tenable.sc user account to authenticate against the Tenable.sc instance.	
Password	The password used to authenticate the user with the Tenable.sc instance.	
Access Key	The API Access Key to authenticate on the Tenable.sc instance.	
Secret Key	The API secret key to authenticate on the	

Setting	Description	Example
	Tenable.sc instance.	

Advanced settings

Setting	Description	Example
Limit Key Size	The maximum length of a primary key column.	255
Map To Long Varchar	This property controls whether a column is returned as SQL_ LONGVARCHAR or not.	-1
Map To WVarchar	This property controls whether string types map to SQL_ WVARCHAR instead of SQL_ VARCHAR.	
Pseudo Columns	This property indicates whether to include pseudo columns as columns to the table. The value of this connection setting is of the format "Table1=Column1, Table1=Column2, Table2=Column3".	
SSL Server Certificate	The certificate that must be accepted from the server when connecting using SSL. You can provide any of the following: ◦ a full PEM certificate ◦ path to a local file containing the certificate ◦ public key ◦ MD5 or SHA1 thumbprint (hex values can also be either space or colon separated) Any other certificate that is not trusted by the machine is rejected.	
Convert Datetime To GMT	Converts datetime fields to GMT time zone during import. If this option is disabled, the datetime value is converted to the operating system time zone of the system running Analytics.	

Setting	Description	Example
Proxy Authentication Scheme	Specifies the authentication scheme to authenticate to the Proxy Server. The options available are as follows: ◦ **BASIC -** The driver performs HTTP BASIC authentication. ◦ **DIGEST -** The driver performs HTTP DIGEST authentication. ◦ **NEGOIATE -** The driver retrieves an NTLM or Kerberos token, based on the protocol set for authentication. ◦ **PROPRIETARY -** The driver does not generate an NTLM or Kerberos token. You must supply this token in the Authorization header of the HTTP request.	BASIC
Proxy Auto Detect	Specifies whether to use the system proxy settings or not. The value set for this option takes precedence over other proxy settings. So, if you want to use custom proxy settings, disable this option.	
Proxy User	The user name to authenticate to the Proxy Server.	
Proxy Password	The password to be used with the Proxy User to authenticate to the Proxy Server.	
Proxy Server	The hostname or IP address of a proxy to route HTTP traffic through.	
Proxy Port	The TCP port the Proxy Server is running on.	
Proxy SSL Type	The SSL type to use when connecting to the ProxyServer proxy. The options available are as follows: ◦ **AUTO -** If the URL is an HTTPS URL, the driver will use the TUNNEL option. If the URL is an HTTP URL, the component will use the NEVER option. ◦ **ALWAYS -** The connection is always SSL enabled. ◦ **NEVER -** The connection is not	AUTO

Setting	Description	Example
	SSL enabled. ◦ **TUNNEL -** The connection is through a tunneling proxy. This option is enabled only when you provide the value for the **Proxy Server**.	

Connecting to Teradata

Teradata is a cloud data service. You can use the Teradata data connector to import your company's Teradata data.

> **Note**
> Analytics provides Teradata as an optional connector and if it is not available in your Data Access window, it is likely that the connector was not selected during installation. For more information, see "Install optional Analytics data connectors and Python engine" on page 2567.

Before you start

To connect to Teradata, you must gather the following:

- name or IP address of the Teradata database instance
- username
- password

For help gathering the connection prerequisites, contact the Teradata administrator in your organization. If your administrator cannot help you, you or your administrator should contact Teradata Support.

Create a Teradata connection

1. From the Analytics main menu, select **Import > Database and application**.
2. From the **New Connections** tab, in the **ACL Connectors** section, select **Teradata.**

   > **Tip**
   > You can filter the list of available connectors by entering a search string in the **Filter connections** box. Connectors are listed alphabetically.

3. In the **Data Connection Settings** panel, enter the connection settings and at the bottom of the panel, click **Save and Connect**.

   You can accept the default **Connection Name**, or enter a new one.

The connection for Teradata is saved to the **Existing Connections** tab. In the future, you can reconnect to Teradata from the saved connection.

Once the connection is established, the Data Access window opens to the **Staging Area** and you can begin importing data. For help importing data from Teradata, see "Working with the Data Access window" on page 357.

Connection settings

Basic settings

Setting	Description	Example
Name or IP address	The fully qualified domain name or IP address of the Teradata database instance.	
Use Integrated Security	Specifies whether the driver authenticates the connection using Single Sign-On (SSO) or Conventional Sign-On (CSO): ◦ Enabled - The driver uses SSO and authenticates the connection by using Teradata database credentials derived from the user information on the client machine. ◦ Disabled - The driver uses CSO and requires user to provide Teradata database credentials.	Disabled
Mechanism	The mechanism that the driver uses to authenticate the connection to the database: ◦ **KRBS** - Uses the Kerberos protocol. The application provides the user name and password. ◦ **LDAP** - Uses the LDAP protocol. The application provides the user name and password. ◦ **SPNEGO** - Supports Kerberos authentication for users that log on to Teradata database from Windows .NET clients. ◦ **TD2** - Uses the Teradata 2 mechanism, which requires users to provide a Teradata database username and password. ◦ **TDNEGO** - Uses the mechanism that is selected automatically through Teradata	LDAP

Setting	Description	Example
	Negotiating, which can include single sign-on.	
Username	User name to authenticate the Teradata database connection through the specified authentic-ation mechanism.	
Password	The password to access the database.	
Enable Teradata Wallet	Specifies whether the driver authenticates the connection using a Teradata Wallet reference string.	
Default Database	The name of the database to access when a Teradata connection is opened.	
Account String	Identifies an individual user account and is associated with a specific User Id.	
Session Character Set	The character set (character encoding) to use for the session. The default value is ASCII.	ASCII

Advanced settings

Setting	Description	Example
No Help Database	Specifies whether the Help Database is used: ◦ Enabled - SQLTables uses a SELECT statement when no wildcard characters are used in SQLTables. ◦ Disabled - The driver uses the HELP DATABASE command.	Disabled
Ignore Search Patterns	Specifies whether the underscore (_) and percent sign (%) characters are parsed as normal characters or as search wildcards.	
Enable Legacy Parser	Specifies whether to enable the legacy parser or not.	

Setting	Description	Example
Log Error Events	Specifies whether to log error events or not.	
Use Regional Settings for Decimal Symbol	Specifies whether the driver uses the regional settings for decimal symbols, or uses a period (.) regardless of the regional settings.	Enabled
Enable Data Encryption	Specifies whether the driver encrypts all communication with the database or authentication information only.	
Enable Extended Statement Information	Specifies whether extended statement information is used when it is available from the database. If this option is enabled, the ODBC API function SQLDescribeParam is supported.	
Session Mode	Specifies the session mode that the driver uses during sessions on the database: ◦ **ANSI** ◦ **Teradata** ◦ **System Default** The default value is System Default.	ANSI
Maximum Response Buffer	The maximum size of the response buffer for SQL requests, in kilobytes. The default value is 65536.	65536
TDMST Port Number	The port number used to access Teradata Database. The default value is 1025.	1025
Translation DLL Name	The full path to the .dll file that contains functions for translating all data that is transferred between the Teradata server and the driver.	
Translation Option	Options used by the Translation DLL file.	
Login Timeout	The number of seconds that the driver waits for a response when logging in to the database before canceling the operation.	20

Setting	Description	Example
	The default value is 20.	
DataSource DNS Entries	The number of entries defined in DNS for the datasource .	
Use TCP_NODELAY	Specifies whether TCP sends small packets immediately or waits to gather packets into a single, larger packet: ◦ Enabled - TCP immediately sends small packets. This option can avoid transmission delays but might increase network traffic. ◦ Disabled - TCP gathers small packets into a single larger packet. This option can reduce network traffic but might cause transmission delays.	Enabled
Use Null for Catalog Name	Specifies whether the driver sets any Catalog Name parameters to NULL.	
Enable Read Ahead	Specifies whether to request the next response message while the current message is being processed.	
Retry System Calls (EINTR)	Specifies whether the driver retries the socket system calls or returns a SQL_ERROR when an EINTR error occurs.	
Use DATE Data for TIMESTAMP Parameters	Specifies whether the driver sends DATE data for parameters that are bound as SQL_TIMESTAMP and SQL_C_TIMESTAMP.	
Use Custom Catalog Mode for 2.x Applications	If this option is enabled, provides backwards compatibility for ODBC 2.x applications that use noncompliant search patterns.	
Return Empty String in CREATE_ PARAMS Columns for SQL_ TIMESTAMP	Specifies whether the driver returns an empty string or the given value for the CREATE_PARAMS column when you call SQLGetTypeInfo for SQL_TIMESTAMP data.	
Return Max. CHAR/VARCHAR	Specifies whether the driver returns	

Setting	Description	Example
Length as 32K	a hard-coded value for the COLUMN_SIZE column when you call SQLGetTypeInfo for SQL_CHAR and SQL_VARCHAR data.	

Data connector updates

When you upgrade Analytics, the Robots Agent, or AX Server, you should test any of your scripts that import data using one of the Analytics data connectors (ACCESSDATA command).

The possibility exists that changes made by third-party data sources or ODBC driver vendors required updates to one or more of the data connectors. Scripted data connections may need to be updated in order to continue working correctly.

- **Re-run the import** – The easiest way to update a connection is to manually perform an import using the Data Access window in the upgraded version of Analytics. Copy the ACCESSDATA command from the log and use it to update your script.

  > **Note**
  > Before connecting to a data source and re-running the import, clear the connector cache to flush the existing set of table names.
  > In the **Existing Connections** tab in the Data Access window, beside the connector name, select ··· **> Clear cache**.

- **Update field specifications** – You may also need to update field specifications in the script body to align with table schema changes in the data source or ODBC driver. Possible changes include field names, field data types, and field and record lengths.
- **Check the results of any filtering** – You should also check the results of any filtering that you apply as part of the data import. Confirm that the import filtering is including and excluding records correctly.

Connecting to Twitter

Connect to live data in Twitter and access Tweets, Followers, Messages, Searches, and more. The connector uses **Application-only authentication**, therefore you must register an application with Twitter and obtain the necessary credentials.

> **Note**
> You cannot use this connector independently of Analytics. You can configure a DSN for the connector driver using the Windows **ODBC Data Source Administrator**, however you must test the DSN connection from within Analytics and not from the connector's Windows DSN configuration dialog.

Before you start

To connect to Twitter, you must register an application and obtain Oauth credentials from Twitter using the Twitter Application Management portal.

For help gathering the connection prerequisites, contact the Twitter administrator in your organization. If your administrator cannot help you, you or your administrator should contact Twitter Support.

Create a Twitter connection

1. From the Analytics main menu, select **Import > Database and application**.
2. From the **New Connections** tab, in the **ACL Connectors** section, select **Twitter.**

   > **Tip**
   > You can filter the list of available connectors by entering a search string in the **Filter connections** box. Connectors are listed alphabetically.

3. In the **Data Connection Settings** panel, enter the connection settings and at the bottom of the panel, click **Save and Connect**.

   You can accept the default **Connection Name**, or enter a new one.

The connection for Twitter is saved to the **Existing Connections** tab. In the future, you can reconnect to Twitter from the saved connection.

Once the connection is established, the Data Access window opens to the **Staging Area** and you can begin importing data. For help importing data from Twitter, see "Working with the Data Access window" on page 357.

Connection settings

Basic settings

Setting	Description	Example
OAuth Client Id	The client Id assigned when you register your application with Twitter's OAuth authorization server. OAuth requires you to register your application. As part of the registration, you will receive a client Id, sometimes also called a *consumer key*, and a client secret. You must specify both the OAuthClientId and OAuthClientSecret to connect to an OAuth server.	xvz1evFS4wEEPTGEFPHBog
OAuth Client Secret	The client secret assigned when you register your application with Twitter's OAuth authorization server. OAuth requires you to register your application. As part of the registration you will receive a client Id and a client secret, sometimes also called a *consumer secret*. You must specify both the OAuthClientId and OAuthClientSecret to connect to an OAuth server.	L8qq9PZyRg6ieKGEKhZolGC0vJWLw8iEJ88DRdyOg
OAuth Access Token	The access token for connecting using OAuth. The OAuthAc-	xvz1evFS4wEEPTGEFPHBo-g:L8qq9PZyRg6ieKGEKhZolGC0vJWLw8iEJ88DRdyOg

Setting	Description	Example
	cessToken property is used to connect using OAuth. The OAuthAccessToken is retrieved from the OAuth server as part of the authentication process. It has a server-dependent timeout and can be reused between requests. The access token is used in place of your username and password. The access token protects your credentials by keeping them on the server.	
OAuth Access Token Secret	The OAuth access token secret for connecting using OAuth. The OAuthAccessTokenSecret property is used to connect and authenticate using OAuth. The OAuthAccessTokenSecret is retrieved from the OAuth server as part of the authentication process. It is used with the OAuthAccessToken and can be used for multiple requests until it times out.	Ewy4p5VygSPOUI1rhbCIVZcLnu05Y23Md22F0AKQYh1Hg

Advanced settings

Setting	Description	Example
Convert Datetime to GMT	Converts datetime fields to GMT time zone during import. If false, the datetime value is converted to the operating system time zone of the machine running Analytics.	true

Setting	Description	Example
Limit Key Size	The maximum length of a primary key column. Setting the size to 0 will make the key length revert to the original length. This property makes the connector override the reported length of all the primary key columns.	255
Map to Long Varchar	Controls whether or not a column is returned as SQL_LONGVARCHAR. Use this setting to map any column larger than the specified size so they are reported as SQL_LONGVARCHAR instead of SQL_VARCHAR.	-1
Map To WVarchar	Controls whether or not string types map to SQL_WVARCHAR instead of SQL_VARCHAR. It is set by default. String columns must be mapped to SQL_WVARCHAR to accommodate various international character sets, so MapToWVarchar is set to true by default. You may set it to false to use SQL_VARCHAR instead.	true
Pseudo Columns	Indicates whether or not to include pseudo columns as columns to the table. This setting is particularly helpful in Entity Framework, which does not allow you to set a value for a pseudo column unless it is a table column. The value of this connection setting is of the format "Table1=Column1, Table1=Column2, Table2=Column3". You can use the "*" character to include all tables and all columns.	MyTable=*
Upper Case Identifiers	Report all identifiers in uppercase, including table and column names.	false
SSL Server Cert	The certificate to be accepted from the server when connecting using TLS/SSL. You can specify any of the following:	C:\cert.cer

Setting	Description	Example
	◦ full PEM certificate ◦ path to a local file containing the certificate ◦ the public key ◦ the MD5 Thumbprint (hex values can also be either space or colon separated) ◦ the SHA1 Thumbprint (hex values can also be either space or colon separated) If using a TLS/SSL connection, this property can be used to specify the TLS/SSL certificate to be accepted from the server. Any other certificate that is not trusted by the machine will be rejected.	
Support Enhanced SQL	Enhances SQL functionality beyond what can be supported through the API directly, by enabling in-memory client-side processing: ◦ **true** - the connector offloads as much of the SELECT statement processing as possible to IMAP and then processes the rest of the query in memory. In this way the driver can execute unsupported predicates, joins, and aggregation ◦ **false** - the connector limits SQL execution to what is supported by the IMAP API ## Execution of predicates The connector determines which of the clauses are supported by the data source and then pushes them to the source to get the smallest superset of rows that would satisfy the query. It then filters the rest of the rows locally. The filter operation is streamed, which enables the driver to filter effectively for even very large datasets. ## Execution of Joins The connector uses various techniques to join in memory. The driver trades off memory utilization	

Setting	Description	Example
	against the requirement of reading the same table more than once. ### Execution of Aggregates The connector retrieves all rows necessary to process the aggregation in memory.	
Proxy Auth Scheme	The authentication type to use to authenticate to the ProxyServer proxy. This value specifies the authentication type to use to authenticate to the HTTP proxy specified by ProxyServer and ProxyPort. **Note** The connector will use the system proxy settings by default, without further configuration needed; if you want to connect to another proxy, you will need to set ProxyAutoDetect to false, in addition to ProxyServer and ProxyPort. To authenticate, set ProxyAuthScheme and set ProxyUser and ProxyPassword, if needed. The authentication type can be one of the following: ◦ **BASIC** - The driver performs HTTP BASIC authentication ◦ **DIGEST** - The driver performs HTTP DIGEST authentication. ◦ **NEGOTIATE** - The driver retrieves an NTLM or Kerberos token based on the applicable protocol for authentication ◦ **PROPRIETARY** - The driver does not generate an NTLM or Kerberos token. You must supply this token in the Authorization header of the HTTP request	BASIC

Setting	Description	Example
Proxy Auto Detect	Indicates whether to use the system proxy settings or not. Set ProxyAutoDetect to FALSE to use custom proxy settings. This takes precedence over other proxy settings.	true
Proxy User	A user name to be used to authenticate to the ProxyServer proxy. The ProxyUser and ProxyPassword options are used to connect and authenticate against the HTTP proxy specified in ProxyServer. You can select one of the available authentication types in ProxyAuthScheme. If you are using HTTP authentication, set this to the username of a user recognized by the HTTP proxy. If you are using Windows or Kerberos authentication, set this property to a username in one of the following formats: ◦ user@domain ◦ domain\user	john_doe@example.com
Proxy Password	A password to be used to authenticate to the ProxyServer proxy. This property is used to authenticate to an HTTP proxy server that supports NTLM (Windows), Kerberos, or HTTP authentication. To specify the HTTP proxy, you can set ProxyServer and ProxyPort. To specify the authentication type, set ProxyAuthScheme. If you are using HTTP authentication, additionally set ProxyUser and ProxyPassword to HTTP proxy. If you are using NTLM authentication, set ProxyUser and ProxyPassword to your Windows password. You may also need these to complete Kerberos authentication	
Proxy Server	The hostname or IP address of a proxy to route HTTP traffic through.	206.174.193.115

Setting	Description	Example
	The hostname or IP address of a proxy to route HTTP traffic through. The driver can use the HTTP, Windows (NTLM), or Kerberos authentication types to authenticate to an HTTP proxy. By default, the driver uses the system proxy. If you need to use another proxy, set ProxyAutoDetect to false.	
Proxy Port	The TCP port the ProxyServer proxy is running on.	80
Proxy SSL Type	The SSL type to use when connecting to the ProxyServer proxy: ◦ **AUTO** - If the URL is an HTTPS URL, the connector will use the TUNNEL option. If the URL is an HTTP URL, the connector will use the NEVER option (default) ◦ **ALWAYS** - the connection is always SSL enabled ◦ **NEVER** - the connection is not SSL enabled ◦ **TUNNEL** - the connection is through a tunneling proxy: The proxy server opens a connection to the remote host and traffic flows back and forth through the proxy	AUTO

Streaming tables

Avoid querying tables that capture continuously updated data, such as the **TweetStream** table. Streaming tables are not archives of historical data, and only return live activity. These tables create a connection that remains open and can cause you to exceed your account's API rate limit.

Data connector updates

When you upgrade Analytics, the Robots Agent, or AX Server, you should test any of your scripts that import data using one of the Analytics data connectors (ACCESSDATA command).

The possibility exists that changes made by third-party data sources or ODBC driver vendors required updates to one or more of the data connectors. Scripted data connections may need to be updated in order to continue working correctly.

- **Re-run the import** – The easiest way to update a connection is to manually perform an import using the Data Access window in the upgraded version of Analytics. Copy the ACCESSDATA command from the log and use it to update your script.

  > **Note**
  > Before connecting to a data source and re-running the import, clear the connector cache to flush the existing set of table names.
  > In the **Existing Connections** tab in the Data Access window, beside the connector name, select ··· **> Clear cache**.

- **Update field specifications** – You may also need to update field specifications in the script body to align with table schema changes in the data source or ODBC driver. Possible changes include field names, field data types, and field and record lengths.
- **Check the results of any filtering** – You should also check the results of any filtering that you apply as part of the data import. Confirm that the import filtering is including and excluding records correctly.

Twitter data connector changes

Specific changes made to the Twitter data connector are listed below.

Analytics version	Change
14.2	The DirectMessagesSent and the DirectMessagesReceived tables have been removed, and they are replaced by the new DirectMessages table.
	The data type of the IdLong field in the Tweets table has been changed from long to string.

Connecting to UPS

UPS is a global shipping and logistics firm that provide live tracking data from reporting tools, databases, and other custom applications. You can use the UPS data connector to import your organization's UPS data.

> **Note**
> The UPS data connector is provided by our data partner, CData. For information on any of the connection fields, refer the documentation available at the CData website.

Before you start

To connect to UPS, you must gather the following:

- the server to process requests to UPS
- the access key to connect to the UPS server
- credentials to connect to the UPS server
- your UPS account number

For help gathering the connection prerequisites, contact the UPS administrator in your organization. If your administrator cannot help you, you or your administrator should contact UPS Support.

Create a UPS connection

1. From the Analytics main menu, select **Import > Database and application**.
2. From the **New Connections** tab, in the **ACL DSN Connectors (Bundled)** section, select **UPS.**

   > **Tip**
   > You can filter the list of available connectors by entering a search string in the **Filter connections** box. Connectors are listed alphabetically.

   The DSN Configuration dialog box appears.
3. In the **DSN Configuration** dialog box, click **Show Required**.
4. Provide values for the required fields, if any.

5. Click **Test Connection**.

> **Note**
> When you click the **Test Connection** button, it validates whether the connection and authentication details provided are correct. If you click **OK** without clicking Test Connection, the connection details are saved without any validation and may not work later if the values provided are incorrect.

   The log in page for UPS appears.

6. Provide the connection details and authenticate your login.

   When the connection is successful, a dialog box opens displaying the success message.

7. Click **OK** in the connection successful dialog box that appears.

   The connection to UPS is established successfully.

8. In the **DSN Configuration** dialog box, click **OK**.

The connection for UPS is saved to the **Existing Connections** tab. In the future, you can reconnect to UPS from the saved connection.

Once the connection is established, the Data Access window opens to the **Staging Area** and you can begin importing data. For help importing data from UPS, see "Working with the Data Access window" on page 357.

Rename the connection

When you create the UPS data connector, you cannot provide a Connection name of your choice and the connection is saved to the **Existing Connections** tab with the same name as your data source. If you want to rename it, navigate to the **Existing Connections** tab, expand the **ACL Connectors (DSN)** section, click the ellipsis icon corresponding to UPS and select **Rename connection**.

Connecting to USPS

United States Postal Service (USPS) is a shipping and mailing service and provides live tracking data from reporting tools, databases, and other custom applications. You can use the USPS data connector to import your organization's USPS data.

> **Note**
> The USPS data connector is provided by our data partner, CData. For information on any of the connection fields, refer the documentation available at the CData website.

Before you start

To connect to USPS, you must gather the following:

- the postage provider
- whether requests are sent through sandbox or production servers
- the account number of the shipper
- Stamps User ID to authenticate to Stamps server
- password to authenticate to the server

For help gathering the connection prerequisites, contact the USPS administrator in your organization. If your administrator cannot help you, you or your administrator should contact USPS Support.

Create a USPS connection

1. From the Analytics main menu, select **Import > Database and application**.
2. From the **New Connections** tab, in the **ACL DSN Connectors (Bundled)** section, select **USPS.**

   > **Tip**
   > You can filter the list of available connectors by entering a search string in the **Filter connections** box. Connectors are listed alphabetically.

   The DSN Configuration dialog box appears.
3. In the **DSN Configuration** dialog box, click **Show Required**.
4. Provide values for the required fields, if any.

5. Click **Test Connection**.

> **Note**
> When you click the **Test Connection** button, it validates whether the connection and authentication details provided are correct. If you click **OK** without clicking Test Connection, the connection details are saved without any validation and may not work later if the values provided are incorrect.

   The log in page for USPS appears.

6. Provide the connection details and authenticate your login.

   When the connection is successful, a dialog box opens displaying the success message.

7. Click **OK** in the connection successful dialog box that appears.

   The connection to USPS is established successfully.

8. In the **DSN Configuration** dialog box, click **OK**.

The connection for USPS is saved to the **Existing Connections** tab. In the future, you can reconnect to USPS from the saved connection.

Once the connection is established, the Data Access window opens to the **Staging Area** and you can begin importing data. For help importing data from USPS, see "Working with the Data Access window" on page 357.

Rename the connection

When you create the USPS data connector, you cannot provide a Connection name of your choice and the connection is saved to the **Existing Connections** tab with the same name as your data source. If you want to rename it, navigate to the **Existing Connections** tab, expand the **ACL Connectors (DSN)** section, click the ellipsis icon  corresponding to USPS and select **Rename connection**.

Connecting to xBase

xBase is a complex of data and note files for storing large amounts of formatted data in a structured form. You can use the xBase data connector to import your organization's xBase data.

> **Note**
> The xBase data connector is provided by our data partner, CData. For information on any of the connection fields, refer the documentation available at the CData website.

Before you start

To connect to xBase, you must gather the data source path to the xBase account.

For help gathering the connection prerequisites, contact the xBase administrator in your organization. If your administrator cannot help you, you or your administrator should contact xBase Support.

Create an xBase connection

1. From the Analytics main menu, select **Import > Database and application**.
2. From the **New Connections** tab, in the **ACL DSN Connectors (Bundled)** section, select **xBase.**

   > **Tip**
   > You can filter the list of available connectors by entering a search string in the **Filter connections** box. Connectors are listed alphabetically.

   The DSN Configuration dialog box appears.
3. In the **DSN Configuration** dialog box, click **Show Required**.
4. Provide values for the required fields, if any.
5. Click **Test Connection**.

   > **Note**
   > When you click the **Test Connection** button, it validates whether the connection and authentication details provided are correct. If you click **OK** without clicking Test Connection, the connection details are saved without any validation and may not work later if the values provided are incorrect.

   The log in page for xBase appears.
6. Provide the connection details and authenticate your login.

When the connection is successful, a dialog box opens displaying the success message.

7. Click **OK** in the connection successful dialog box that appears.

   The connection to xBase is established successfully.

8. In the **DSN Configuration** dialog box, click **OK**.

The connection for xBase is saved to the **Existing Connections** tab. In the future, you can reconnect to xBase from the saved connection.

Once the connection is established, the Data Access window opens to the **Staging Area** and you can begin importing data. For help importing data from xBase, see "Working with the Data Access window" on page 357.

Rename the connection

When you create the xBase data connector, you cannot provide a Connection name of your choice and the connection is saved to the **Existing Connections** tab with the same name as your data source. If you want to rename it, navigate to the **Existing Connections** tab, expand the **ACL Connectors (DSN)** section, click the ellipsis icon  corresponding to xBase and select **Rename connection**.

Connecting to ZenDesk

Zendesk is a customer support platform designed to improve customer relationships and helps organizations to connect with customers on any channel. You can use the Zendesk data connector to import your organization's Zendesk data.

> **Note**
> The Zendesk data connector is provided by our data partner, CData. For information on any of the connection fields, refer the documentation available at the CData website.

Before you start

To connect to Zendesk, you must gather the following information:

- the Zendesk support URL
- the credentials to authenticate to your Zendesk account
- whether to use Zendesk incremental API
- API token of the user authenticating to Zendesk

For help gathering the connection prerequisites, contact the ZenDesk administrator in your organization. If your administrator cannot help you, you or your administrator should contact ZenDesk Support.

Create a Zendesk connection

1. From the Analytics main menu, select **Import > Database and application**.
2. From the **New Connections** tab, in the **ACL DSN Connectors (Bundled)** section, select **ZenDesk.**

   > **Tip**
   > You can filter the list of available connectors by entering a search string in the **Filter connections** box. Connectors are listed alphabetically.

   The DSN Configuration dialog box appears.
3. In the **DSN Configuration** dialog box, click **Show Required**.
4. Provide values for the required fields, if any.

5. Click **Test Connection**.

> **Note**
> When you click the **Test Connection** button, it validates whether the connection and authentication details provided are correct. If you click **OK** without clicking Test Connection, the connection details are saved without any validation and may not work later if the values provided are incorrect.

   The log in page for ZenDesk appears.

6. Provide the connection details and authenticate your login.

   When the connection is successful, a dialog box opens displaying the success message.

7. Click **OK** in the connection successful dialog box that appears.

   The connection to ZenDesk is established successfully.

8. In the **DSN Configuration** dialog box, click **OK**.

The connection for ZenDesk is saved to the **Existing Connections** tab. In the future, you can reconnect to ZenDesk from the saved connection.

Once the connection is established, the Data Access window opens to the **Staging Area** and you can begin importing data. For help importing data from ZenDesk, see "Working with the Data Access window" on page 357.

Rename the connection

When you create the ZenDesk data connector, you cannot provide a Connection name of your choice and the connection is saved to the **Existing Connections** tab with the same name as your data source. If you want to rename it, navigate to the **Existing Connections** tab, expand the **ACL Connectors (DSN)** section, click the ellipsis icon  corresponding to ZenDesk and select **Rename connection**.

Import HighBond Projects data

You can create an Analytics table by importing data from the projects that you have permission to work with in HighBond Projects.

You can import a variety of different types of tables from Projects that collectively contain all the text-based information in all the active projects for a HighBond instance.

How the information can be useful

You can use the information imported from Projects to perform different types of analysis. For example:

- join the imported tables on key fields as one way of finding incomplete information in a project

  For more information, see "Joining tables imported from Projects" on the facing page.
- export the information from Analytics for use in various reporting tools, such as Tableau or Microsoft Power BI

  For more information, see "Connecting to Analytics from a third-party reporting application" on page 1348.

Record length restriction

The maximum record length you can import into Analytics is 32,767 characters, or 16,383 Unicode characters.

If you try to import data from Projects that exceeds the maximum you get an error message and the import fails. Try removing large narrative fields from the import to reduce the record length.

Steps

> **Note**
> You may be required to specify a password when you connect to HighBond. For more information, see "Password requirement" on page 690.

1. Select **Import > HighBond > Projects**.
2. In the **Select data to import** dialog box, double-click the organization folder.

3. Double-click the folder for the table you want to import.

> **Note**
> You can import only one table at a time to Analytics. You can subsequently join tables imported from Projects if they have a common key field.

4. Select the fields you want to import, or select **Select All** to import the entire table, and click **OK**.

   The data is imported from Projects. To improve usability of the imported table, any rich-text fields with HTML markup are positioned last.

5. In the **Save Data File As** dialog box, enter a name for the Analytics data file, and if necessary modify the location where the file will be saved, and click **Save**.
6. Enter a name for the Analytics table you are adding to the Analytics project, or keep the default name, and click **OK**.

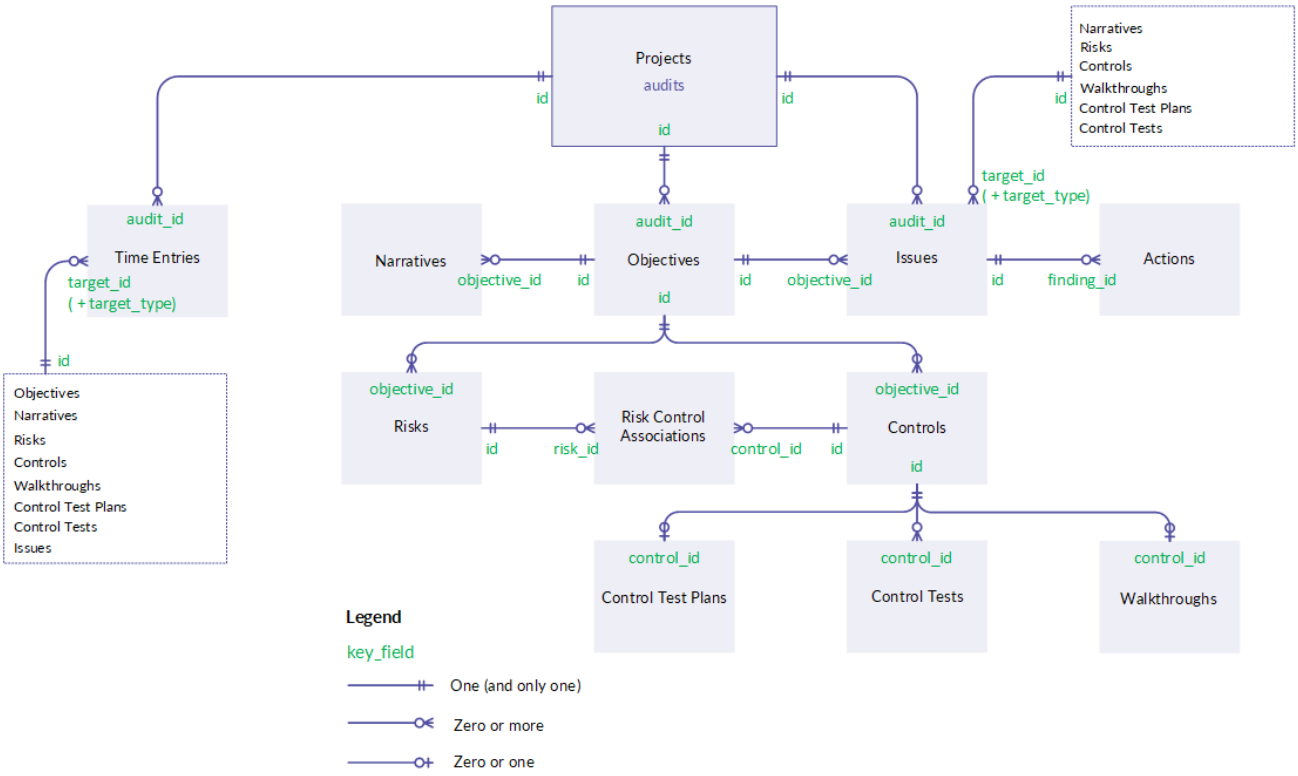
Joining tables imported from Projects

After you have imported tables from Projects to Analytics, you can join them to assemble part, or all, of the text-based information in your projects.

The diagram below shows how the various tables in Projects are related, and provides the common key fields that you can use to join the tables once you have imported them.

The `target_type` field in the Issues and Time Entries table is not a key field. Instead, it identifies the particular type of table each issue or time entry is related to. You can use the `target_type` field to filter a join so that it includes only the table relations you are interested in.

> **Tip**
> When you construct the join in Analytics, make `id` the primary key, and `<table>_id` the secondary key.
>
> Use the join type that includes all secondary records, or a many-to-many join. For more information, see "Joining tables" on page 889.



Password requirement

Password not required

You do not need to specify a password to import from HighBond if you used online activation to activate your copy of Analytics. The password is automatically created and sent to HighBond based on activation information stored on your computer.

Password required

You do need to specify a password to import from HighBond if you used offline activation to activate your copy of Analytics. The required password value is a HighBond access token.

> **Note**
> A password is also required if you use a script to import from HighBond, and you run the script in Robots, Analytics Exchange, or the Analysis App window.

Acquire a HighBond access token

1. On the Analytics main menu, select **Tools > HighBond Access Token**.

   The **Manage API tokens** page opens in your browser. You may be required to first sign in to Launchpad.

2. Do one of the following:
   - **Use an existing token** – In the **Token** column, click the partially masked token that you want to use and enter your HighBond account password. The unmasked token is displayed.

     > **Tip**
     > Use an existing token unless you have a reason for creating a new one. If the existing token does not work, create a new one.
     >
     > Using an existing token cuts down on the number of tokens you need to manage.

   - **Create a new token** – Click **Create token > Analytics** and enter your HighBond account password.

     A new Analytics token is created.

     > **Note**
     > If you are a Launchpad System Admin, you also have the option of creating an API token. You should reserve API tokens for their intended purpose, which is programmatic access to the HighBond platform.

3. Click **Copy** to copy the token.

   > **Tip**
   > Do not close the dialog box containing the token until you have successfully pasted the token.

4. In Analytics, paste the token into the password prompt.
5. In Launchpad, close the dialog box containing the token.

   If you created a new token, a partially masked version of the token is added to the top of your list of tokens.

   For more information, see Creating and managing access tokens.

> **Caution**
> Safeguard your access tokens like any account password. They contain information unique to your HighBond account. You should not share access tokens.

Import HighBond Results data

You can create an Analytics table by importing data from the collections that you have permission to work with in HighBond Results. You can import data tables and interpretations from individual control tests.

> **Note**
> In Results, a control test is called a "data analytic".

How the information can be useful

The ability to import data from Results to Analytics allows you to perform secondary or follow-up analysis on exceptions. After performing the analysis, you can export data back to Results in a round-trip process. For more information, see "Exporting exceptions to HighBond Results" on page 208.

Field name considerations when round-tripping Results data

If you are round-tripping data between Results and Analytics, you need to ensure that all field names in the Results table meet the more stringent Analytics field name requirements. If you do not, you risk misaligning your Analytics and Results data.

For example, any special characters in Results field names are automatically converted to underscores when they are imported into Analytics, which means the field names no longer match the original names in Results. If you then export the Analytics data back to the original table in Results, fields are no longer correctly matched.

To avoid this problem with data that you intend to round-trip, make sure that before you upload the data to Results from CSV or Excel files it meets these Analytics field name requirements:

- no special characters or spaces
- does not start with a number
- contains only alphanumeric characters, or the underscore character (_)

Steps

> **Note**
> You may be required to specify a password when you connect to HighBond. For more information, see "Password requirement" on page 694.

1. Select **Import > HighBond > Results**.
2. In the **Select data to import** dialog box, double-click the organization folder.
3. Navigate to the appropriate control test.

> **Note**
>
> You do not have the necessary permissions to access the data if:
>
> - the collection containing the control test does not appear
> - the message **Error retrieving list of interpretations** appears when you try to access the control test
>
> For more information, see Assigning privileges and roles in Results.
>
> For help with permissions, contact your company's HighBond account administrator, or Results administrator.

4. Select one of the following tables to import to Analytics:
   - an interpretation, if one exists
   - audit trail
   - comments
   - results table (double-click the **Table Field Selection** folder)

> **Note**
>
> You can import only one table at a time to Analytics.

> **Tip**
>
> You can join the results table, audit trail, and comments in Analytics using Record ID as the key field. Use the results table as the primary table in the join.

5. If you are importing the results table, select the individual fields you want to import from the following categories:
   - **Metadata** – fields containing user-generated and system-generated workflow information
   - **Extras** – collection and control test names, and Record ID

     Make sure to select Record ID if you intend to join the results table in Analytics.
   - **Data** – fields containing data that was imported to Results, or answers to a Results questionnaire
   - **Select All** – imports the entire table
6. Click **OK**.

   The data is imported from Results.
7. In the **Save Data File As** dialog box, enter a name for the Analytics data file, and if necessary modify the location where the file will be saved, and click **Save**.
8. Enter a name for the Analytics table you are adding to the Analytics project, or keep the default name, and click **OK**.

Password requirement

Password not required

You do not need to specify a password to import from HighBond if you used online activation to activate your copy of Analytics. The password is automatically created and sent to HighBond based on activation information stored on your computer.

Password required

You do need to specify a password to import from HighBond if you used offline activation to activate your copy of Analytics. The required password value is a HighBond access token.

> **Note**
> A password is also required if you use a script to import from HighBond, and you run the script in Robots, Analytics Exchange, or the Analysis App window.

Acquire a HighBond access token

1. On the Analytics main menu, select **Tools > HighBond Access Token**.

   The **Manage API tokens** page opens in your browser. You may be required to first sign in to Launchpad.

2. Do one of the following:
   - **Use an existing token** – In the **Token** column, click the partially masked token that you want to use and enter your HighBond account password. The unmasked token is displayed.

     > **Tip**
     > Use an existing token unless you have a reason for creating a new one. If the existing token does not work, create a new one.
     >
     > Using an existing token cuts down on the number of tokens you need to manage.

   - **Create a new token** – Click **Create token > Analytics** and enter your HighBond account password.

     A new Analytics token is created.

     > **Note**
     > If you are a Launchpad System Admin, you also have the option of creating an API token. You should reserve API tokens for their intended purpose, which is programmatic access to the HighBond platform.

3. Click **Copy** to copy the token.

> **Tip**
> Do not close the dialog box containing the token until you have successfully pasted the token.

4. In Analytics, paste the token into the password prompt.
5. In Launchpad, close the dialog box containing the token.

   If you created a new token, a partially masked version of the token is added to the top of your list of tokens.

   For more information, see Creating and managing access tokens.

> **Caution**
> Safeguard your access tokens like any account password. They contain information unique to your HighBond account. You should not share access tokens.

Structuring data with table layouts

When you import data into Analytics, or connect to data with Analytics, you need to **structure** the raw data in order to read it, and perform analysis on it. The structuring device in Analytics is called a **table layout**. Analytics automatically generates a table layout for every table in an Analytics project.

A table layout is a collection of metadata that describes and defines the raw data in the source data file or source location. Without a table layout, source data would appear like a mass of undifferentiated numbers and letters. The table layout imposes order.

The metadata elements in a table layout

A table layout specifies a number of metadata elements, including:

- **names** – field names – both physical field names, and display names of corresponding columns in views
- **types** – data category and data type of fields
- **physical characteristics** – starting position and byte length of fields
- **format** – format details such as the number of decimal places in numeric fields, and the date format in datetime fields
- **connection** – the location of the source data

How table layouts relate to Analytics tables

The diagram below illustrates the table layout in relation to the other components of an Analytics table.

For more information about how table layouts relate to Analytics tables generally, see "The structure of Analytics tables" on page 115.

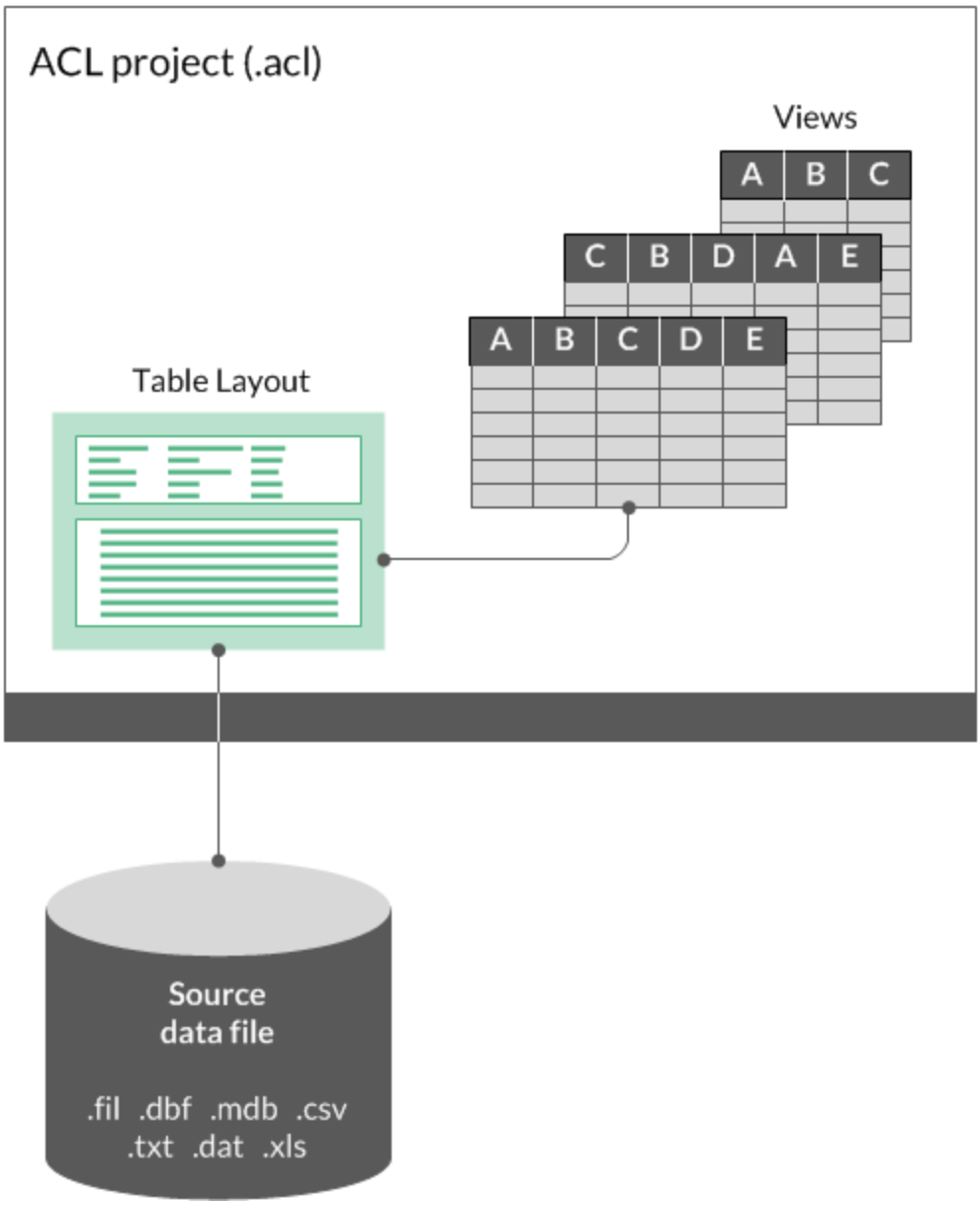
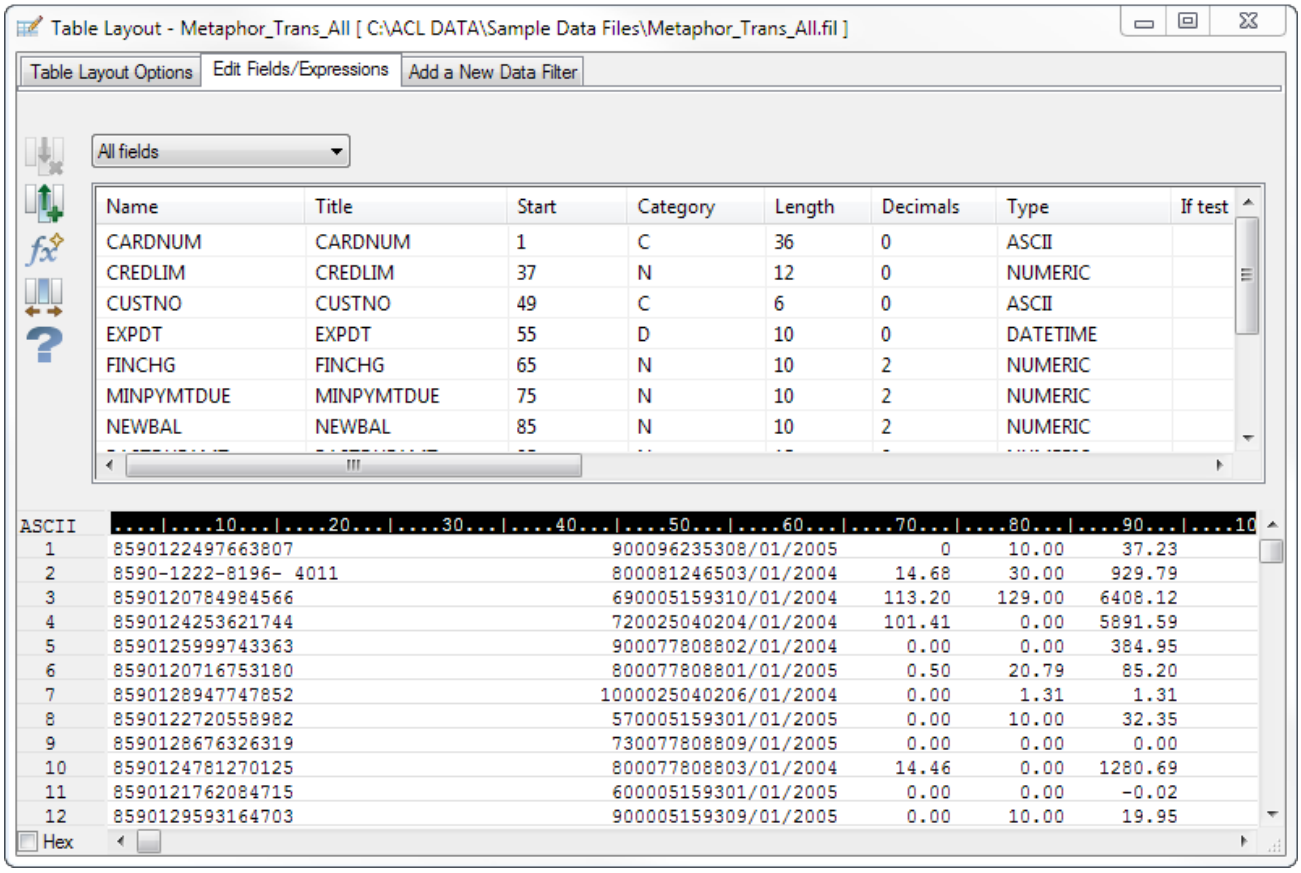


Table layout and source data file

The table layout for the `Metaphor_Trans_All` table, from the `ACL_Demo.acl` sample data project, appears below, followed by the associated source data file, `Metaphor_Trans_All.fil`.

You can clearly see how the table layout organizes the source data, and defines a number of fields.

Table layout



Source data file

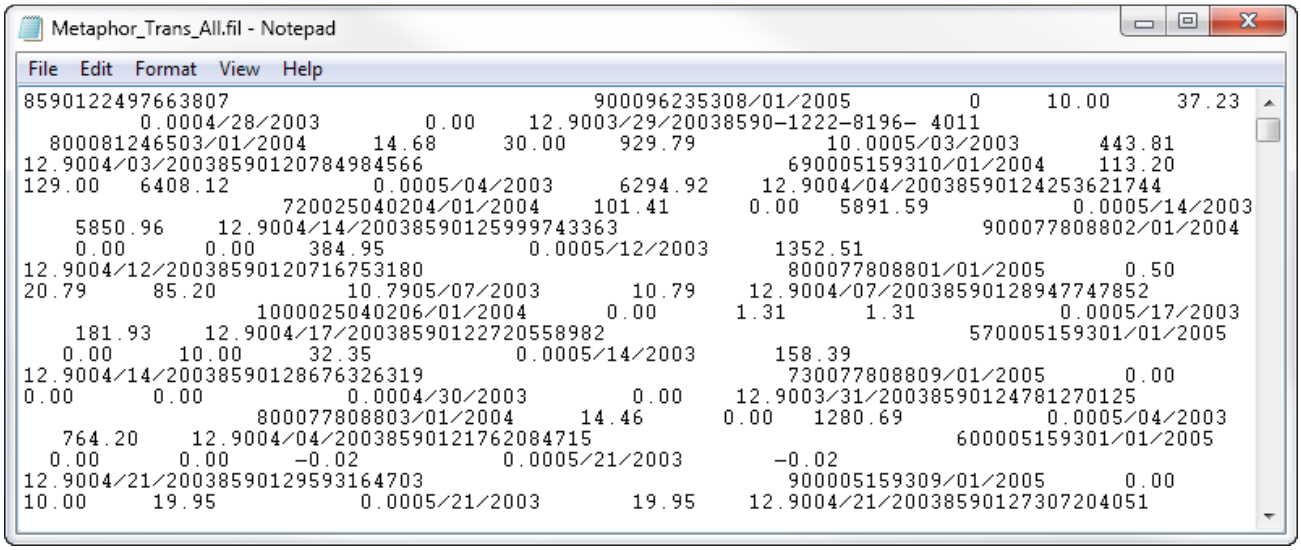


Table Layout dialog box

You use the **Table Layout** dialog box (shown above) to define and modify table layouts. You can access the dialog box at any time by selecting **Edit > Table Layout**. The dialog box includes three tabs for working with particular aspects of the table layout:

Tab	Description
Table Layout Options	Use this tab to configure general properties for the table layout, such as the record length, the data source associated with the table layout, and to add notes about the table layout.
Edit Field-s/Expressions	Use this tab to create, modify, or delete fields from the table layout. You can work with both physical data fields and computed fields.
Add a New Data Filter	Use this tab to define data filters, which are rules that specify which data from the data source to include or exclude from the record you are defining. Data filters in a table layout are different than filters in Analytics views, and are typically only necessary when you are unsuccessful in defining a data source using the options available in the **Data Definition Wizard**.

Working with table layouts

If a table layout automatically generated by Analytics provides everything you need, you can use it without modification. There may be occasions, however, when you need to edit the table layout.

You may also want to perform other operations involving table layouts, such as renaming, copying, importing, and exporting.

Edit a table layout

In some cases, you may need to edit the automatically created table layout, or create the table layout manually. You can add, remove, or modify fields or records in a table layout as necessary.

Why would I edit a table layout?

You may want to edit a table layout for any of the following reasons:

- **Add or delete fields** – You want to add or delete data fields, computed fields, or data filters.
- **Modify default field definitions** – You want to modify the Analytics-created field definitions to change the data type, the format of numeric or datetime fields, or other properties of individual fields.
- **Define overlapping fields** – You want to define overlapping fields.

  Overlapping fields share one or more byte positions in a record. For example, if the first six positions in a record are defined as a full date field, you could define the first two positions as a month field. The two fields are overlapping.
- **Correct errors** – You want to correct errors identified after you defined the table using the **Data Definition Wizard**.

  For example, data may be incorrectly displayed, and you may need to modify the record length or the skip length.
- **Create a table layout manually** – You selected the **Define Flat Files Manually** option in the **Options** dialog box.

  This option changes the default behavior of Analytics. After you select a flat file as a data source, and provide other basic information in the **Data Definition Wizard**, the **Table Layout** dialog box is displayed and you must define the record(s) and fields for the Analytics table manually.

Rename a table layout

You can rename a table layout. Renaming a table layout does not rename the associated data file. You cannot rename a table layout if the table is currently open.

> **Note**
>
> If you rename a table layout that is referenced in an Analytics script you must also update all references to the table in the script or the script will fail when it is run.

Show me how

1. If the table you want to rename is open, close it.
2. In the **Overview** tab, right-click the table layout and select **Rename**.
3. Type a new name for the table layout and press **Enter**.

> **Note**
>
> Table layout names are limited to 64 alphanumeric characters. The name can include the underscore character (_), but no other special characters, or any spaces. The name cannot start with a number.

Copy a table layout

You can copy a table layout to associate an identical table layout with the same Analytics data file or data source, and then subsequently modify the copied table layout. Copying and modifying a table layout may be easier than creating a new table layout from scratch.

Having more than one table layout associated with the same data file or data source allows you to define fields differently for the same data, create different subsets of fields, create different computed fields, and so on, which can be useful when working with large tables that would be unwieldy if all field definitions were included in a single table layout.

> **Caution**
>
> Be careful when you have more than one table layout associated with the same Analytics data file. If you have **Delete Data File with Table** selected in the **Options** dialog box, deleting any of the table layouts also deletes the data file, which means the data is no longer available for the remaining table layouts.
>
> Data files are deleted outright. They are not sent to the Windows Recycle Bin.

Show me how

1. In the **Overview** tab, right-click the table layout you want to copy and select **Copy**.
2. Right-click a project folder and select **Paste**.
3. If a message appears asking whether you want to copy or share field definitions, click **Copy** unless you have a specific reason for clicking **Share**.

> **Note**
>
> The message appears if **Don't Share Table Layouts** is deselected in the **Options** dialog box. For more information, see "Share a table layout" on the next page.

The table layout is copied and given an incrementing numeric suffix – for example, “Table2”. You can rename the table layout if necessary.

Share a table layout

You can share a table layout to associate the same table layout with two or more Analytics data files. Generally, you should maintain a separate table layout for each data file. However, sharing a single table layout can save labor by allowing you to centrally manage a table layout for multiple data files.

For more information, see **Don’t Share Table Layouts** in"Table options" on page 122.

> **Note**
>
> Sharing a table layout is not the same as copying a table layout and sharing a data file.
>
> - When you share a table layout, a single table layout is associated with two or more data files.
> - When you share a data file, two or more table layouts are associated with a single data file.

Show me how

1. Ensure that **Don’t Share Table Layouts** is deselected in the **Table** tab in the **Options** dialog box (**Tools > Options > Table**).
2. Do one of the following:
   - Perform an Analytics operation such as extract or sort that outputs results to a new data file with the same record structure as the source table.

     The results table and the source table now share the same table layout.
   - Copy and paste a table layout and click **Share** in the confirmation dialog box that appears.
3. If you copied a table layout, do the following:
   a. Right-click the new table layout and select **Link to New Source Data**.
   b. If the **Select File Location** dialog box is displayed, select the location where the data file or data source is located and click **OK**.

      You can select **Client** for a local or network location, or **Server** and a server profile for a location on an Analytics server.
   c. In the **Select File** dialog box, locate and select the new data file or data source and click **Open**.

   For more information, see "Modifying data sources for Analytics tables" on page 711.
4. If you have finished creating shared table layouts, select **Don’t Share Table Layouts** in the **Options** dialog box to prevent inadvertently creating shared table layouts in subsequent operations.

Copy a table layout from another Analytics project

You can copy a table layout from one Analytics project to another, which allows you to reuse the table layout, and the field definitions it contains, rather than creating them from scratch. In addition to saving labor, reusing table layouts, or sharing them with other Analytics users, ensures consistency. You can copy a single table layout, or multiple table layouts simultaneously.

If a table layout specifies an association with a particular Analytics data file (.fil), and a data file of the same name exists in the folder containing the destination Analytics project, the copied table layout is automatically associated with the data file in the destination folder. If there is no data file with the same name in the destination folder, you need to link the copied table layout to a new data source.

> **Note**
>
> The copied table layout and the data file it is associated with must match – that is, the structure of the data in the data file must match the field definitions specified by the table layout.
>
> Data structure refers to the data elements (fields) contained in a data file, the number and order of the fields, and the data type and length of the fields. If the table layout and the data file do not match, jumbled or missing data results.

Show me how

1. Open the project that will contain the copied table layout or table layouts.
2. In the **Overview** tab of the **Navigator**, right-click the Analytics project entry, or a project folder, and select **Copy from another Project > Table**.

   The Analytics project is the top-level folder in the treeview.
3. In the **Locate Project File** dialog box, locate and select the Analytics project that you want to copy the table layout or layouts from and click **Open**.
4. In the **Import** dialog box, complete any of the following tasks to add one or more table layouts to the **To *project_name*** list:
   - Double-click a table layout.
   - **Ctrl+click** multiple table layouts and then click the right-arrow button.
   - Click **Add All** to add all the table layouts.

   You can remove table layouts from the **To *project_name*** list by double-clicking an individual table layout, by using **Ctrl+click** to select multiple table layouts and then clicking the left-arrow button, or by clicking **Clear All**.
5. Click **OK** to copy the table layout or layouts into the destination project.

   If a table layout with the same name already exists in the project, the copied table layout is given an incrementing numeric suffix.
6. If you need to link the copied table layout to a new data source, see "Modifying data sources for Analytics tables" on page 711.

Import a table layout

You can import a table layout that exists as a separate .layout file outside an Analytics project, which allows you to reuse the table layout, and the field definitions it contains, rather than creating them from scratch. In addition to saving labor, reusing table layouts, or sharing them with other Analytics users, ensures consistency. You can import only one table layout at a time.

If a table layout specifies an association with a particular Analytics data file (.fil), and a data file of the same name exists in the folder containing the project, the imported table layout is automatically associated with the data file in the folder. If there is no data file with the same name in the project folder, you need to link the imported table layout to a new data source.

> **Note**
>
> The imported table layout and the data file it is associated with must match – that is, the structure of the data in the data file must match the field definitions specified by the table layout.
>
> Data structure refers to the data elements (fields) contained in a data file, the number and order of the fields, and the data type and length of the fields. If the table layout and the data file do not match, jumbled or missing data results.

Show me how

1. In the **Overview** tab of the **Navigator**, right-click the Analytics project entry, or a project folder, and select **Import Project Item > Table**.

   The Analytics project is the top-level folder in the treeview.
2. In the **Project** dialog box, locate and select a table layout file (.layout) and click **Open**.
3. Click **OK** in the confirmation dialog box.

   The table layout is imported into the project. If a table layout with the same name already exists in the project, the imported table layout is given an incrementing numeric suffix.
4. If you need to link the imported table layout to a new data source, see "Modifying data sources for Analytics tables" on page 711.

Export a table layout

You can export a table layout as a separate .layout file saved outside the Analytics project. A table layout exported as a separate file can later be imported into any Analytics project, which allows you to reuse the table layout, and the field definitions it contains, rather than creating them from scratch. You can export only one table layout at a time.

Show me how

1. Right-click the table layout in the **Overview** tab of the **Navigator** and select **Export Project Item**.
2. In the **Save As** dialog box, choose a location to save the table layout, rename the table layout if required, click **Save**, and click **OK** in the confirmation dialog box.

The table layout is exported to the location you specified.

> **Note**
>
> Limit the table layout name to 64 alphanumeric characters, not including the .layout extension, to ensure that the name is not truncated when the table layout is imported back into Analytics.
>
> The name can include the underscore character (_), but do not use any other special characters, or any spaces, or start the name with a number. Special characters, spaces, and a leading number are all replaced by the underscore character when the table layout is imported.

Delete a table layout

You can delete a table layout associated with an Analytics data file or a data source whenever necessary. Deleting a table layout also deletes the associated views, and any associated indexes or table relations.

> **Caution**
>
> Be careful when deleting table layouts. If you have **Delete Data File with Table** selected in the **Options** dialog box, deleting a table layout also deletes the data file, which means the data is no longer available.
>
> The **Delete** confirmation dialog box warns you if the associated data file will be deleted along with the table layout.
>
> Data files are deleted outright. They are not sent to the Windows Recycle Bin.

Show me how

1. In the **Overview** tab, right-click the table layout you want to delete and select **Delete**.
2. Click **Delete** in the confirmation dialog box.

Configuring properties for table layouts

You can change many of the basic properties of a table layout using the **Table Layout Options** tab in the **Table Layout** dialog box. Any properties or options that do not apply to the selected table layout, because of the data source it is associated with, are disabled.

The data preview area in the **Table Layout** dialog box dynamically updates based on any changes you make.

To configure properties for a table layout:

1. Select **Edit > Table Layout**.
2. Click the **Table Layout Options** tab.

   The **Media Type** drop-down list specifies how the data is physically accessed. This value is typically set to **Disk** and cannot be changed.
3. The **File Type** drop-down list specifies how records are stored in the data source. You can select from the following options:
   - **Fixed Record Length** – Select this option for data files in which each record has the same maximum length, and the position of each field is consistent from record to record.
   - **IBM Variable Record Length** – Select this option for data files in which records have a varying length.
   - **Text File (CR or CRLF)** – Select this option if the data file is a text file in which the end of each record is specified by a carriage return (CR), or a carriage return and line feed sequence (CRLF).
4. If you want to modify the character set used to display the data file, you can select the appropriate option from the **Character Type** drop-down list.

   You can select **ASCII**, **EBCDIC**, or **Unicode** (if you are using the Unicode edition of Analytics). If the character set selected does not match the encoding for the data file, the data in the data preview area is not readable.

   You can also change the character set by clicking the character set toggle switch at the top left corner of the data preview area.

   You can select the **Hex** checkbox at the bottom left corner of the data preview area to view the data in hexadecimal format. This option is useful if you are working with unprintable characters or compressed data, such as packed numeric data originating from an IBM mainframe computer, and you need to modify the **Record Length** or **Skip Length**.
5. If you want to modify the record length, increase or decrease the value in the **Record Length** text box.

   The record length refers to the length of each record in fixed length files, or the length of the longest record in variable length files. If the values in a field are shifted to the right the record length value probably needs to be increased. If the values in a field are shifted to the left, the

record length value probably needs to be decreased.

6. If you want to exclude data at the beginning of the file, modify the value in the **Skip Length** text box.

   The **Skip Length** value specifies the number of bytes to skip at the start of the file. The default value is zero, which means that the entire file is processed. To skip data, enter a value greater than zero. For example, if the first 32 bytes contain only header information, specify a value of 32 to omit this information.

7. If you want to add a note about the table layout, click **Edit Table Layout Note**  , enter the note text, and click **Close**  .
8. If data is incorrectly displayed in the data preview area, you can have Analytics attempt to identify the properties of the data file automatically by clicking **Analyze File**  .
9. If the data source is a delimited text file, and the data is incorrectly displayed in the data preview area (or no data is displayed), you may need to specify the field separator and delimiter characters manually:
   a. Click **Convert Delimited File**  .
   b. Enter the required **Field Separator** and **String Delimiter** values in the **Custom Delimit** dialog box and click **OK**.
   c. Enter a name for the new data file (.fil) and click **Save**.
10. Click **Close**  to close the **Table Layout** dialog box and save your changes.

Viewing table layout properties

You can use the procedure below to view or maintain certain table layout properties. To view or maintain table layout properties associated with displaying data, see "Configuring properties for table layouts" on page 706.

To view a table layout's properties:

1. Right-click the Analytics table in the **Overview** tab in the **Navigator**.
2. Select **Properties**.
3. In the **Table Properties** dialog box, click on the following tabs to view or modify table properties:
   - **General** – This tab displays the basic properties of the Analytics data file (.fil), or the data source, associated with the table layout: the name, the location, the last modification date and time, and the physical size of the file.

     You can click **Open file location** to navigate directly to the folder containing the Analytics data file or data source.
   - **Notes** – This tab displays any notes associated with the table layout. You can modify existing notes or add new notes. For more information, see "Add or edit table layout notes" on page 177.
   - **Views** – This tab displays all the views in the table layout. You can maintain the views in the table layout from this tab. For more information, see "Working with views" on page 770.
   - **Indexes** – This tab displays any indexes associated with the table layout. You can modify existing indexes or add new indexes. For more information, see "Maintain indexes" on page 1141.

     > **Note**
     > The table must be open in order to make any changes to views or indexes.

4. Click **OK** to close the dialog box and save any changes you have made.

Updating data in Analytics tables

Depending on the data source, you can update an Analytics table with the current contents of the table's data source without needing to redefine the table layout. You can update the Analytics table whenever necessary, as long as there are no changes to the structure of the data in the data source.

Data sources that support updating

Analytics tables with the following data sources can be updated using the **Refresh from Source** option.

- File-based data sources:
  - delimited text
  - Microsoft Access
  - Microsoft Excel
  - Adobe Acrobat (PDF)
  - Print Image (Report)
  - SAP private file format/DART
  - XML
  - XBRL
- ODBC-compliant data sources:
  - any file or database that you connect to using the Data Access window or the `IMPORT ODBC` command

> **Note**
>
> An Analytics table cannot be updated from a file-based data source if a password is required to access the file. The one exception is updating from a PDF file.
>
> Updating from a database does support the use of a password.

How updating works

When you update a table using **Refresh from Source**, the command used to originally define the table is re-run, and any new records or changes in the data source are added to the Analytics table. As long as the structure of the data in the data source remains the same the field definitions in the Analytics table layout still apply.

If any of the new field values in the data source are longer than the specified field lengths in the Analytics table layout, the values are truncated in the Analytics table. To acquire the full values, you need to define the Analytics table again instead of using **Refresh from Source**.

Update a table from the Overview tab in the Navigator

1. Ensure that the file with the source data is closed.
2. Right-click the Analytics table you want to update and select **Refresh from Source**.
3. Click **Yes** in the confirmation dialog box.

Update a table using the Table Layout dialog box

1. Ensure that the file with the source data is closed.
2. Open the Analytics table you want to update.
3. Select **Edit > Table Layout**.
4. Select the **Table Layout Options** tab.
5. Click **Refresh from Source File**  .
6. Click **Yes** in the confirmation dialog box.

Modifying data sources for Analytics tables

If multiple source data files have an identical structure, you can save effort by using the same Analytics table layout with all the files.

How it works

Say you define an Analytics table from a source data file that contains January invoice data. The next month, the source file containing February data is identically structured, and the same is true for subsequent months. Instead of recreating a table layout for each month, you can continue to reuse the same table layout.

When you get the data for February, you can either relink the January table layout with the new data file, or you can copy the January table and then modify the link so that you retain tables for both months.

Supported data sources

You can link a table layout to a new data source if the source is a delimited text file, a text file with line breaks, or an Analytics source data file (.fil).

Identical data structure requirement

A table layout and a data file it is associated with must match. That is, the structure of the data in the data file must match the field definitions specified by the table layout.

Data structure refers to the data elements (fields) contained in a data file, the number and the order of the fields, and the data type and length of the fields. If a table layout and a data file do not match, jumbled or missing data results.

Modify a table data source from the Overview tab in the Navigator

1. In the **Navigator**, right-click the table you want to update and select **Link to New Source Data**.
2. If the **Select File Location** dialog box is displayed, select the location where the data source is located and click **OK**.

You can select **Client** for a local or network location, or **Server** and a server profile for a location on an Analytics server.

3. In the **Select File** dialog box, locate and select the new data source and click **Open**.

Modify a table data source using the Table Layout dialog box

1. In the **Navigator**, select the table you want to update and select **Edit > Table Layout**.
2. Select the **Table Layout Options** tab.
3. Click **Link to New Source Data**  .
4. If the **Select File Location** dialog box is displayed, select the location where the data source is located and click **OK**.

   You can select **Client** for a local or network location, or **Server** and a server profile for a location on an Analytics server.
5. In the **Select File** dialog box, locate and select the new data source and click **Open**.

Defining fields

In an Analytics table layout, a field is a single unit of data, such as employee ID, that together with other units of data form a record.

You can define two types of fields in Analytics table layouts:

- Physical fields
- Computed fields

All fields in Analytics must be assigned a data type (Character, Numeric, Datetime, or Logical) that determines how the values in the physical or computed fields are displayed and processed.

Physical fields

A physical field corresponds to actual data physically present in a data source, such as a file or a database. For example, a physical field named **Amount** could contain sales amounts, such as $88.50, $123.00, and so on.

In a table layout, a record specifies where data is located in the data source, and a physical field specifies the location of field data in the record.

Before you can open an Analytics table, the table layout must have at least one physical field defined. Typically, the physical fields in a table layout are automatically defined by Analytics when you define and import data using the Data Definition Wizard or the Data Access window. You can also manually define physical fields, if necessary, in the **Table Layout** dialog box.

For more information, see "Physical fields" on page 715.

Computed fields

A computed field is a "virtual field", created using an Analytics expression, that allows you to compute or calculate values that are not physically present in the data source. For example, you could create a computed field named **Total Amount** that uses the expression `Amount * 1.05` to calculate total amounts including a 5% sales tax.

Amount field (physical)	Analytics expression	Total Amount field (computed)
$88.50	`Amount * 1.05`	$92.93
$123.00	`Amount * 1.05`	$129.15

Although computed fields do not correspond directly to physical data, they often reference one or more physical fields, such as the **Amount** field in the example above. Computed field expressions

may also reference other computed fields, or they may contain functions that do not require fields as input parameters.

You define computed fields in the **Table Layout** dialog box.

For more information, see "Computed fields" on page 722.

Physical fields

In an Analytics table layout, a field that corresponds to actual physical data in a data source is called **a physical field**.

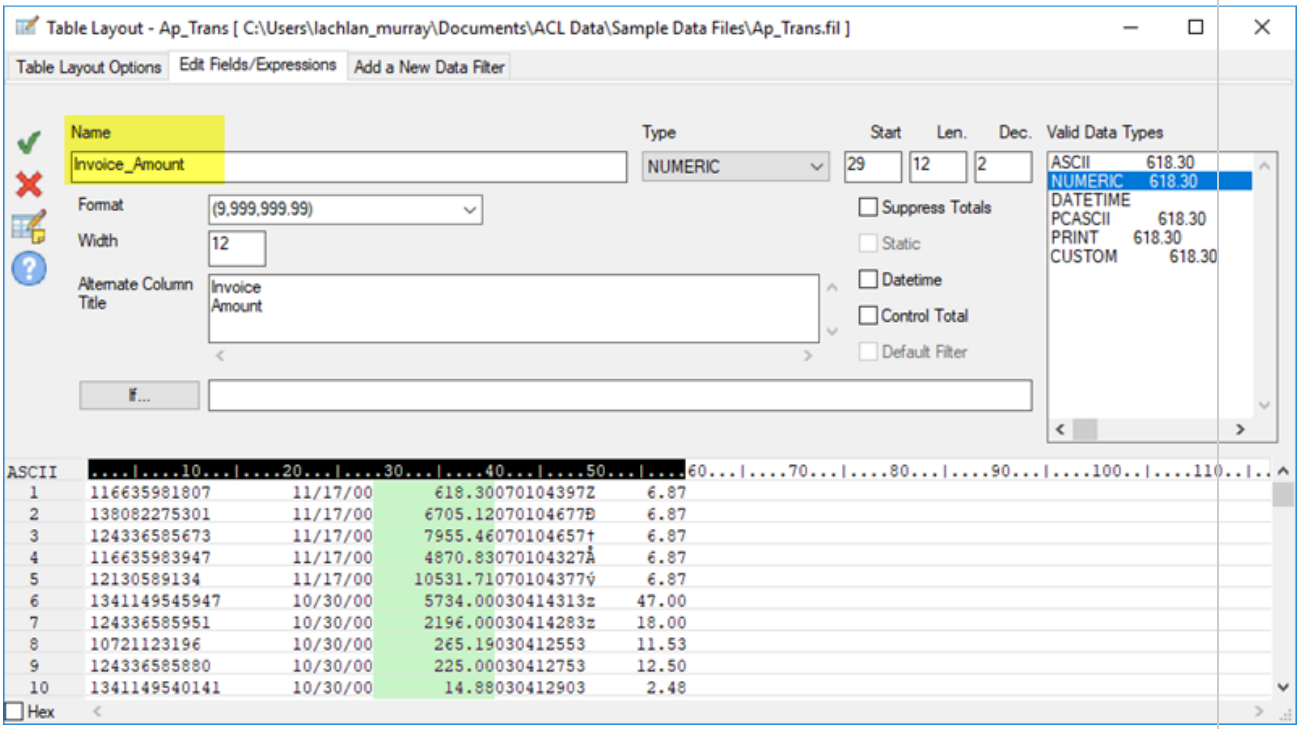
A physical field, also called **a field definition**, structures raw field data by specifying metadata information, including:

- the name of the field
- the start position of the field in the record
- the length of the field
- the data type of the field, which determines how Analytics reads and processes the data stored in the field

Additional information may also need to be specified based on the data type, and the settings you want to provide to override default values. For example, the formatting to apply to numeric fields, or the column title used in views and reports, can either be left blank and a default value is assigned, or you can specify the value to use.

Example of a physical field definition

The example below shows the definition for the **Invoice_Amount** field in the **Table Layout** dialog box. In the data preview area, the actual physical data included in the field is highlighted green.



Metadata element	Description	Value
Name	physical field name	Invoice_Amount
Type	data type	Numeric
Start	field start position	byte position 29
Len.	field length	12 bytes
Dec.	decimal places	2
Valid Data Types	Clickable list of suggested data types	Numeric includes preview of first value in the field
Format	numeric format	(9,999,999.99) ◦ numbers 0 to 9 supported ◦ thousands separator is a comma ◦ decimal separator is a period ◦ negative numbers are indicated by parentheses
Width	field display width in views and	12 characters

Metadata element	Description	Value
	reports	
Alternate Column Title	field display name in views and reports	Invoice Amount (two lines)

Define a physical field

You need to define a physical field for each field in a data source that you want to add to an Analytics table layout.

In most cases, the required physical fields are defined for you when you define and import data using the Data Definition Wizard or the Data Access window. However, you can define additional fields manually or you can choose to define all fields in a table layout manually.

Show me how

Specify the start position and length of the field

1. Select **Edit > Table Layout**.
2. In the **Edit Fields/Expressions** tab, click **Add a New Data Field**  .
3. Do one of the following to specify the field start position and length:
   - **Click and drag** – In the data preview area, click and drag in any of the data rows in the grid to highlight the field.
   - **Manually specify** – In the **Start** and **Len** text boxes, manually specify the field start position and length in bytes.

     If you manually specify the **Start** and **Len** values, follow these guidelines:

non-Unicode Analytics	1 byte = 1 character
Unicode Analytics, extended ASCII (ANSI) data	1 byte = 1 character
Unicode Analytics, Unicode data	2 bytes = 1 character
For Unicode data: ◦ **Start** - typically you should specify an odd-numbered starting byte position. Specifying an even-numbered starting position can cause characters to display incorrectly. ◦ **Len** - specify an even number of bytes only. Specifying an odd number of bytes can cause characters to display incorrectly.	

Specify field metadata

1. Enter the name for the field in the **Name** text box.

   > **Note**
   >
   > Field names are limited to 256 upper and lowercase alphanumeric characters. The name can include the underscore character (_), but no other special characters, or any spaces. The name cannot start with a number.
   >
   > Analytics has a number of reserved keywords that cannot be used as field names. For a complete list, see "Reserved keywords" on page 1364.

2. Select or confirm the appropriate data type in the **Type** drop-down list.

   The type you specify must match the data type in the source data, or must be appropriate for how you are using the data. For example, a field may be numeric data in the data source but you might want to define the field in Analytics as character data.

   Under **Valid Data Types**, a clickable list displays the data types that match the physical data you have specified. The most likely matches are listed first, with common types being listed before system or application-specific types.

3. (Optional) Specify the display width for the field in characters in the **Width** text box.

   The **Width** value is used as the column size when displaying the field in Analytics views and reports.

4. (Optional) Specify the display name in the **Alternate Column Title** text box.

   The display name is used as the column heading, instead of the field name, when displaying the field in Analytics views and reports. If a value is not specified, the field name is used.

5. (Optional) If you want to limit the records evaluated by the computed field, enter a condition in the **If** text box, or click **If** to create an IF statement using the **Expression Builder**.

   Records excluded by the IF statement are not evaluated by the computed field. For example, the IF statement `Invoice_Amount >= 1000` prevents records with invoice amounts less than $1000 from being evaluated.

   For excluded records, the computed field values are blank, 0.00, or False (F), depending on the data category of the computed field.

6. Depending on the data type you select, you may need to specify values for the following settings:

Setting	Description
Dec	Specifies the number of decimal places. This option is enabled only for numeric fields.
Format	Controls the display format of numeric fields in views and reports. It also specifies the input format of datetime fields in source data. The drop-down list is disabled when data types other than numeric or datetime are selected. You can select the format from the drop-down list, type in the format manually, or edit a format from the list after you have selected it.

Setting	Description
	If the **Format** drop-down list is blank, the default display format specified in the **Options** dialog box applies to the data in this field. The format you specify here overrides the default format.
Suppress Totals	Prevents the values in this field from being totaled. Analytics automatically totals numeric fields in reports. Some numeric fields contain information that should not be totaled, such as unit prices or account numbers. This option is enabled only for Numeric data types.
Static	Alters the default behavior Analytics uses when evaluating an IF statement associated with the field. (For more information about the optional IF statement, see "Finalize the field definition" below.) **Static deselected (default)** – if the IF statement evaluates to False, the field is assigned an empty value – blank, zero (0), or False (F), depending on the data category of the field. **Static selected** – if the IF statement evaluates to False, Analytics repeats the last valid value in the field rather than using an empty value. The last valid value is repeated in each row until the IF statement evaluates to True and a new valid value is used.
Datetime	Specifies that a numeric field should be interpreted as a datetime field. If the **Datetime** checkbox is selected, you must also specify the datetime format to use in the **Format** drop-down list.
Control Total	Specifies that the field is a control total field. A control total is the sum of values in a numeric field, which can be used to check data integrity. When you extract or sort data to a new table, Analytics includes the input and output totals of a control total field in the table history. Input refers to the original table. Output refers to the new table. If the two totals match, no data was lost in the extract or sort operation. You can also compare the control totals computed by Analytics with those supplied by a data provider to determine whether you received all the data. If you specify control totals for more than one field, the table history reports on only the numeric field with the leftmost starting position.
Default Filter	Filters the records in the default view based on the value of this field each time the Analytics table is opened. Only records that evaluate to true are displayed, and the filter is applied automatically. This option is enabled only for the Logical data type, and only one default filter can be specified for each table layout.

Finalize the field definition

1. (Optional) If you want to limit the values included in the field, enter a condition in the **If** text box, or click **If** to create an IF statement using the **Expression Builder**.
   - **included in the field** – values that satisfy the IF statement
   - **excluded from the field** – values that do not satisfy the IF statement

For example, the IF statement `Invoice_Amount >= 1000` includes invoice amounts of $1000 or greater, and excludes invoice amounts less than $1000.

Excluded values are not displayed in the field, or included in command processing. Depending on the data category of the field, excluded values appear as blank, zero (0), or False (F). You can undo the exclusion at any point by deleting the IF statement.

2. (Optional) Deselect **Add created field to current view** if you do not want the newly defined field to be automatically added to the open table view.

   If you leave the option selected, the new field is added to the table view. The field is positioned as the last column in view, or to the left of any selected column in the view.

   You can manually add a field to a view at any time. For more information, see "Add columns to a view" on page 778.

3. (Optional) If you want to add a note to accompany the field definition, click **Edit Field Note**  , enter the note text, and click **Close**  .
4. Click **Accept Entry**  .

   Analytics adds the field definition to the table layout.

5. Click **Close**  to exit the **Table Layout** dialog box.

   The associated column is added to the table view if you left **Add created field to current view** selected.

Defining datetime fields

Depending on the data source you are working with, datetime information (dates, datetimes, or times) may be stored as character data or numeric data. When you manually define a field that contains datetime information, Analytics treats it as character data by default. To ensure Analytics reads datetime information correctly, you need to select **Datetime** as the data type, and specify the datetime source format in the **Format** drop-down list.

Datetime source format

The datetime source format identifies the characters or digits **in the source data** that represent year, month, day, hour, minutes, and seconds, and any characters used to separate these parts of datetime data.

To match the way datetimes are stored in the source data, you can:

- select an existing datetime format
- specify your own datetime format
- select an existing format and modify it

For example, if December 31, 2014 is stored in the data source as 14-31-12, enter `YY-DD-MM` as the datetime format so that Analytics can interpret the date values correctly.

For more information, see "Formats of date and time source data" on page 347.

Datetime display format

The datetime source format you select or specify does not affect how datetime values are displayed in Analytics views or formatted in reports. The datetime display format depends on the **Date Display Format** and **Time Display Format** settings specified in the **Date and Time** tab in the **Options** dialog box.

For more information, see "Date and Time options" on page 133.

Defining overlapping fields

In most cases, when you define the physical fields in a record, each byte position in the record is assigned to only one field. At its most basic, defining a table is a matter of defining the start position and length of each field in the record, and one field starts after the previous field ends.

In some cases, however, you may need to define fields that overlap with each other, and some byte positions are used in more that one field. This situation might occur if the structure of the source data is non-standard, or if you want to work with the data in Analytics in a certain way.

For example, you could define the first six positions in a data source as a datetime field with the format DDMMYY, and then separately define a two-byte numeric field in position 3 and 4 for the month. This approach would allow you to access the entire date in one field for aging purposes, and have the month as a separate value in another field for generating monthly totals.

Computed fields

In an Analytics table, **a computed field** is a field that displays the results of an expression, rather than actual physical data. Computed fields typically perform some kind of calculation, operation, or data conversion.

For more information about expressions, see "Using expressions" on page 795.

How are computed fields useful?

The physical data you work with provides the basis for analysis, but frequently you need to extrapolate information from the physical data, or perform calculations, to move your analysis forward.

Without altering the physical source data, computed fields allow you to extrapolate and calculate. They are "virtual fields" that you can use to create useful data that does not directly exist in the physical data sources you are working with.

Some uses for computed fields

Display the result of a calculation	In an inventory file, you create a computed field called **Value** that multiples the **Quantity** field by the **Unit_cost** field to calculate the total value of each inventory item.
Convert a physical data field to a required data type	In order to work with a numeric field as if it were character data, you create a computed field that uses the STRING( ) function to convert the numeric values to character values.
Using conditions, substitute text values for numeric codes	You create a conditional computed field that displays the actual names of countries by mapping them to the numeric country codes in a physical field. For example: “Canada” instead of 01, and “USA” instead of 02.
Evaluate one or more conditions and determine the value of the field based on the result	You create a conditional computed field that calculates the tax on an item based on the region where it is sold. If the item is sold in one region the tax is calculated at 7%. If it is sold in another region the tax is calculated at 6%.

Data category of computed fields

Just like physical fields, computed fields belong to one of the following data categories:

- character
- numeric
- datetime
- logical

Unlike physical fields, you do not explicitly select a data type, and by extension a data category, when defining a computed field. Instead, the **Default Value** you specify for a computed field dictates the computed field data category.

Examples

The table below provides examples of computed field default values and the associated data category.

Computed field default value	Computed field data category
`"Location Unknown"`	Character
`STRING(Employee_number, 10)`	
`0.00`	Numeric
`Quantity * Unit_cost`	
`VALUE(Salary, 0)`	
`` `20180331` ``	Datetime
`CTOD(Invoice_date, "DD/MM/YYYY")`	
`T`	Logical
`Value > 1000`	

Controlling decimal precision in numeric computed fields

In a numeric computed field, the decimal precision of all numeric computed values is controlled by the precision of the expression or the literal value specified in the **Default Value** field.

- **expression** – if you specify the default expression `Invoice_Amount * 0.375`, and values in the Invoice Amount field have two decimal places, all computed values are calculated to three decimal places, and rounded if necessary.

The decimal precision of expressions is governed by the rules outlined in "Controlling rounding and decimal precision in numeric expressions" on page 803.

- **literal value** – if you specify the default value `0.00`, all computed values are calculated to two decimal places, and rounded if necessary.

Increase the decimal precision

To increase the decimal precision of numeric computed values, increase the number of decimal places in the **Default Value** field.

Expression

Multiply an expression by 1 followed by the number of decimal places of precision that you want. Make sure to position the 1 at the start of the expression. The example below increases the precision to four decimal places:

```
1.0000 * Invoice_Amount * 0.375
```

Literal value

Add trailing zeros to the decimal portion of a literal value. The example below increases the precision to three decimal places:

```
0.000
```

Types of computed fields

You can create two types of computed fields, which are described in subsequent sections:

- Basic computed field
- Conditional computed field:
  - with literal values
  - with computed values

Basic computed field

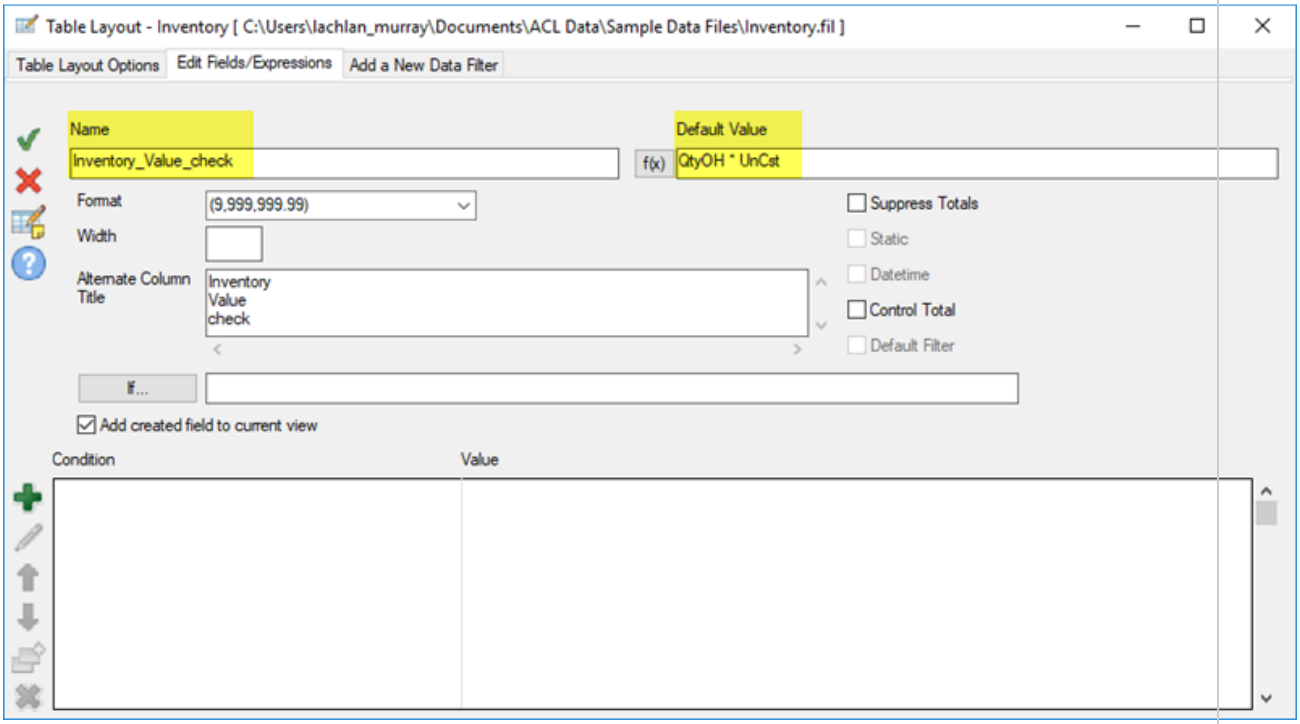
A basic computed field uses a single expression or literal value and applies it to all the records in a table, regardless of the value in each record.

Example of a basic computed field

You want to verify the total inventory value at cost for each product in an inventory report.

You create a computed field, **Inventory Value check**, that multiplies the **Quantity on Hand** field by the **Unit Cost** field. You can compare the values calculated by the computed field to the reported values to see if they match.

The example below shows the definition for the **Inventory_Value_check** computed field in the **Table Layout** dialog box. The computed expression (`QtyOH * UnCst`) appears in the **Default Value** field.



In the table view, you can position the computed field (**Inventory Value check**) beside the physical, source data field (**Inventory Value at Cost**), and compare values.

Quantity On Hand	Unit Cost	Inventory Value at Cost	Inventory Value check
870	6.87	5,976.90	5,976.90
460	6.87	3,160.20	3,160.20
1,480	(6.87)	(10,167.60)	(10,167.60)
1,290	6.87	8,862.30	8,862.30
1,500	6.87	10,305.00	10,305.00
2,420	6.87	16,625.40	16,625.40
1,870	6.87	12,846.90	12,846.90
130	47.00	6,110.00	6,110.00
612	18.00	11,016.00	11,016.00
700	11.53	8,071.00	8,071.00
248	12.50	3,100.00	3,100.00
248	2.48	615.04	615.04
0	8.40	0.00	0.00
(12)	49.60	(595.20)	(595.20)

You can also create a filter that returns any non-matching values:

```
Inventory_Value_check <> Inventory_Value_at_Cost
```

Define a basic computed field

Define a computed field that uses a single expression or literal value and applies it to all the records in a table, regardless of the value in each record.

Show me how

Specify the name and the default value of the computed field

1. Select **Edit > Table Layout**.
2. In the **Edit Fields/Expressions** tab, click **Add a New Expression**  .
3. Enter the name for the field in the **Name** text box.

> **Note**
>
> Field names are limited to 256 upper and lowercase alphanumeric characters. The name can include the underscore character (_), but no other special characters, or any spaces. The name cannot start with a number.
>
> Analytics has a number of reserved keywords that cannot be used as field names. For a complete list, see "Reserved keywords" on page 1364.

4. Do one of the following:
   - Enter an expression or a literal value in the **Default Value** text box.

     This method is only suitable for simple expressions.
   - Click **f(x)** to create an expression using the **Expression Builder**.

     For more information, see "Creating expressions using the Expression Builder" on page 801.

> **Note**
>
> For numeric computed fields, the decimal precision of all numeric computed values is controlled by the precision of the expression or the literal value specified in **Default Value**.
>
> For more information, see "Controlling decimal precision in numeric computed fields" on page 723.

Specify field metadata

1. (Optional) Specify the display width for the field in characters in the **Width** text box.

   The **Width** value is used as the column size when displaying the field in Analytics views and reports.
2. (Optional) Specify the display name in the **Alternate Column Title** text box.

   The display name is used as the column heading, instead of the field name, when displaying the field in Analytics views and reports. If a value is not specified, the field name is used.
3. If required, specify values for one or more of the settings listed below.

   The data category of the expression or the literal value that you specified in the **Default Value** text box dictates which settings are enabled.

Setting	Data category	Description
Format	Numeric only	Controls the display format of numeric fields in views and reports. You can select the format from the drop-down list, type in the format manually, or edit a format from the list after you have selected it. If the **Format** drop-down list is blank, the default display format specified in the **Options** dialog box applies to the data in the field. The format you specify here overrides the default format.
Suppress Totals	Numeric only	Prevents the values in the field from being totaled. Analytics automatically totals numeric fields in reports. Some numeric fields contain information that should not be totaled, such as unit prices or account numbers.
Static		Alters the default behavior Analytics uses when evaluating an IF statement associated with the field. (For more information about the

Setting	Data category	Description
		optional IF statement, see "Finalize the field definition" below.) **Static deselected (default)** – if the IF statement evaluates to False, the field is assigned an empty value – blank, zero (0), or False (F), depending on the data category of the field. **Static selected** – if the IF statement evaluates to False, Analytics repeats the last valid value in the field rather than using an empty value. The last valid value is repeated in each row until the IF statement evaluates to True and a new valid value is used.
Datetime		This option is not available for computed fields.
Control Total	Numeric only	Specifies that the field is a control total field. A control total is the sum of values in a numeric field, which can be used to check data integrity. When you extract or sort data to a new table, Analytics includes the input and output totals of a control total field in the table history. Input refers to the original table. Output refers to the new table. If the two totals match, no data was lost in the extract or sort operation. You can also compare the control totals computed by Analytics with those supplied by a data provider to determine whether you received all the data. If you specify control totals for more than one field, the table history reports on only the numeric field with the leftmost starting position.
Default Filter	Logical only	Filters the records in the default view based on the value of the field (True or False). Only records that evaluate to True are displayed. The filter is applied automatically each time the Analytics table containing the field is opened.

Finalize the field definition

1. (Optional) If you want to limit the records evaluated by the computed field, enter a condition in the **If** text box, or click **If** to create an IF statement using the **Expression Builder**.
   - **evaluated by the computed field** – records that satisfy the IF statement
   - **not evaluated by the computed field** – records that do not satisfy the IF statement

   For example, the IF statement `Invoice_Amount >= 1000` prevents records with invoice amounts less than $1000 from being evaluated.

   For excluded records, the computed field values are blank, zero (0), or False (F), depending on the data category of the computed field. You can undo the exclusion at any point by deleting the IF statement.
2. (Optional) Deselect **Add created field to current view** if you do not want the new computed field to be automatically added to the open table view.

If you leave the option selected, the new field is added to the table view. The field is positioned as the last column in view, or to the left of any selected column in the view.

You can manually add a field to a view at any time. For more information, see "Add columns to a view" on page 778.

3. (Optional) If you want to add a note to accompany the field definition, click **Edit Field Note**  , enter the note text, and click **Close**  .
4. Click **Accept Entry**  .

   Analytics adds the computed field to the table layout. You can use the field in commands or reports.
5. Click **Close**  to exit the **Table Layout** dialog box.

   The associated column is added to the table view if you left **Add created field to current view** selected.

Conditional computed field

A conditional computed field contains multiple expressions or literal values and applies them to the records in a table on a conditional basis. The particular expression or literal value applied to each record depends on the value in the record.

Example of a conditional computed field with literal values

You want to assign a literal value of "Small", "Medium", or "Large" to each record depending on the size of the invoice amount.

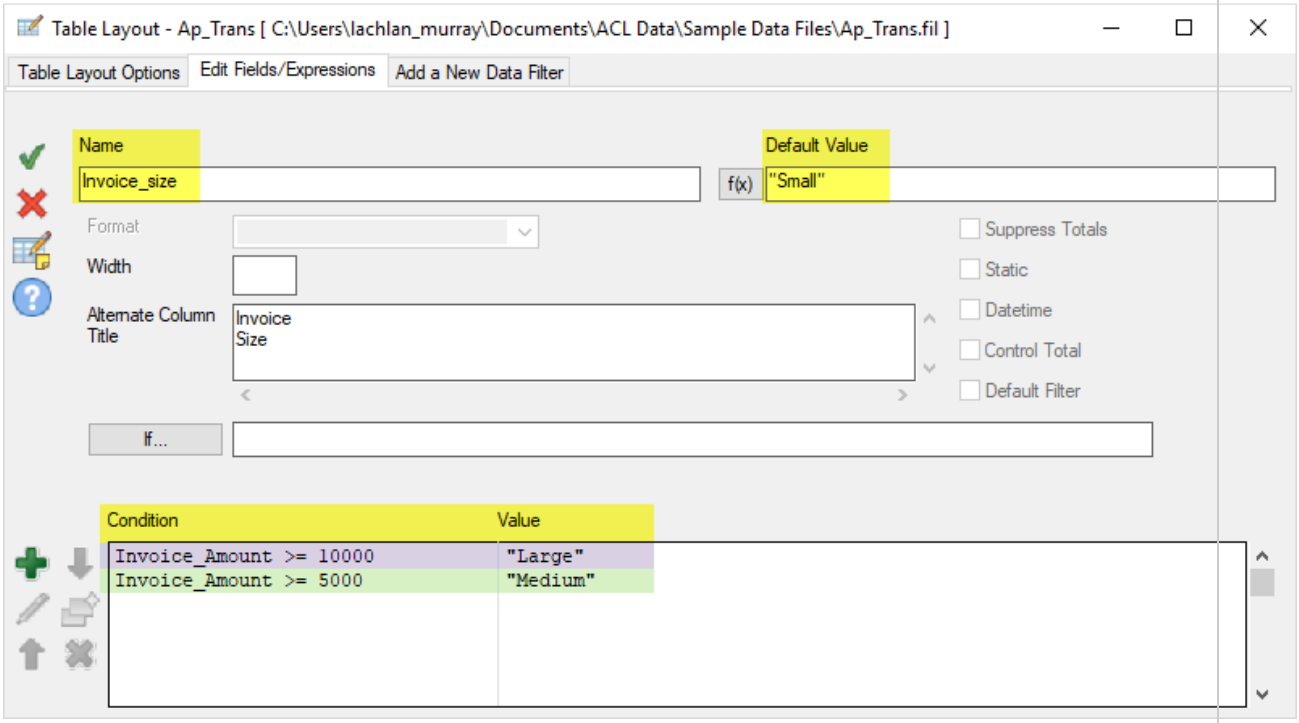
You create a computed field, **Invoice size**, that identifies the size of the invoice amount in each record, and assigns the correct literal value:

- **"Small"** – amounts less than $5,000.00
- **"Medium"** – amounts from $5,000.00 to $9,999.99
- **"Large"** – amounts $10,000.00 and greater

The example below shows the definition for the **Invoice size** computed field in the **Table Layout** dialog box. The literal value `"Small"` appears in the **Default Value** field. The literal values `"Medium"` and `"Large"` each appear in a separate condition.

Each condition contains a computed expression that must evaluate to True in order for the associated value to be used in the computed field. If a record satisfies neither of the conditions, the default value is used.

> **Note**
> The order in which conditions are listed is important. For more information, see "List conditions from most restrictive to least restrictive" on page 733.



In the table view, the invoice size now appears with each record.

Invoice Amount	Invoice Size
618.30	Small
6,705.12	Medium
7,955.46	Medium
4,870.83	Small
10,531.71	Large
5,734.00	Medium
2,196.00	Small
265.19	Small
225.00	Small
14.88	Small
1,217.16	Small
158.60	Small
2,230.41	Small
4,324.00	Small

You can also create a filter that displays only the records of one size:

```
Invoice_size = "Large"
```

Invoice Amount	Invoice Size
10,531.71	Large

Example of a conditional computed field with computed values

You want to calculate the discount amount for each record based on a discount percentage that varies with invoice size.

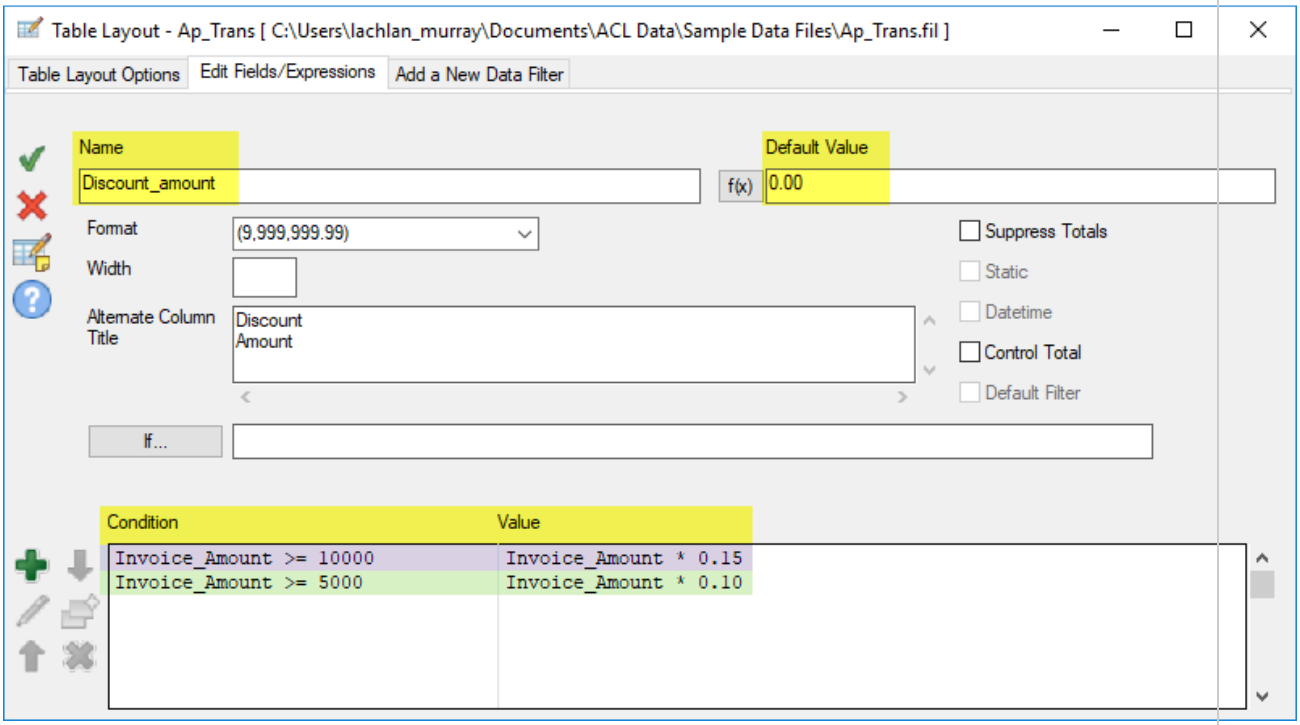
You create a computed field, **Discount amount**, that identifies the size of the invoice amount in each record, and computes the discount amount using the correct percentage:

- **0% discount** – amounts less than $5,000.00
- **10% discount** – amounts from $5,000.00 to $9,999.99
- **15% discount** – amounts $10,000.00 and greater

The example below shows the definition for the **Discount amount** computed field in the **Table Layout** dialog box. The literal value `0.00` appears in the **Default Value** field. The computed values `Invoice_Amount * 0.10` and `Invoice_Amount * 0.15` each appear in a separate condition.

Each condition contains a computed expression that must evaluate to True in order for the associated computed value to be used. If a record satisfies neither of the conditions, the default value is used.

> **Note**
>
> The order in which conditions are listed is important. For more information, see "List conditions from most restrictive to least restrictive" on page 733.



In the table view, the discount amount now appears with each record.

Invoice Amount	Discount Amount
618.30	0.00
6,705.12	670.51
7,955.46	795.55
4,870.83	0.00
10,531.71	1,579.76
5,734.00	573.40
2,196.00	0.00
265.19	0.00
225.00	0.00
14.88	0.00
1,217.16	0.00
158.60	0.00
2,230.41	0.00
4,324.00	0.00

You can also create a filter that displays discounts greater than a certain amount:

```
Discount_amount >= 750
```

Invoice Amount	Discount Amount
7,955.46	795.55
10,531.71	1,579.76

List conditions from most restrictive to least restrictive

When you define multiple conditions, Analytics evaluates them in the order that they are displayed in the condition list in the **Table Layout** dialog box, starting at the top.

In the **Invoice size** example above, invoice amounts are tested against the conditions in this order:

Order	Condition	Value
1	Invoice_Amount >= 10000	"Large"
2	Invoice_Amount >= 5000	"Medium"

To ensure that records that meet more than one condition are processed in the way that you intend, list conditions from most restrictive to least restrictive, with the most restrictive condition at the top.

Show me more

What does "restrictive" mean?

"Restrictive" refers to the proportion of a set of values that is eligible to meet a condition. The more restrictive a condition, the smaller the eligible proportion.

Consider the following set of values:

- $12,000
- $8,000
- $7,000

Least restrictive condition – All three values meet the condition `Invoice_Amount >= 5000`. The condition is the least restrictive because the entire set of values is eligible.

Most restrictive condition – Only $12,000 meets the condition `Invoice_Amount >= 10000`. The condition is the most restrictive because only one of the values in the set is eligible.

How Analytics assigns conditional computed field values

For each record, Analytics assigns the computed field value associated with the first condition that evaluates to True. Once assigned, a computed field value is not changed, even if a record satisfies a subsequent condition.

Consider an invoice amount of $12,000:

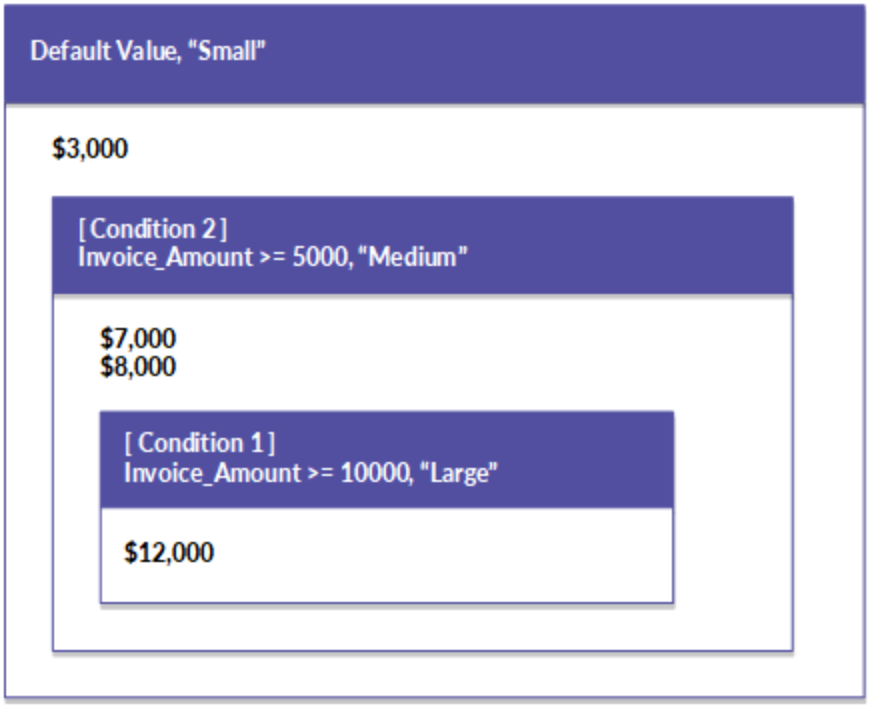
- **Assigned value = Large** – With the order of conditions above, the amount is assigned the value of Large, which is correct because the amount is greater than $10,000.

- **Assigned value = Medium** – If you reverse the order of the conditions, the amount is assigned the value of Medium, which is also correct because the amount is greater than $5,000. However, the assignment of values is not working the way you intended because `Invoice_Amount >= 5000` is a less restrictive condition and it is capturing amounts that you do not want it to capture.

Think of restriction in terms of subsets

Another way to think of restriction is in terms of subsets. Values eligible to meet the first listed condition should form the smallest subset of a set of values. With each additional condition, the size of the eligible subset grows, and contains all previous subsets.

Keep in mind that once Analytics assigns a computed field value to a record, the value is not changed. So in the example below, "Large" is assigned to the record containing an invoice amount of $12,000, and even though the record satisfies Condition 2, the value is not updated to "Medium".



Define a conditional computed field

Define a computed field that contains multiple expressions or literal values and applies them to the records in a table on a conditional basis.

Show me how

Specify the name and the default value of the computed field

1. Select **Edit > Table Layout**.
2. In the **Edit Fields/Expressions** tab, click **Add a New Expression**  .
3. Enter the name for the field in the **Name** text box.

> **Note**
>
> Field names are limited to 256 upper and lowercase alphanumeric characters. The name can include the underscore character (_), but no other special characters, or any spaces. The name cannot start with a number.
>
> Analytics has a number of reserved keywords that cannot be used as field names. For a complete list, see "Reserved keywords" on page 1364.

4. Do one of the following:
   - Enter an expression or a literal value in the **Default Value** text box.

     This method is only suitable for simple expressions.
   - Click **f(x)** to create an expression using the **Expression Builder**.

     For more information, see "Creating expressions using the Expression Builder" on page 801.

> **Note**
>
> For numeric computed fields, the decimal precision of all numeric computed values is controlled by the precision of the expression or the literal value specified in **Default Value**.
>
> For more information, see "Controlling decimal precision in numeric computed fields" on page 723.
>
> Literal text values must be enclosed in quotation marks (" "). Literal date values must be enclosed in backquotes (` `).

Specify field metadata

1. (Optional) Specify the display width for the field in characters in the **Width** text box.

   The **Width** value is used as the column size when displaying the field in Analytics views and reports.
2. (Optional) Specify the display name in the **Alternate Column Title** text box.

   The display name is used as the column heading, instead of the field name, when displaying the field in Analytics views and reports. If a value is not specified, the field name is used.

3. If required, specify values for one or more of the settings listed below.

   The data category of the expression or the literal value that you specified in the **Default Value** text box dictates which settings are enabled.

Setting	Data category	Description
Format	Numeric only	Controls the display format of numeric fields in views and reports. You can select the format from the drop-down list, type in the format manually, or edit a format from the list after you have selected it. If the **Format** drop-down list is blank, the default display format specified in the **Options** dialog box applies to the data in the field. The format you specify here overrides the default format.
Suppress Totals	Numeric only	Prevents the values in the field from being totaled. Analytics automatically totals numeric fields in reports. Some numeric fields contain information that should not be totaled, such as unit prices or account numbers.
Static		Alters the default behavior Analytics uses when evaluating an IF statement associated with the field. (For more information about the optional IF statement, see "Finalize the field definition" on page 738.) **Static deselected (default)** – if the IF statement evaluates to False, the field is assigned an empty value – blank, zero (0), or False (F), depending on the data category of the field. **Static selected** – if the IF statement evaluates to False, Analytics repeats the last valid value in the field rather than using an empty value. The last valid value is repeated in each row until the IF statement evaluates to True and a new valid value is used.
Datetime		This option is not available for computed fields.
Control Total	Numeric only	Specifies that the field is a control total field. A control total is the sum of values in a numeric field, which can be used to check data integrity. When you extract or sort data to a new table, Analytics includes the input and output totals of a control total field in the table history. Input refers to the original table. Output refers to the new table. If the two totals match, no data was lost in the extract or sort operation. You can also compare the control totals computed by Analytics with those supplied by a data provider to determine whether you received all the data. If you specify control totals for more than one field, the table history reports on only the numeric field with the leftmost starting position.
Default Filter	Logical only	Filters the records in the default view based on the value of the field (True or False). Only records that evaluate to True are displayed. The filter is applied automatically each time the Analytics table containing the field is opened.

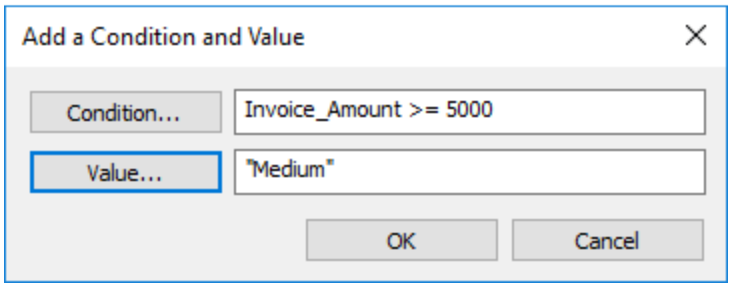
Specify conditional values of the computed field

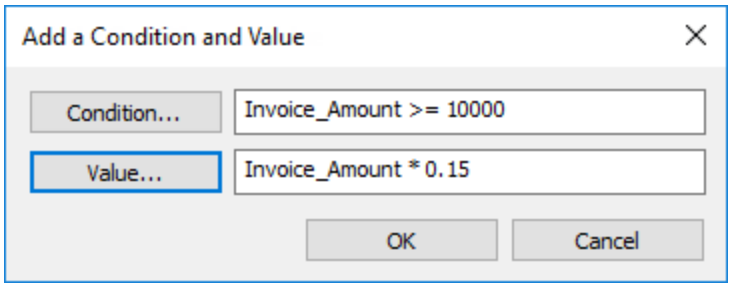
Conditional values are set up as condition-value pairs. If a record meets the **Condition**, the computed field uses the specified **Value**.

> **Note**
> The values you specify, and the **Default Value**, must all be the same data type.

1. Click **Insert a Condition**.
2. In the **Add a Condition and Value** dialog box, do the following, then click **OK**:
   a. Enter an expression in the **Condition** text box, or click **Condition** to create an expression using the **Expression Builder**.
   b. Enter a value in the **Value** text box, or click **Value** to create an expression using the **Expression Builder**.

   Here are conditional values from the two examples above:





3. If you want to create another condition, do one of the following:
   - Click **Insert a Condition** and repeat the steps above.
   - If you want to create a condition that is similar to an existing condition, select the condition you want to copy, click **Duplicate Condition**, and then click **Edit Condition and Value** to modify the settings for the new condition.
4. (Optional) Select a condition and click **Move Condition Up** or **Move Condition Down** to change the order in which it is evaluated.

> **Note**
> The order in which conditions are listed is important. For more information, see "List conditions from most restrictive to least restrictive" on page 733.

Finalize the field definition

1. (Optional) If you want to limit the records evaluated by the computed field, enter a condition in the **If** text box, or click **If** to create an IF statement using the **Expression Builder**.
   - **evaluated by the computed field** – records that satisfy the IF statement
   - **not evaluated by the computed field** – records that do not satisfy the IF statement

   For example, the IF statement `Invoice_Amount >= 1000` prevents records with invoice amounts less than $1000 from being evaluated.

   For excluded records, the computed field values are blank, zero (0), or False (F), depending on the data category of the computed field. You can undo the exclusion at any point by deleting the IF statement.
2. (Optional) Deselect **Add created field to current view** if you do not want the new computed field to be automatically added to the open table view.

   If you leave the option selected, the new field is added to the table view. The field is positioned as the last column in view, or to the left of any selected column in the view.

   You can manually add a field to a view at any time. For more information, see "Add columns to a view" on page 778.
3. (Optional) If you want to add a note to accompany the field definition, click **Edit Field Note**  , enter the note text, and click **Close**  .
4. Click **Accept Entry**  .

   Analytics adds the computed field to the table layout. You can use the field in commands or reports.
5. Click **Close**  to exit the **Table Layout** dialog box.

   The associated column is added to the table view if you left **Add created field to current view** selected.

Data types in Analytics

The data types supported by Analytics are listed below, including the data sources that the data types can be used with.

The same data type may have a different name in the **Data Definition Wizard** and the **Table Layout** dialog box. Both names are shown below.

Analytics data types are grouped into four data categories:

- Character
- Numeric
- Datetime
- Logical

The Analytics operations that you can perform on a field with a specific data type, and how the field is displayed, are determined by the data category. For example:

- You can only stratify fields with data types in the Numeric category.
- Fields with data types in the Character category are left-aligned.

If you use a field of the wrong data type in an operation, Analytics displays an error.

Analytics data type (Data Definition Wizard)	Analytics data type (Table Layout dialog box)	Analytics data category	External data source	Additional information
AccPac Accounting Number	ACCPAC	Numeric	ACCPAC	Used in ACCPAC accounting applications. The length of this data field is always 6 bytes. Analytics overrides any other specified length.
ACL	ACL	Numeric	None. This is an Analytics system data type.	An Analytics-generated 12 byte field that stores the results of Analytics computations. It is designed to store large numbers and is not a printable field. Analytics will automatically assign this type to a field when it is appropriate.
ASCII text	ASCII	Character	Windows-based applications	Used for data stored in the ASCII character encoding (American Standard Code for Information Interchange).

Analytics data type (Data Definition Wizard)	Analytics data type (Table Layout dialog box)	Analytics data category	External data source	Additional information
				Analytics uses extended ASCII which defines 256 printable and non-printable characters. The actual characters available in Analytics are specified by the operating system's default 8-bit code page. The maximum length of an ASCII field is 32767 bytes.
Basic Floating Point	BASIC	Numeric	Windows-based BASIC applications	Used for floating point data types formatted for the BASIC programming language. The field length of this data type can be either 4 or 8 bytes.
Binary Numeric	BINARY	Numeric	◦ PL/1 ◦ COBOL COMPUTATIONAL-1 ◦ Fixed binary data type	The maximum length is 8 bytes. The number of decimals is implied and cannot exceed the number of digits specified by the length. Binary fields of even length are treated as signed binary fields (twos compliment), and odd lengths are treated as unsigned fields (implicit high-order zero bytes are added).
Custom Text Format	CUSTOM	Character	None. This is an Analytics data type that can be assigned by the user as needed.	Used to enable user-defined character substitutions when data is read from the data source. This data type reads data as ASCII text unless there is a substitution character defined in a file named `custom.dat`. For more information, see "Custom data type" on page 748.
Datetime	DATETIME	Datetime	This is an Analytics data type automatically or	Used for date, datetime,

Analytics data type (Data Definition Wizard)	Analytics data type (Table Layout dialog box)	Analytics data category	External data source	Additional information
			manually assigned to fields that store dates, datetimes, and times.	and time data stored using a variety of different formats, such as YYMMDD, or YYMMDD hh:mm:ss. The **Format** setting in the field definition specifies how to read datetime data from the data source.
EBCDIC text	EBCDIC	Character	IBM z/OS and OS/400 applications	Used for Extended Binary Coded Decimal Interchange Code (EBCDIC) data, which is an 8-bit character encoding, on IBM server operating systems. The length of this data type is a maximum of 32767 bytes.
Floating Point	FLOAT	Numeric	Windows-based applica-tions	Used for double precision floating-point numbers. The field length of this data type can be either 4 or 8 bytes.
n/a	HALFBYTE	Numeric	Unisys/Burroughs applications	Used for half-byte aligned packed data found in Unisys/Burroughs systems. Signed numbers must follow the Unisys/Bur-roughs convention. The start position and length of this data type must be specified in half bytes. The start position can be calculated as follows: (byte_position*2)-1 This data type can only be selected in the **Table Layout** dialog box.
IBM Floating Point	IBMFLOAT	Numeric	IBM z/OS and OS/400 applications	Used for IBM floating-point data, which is mainly found in mainframe scientific applications. The field length of this data type can be either 4 or 8 bytes long.
Logical	LOGICAL	Logical	This is an Analytics data	Used for single character

Analytics data type (Data Definition Wizard)	Analytics data type (Table Layout dialog box)	Analytics data category	External data source	Additional information
			type automatically or manually assigned to fields that store logical values.	fields that represent Boolean data (usually true or false). Analytics can interpret the following sets of values, where the first value evaluates to true and the second evaluates to false: 1/0, T/F, t/f, Y/N, y/n, non-blank/ASCII blank (Hex 20)
PC Binary	MICRO	Numeric	Windows-based applications	Used for unsigned binary numeric data representing integer or long data types. The maximum length is 8 bytes. The number of decimals is the implied number of decimal digits, and cannot exceed the number of digits implied by the length. Micro fields with even lengths are treated as signed binary fields, and fields with odd lengths are treated as unsigned fields.
n/a	NOTE	Character	None. This is an Analytics system data type.	Used by Analytics to store information about the record notes associated with an Analytics table. You cannot define fields using the Note data type.
Numeric (Unformatted)	NUMERIC	Numeric	Windows ASCII or Unicode printable numeric data, or z/OS or OS/400 EBCDIC data that uses the COBOL display data type	Used for printable numeric data that corresponds to the COBOL display type. This field type can include any punctuation, but most commonly includes leading or trailing blanks, an optional leading or trailing sign, embedded commas, and an explicit decimal point. This data type can contain a maximum of 22 digits plus 18 characters of punctuation, for a total length of 40 bytes, and

Analytics data type (Data Definition Wizard)	Analytics data type (Table Layout dialog box)	Analytics data category	External data source	Additional information
				leading zeros are treated as blanks. This data type should be used with caution because the number of decimal points specified for the field are applied whether appropriate or not. For example, if you specify 2 decimal places and the values $500.50 and $399 are read, the first value will be interpreted correctly as 500.50, but the second value will be interpreted as 3.99 instead of 399.00. If the specified decimal places differ from the explicit decimals in the field, the field is rounded to the appropriate number of decimals. Analytics correctly interprets parentheses and "CR" as negative, but ignores commas and other punctuation, such as dollar signs ($). Signs can be leading or trailing, fixed, or floating.
Packed Numeric	PACKED	Numeric	PL/1 fixed decimal data type or the COBOL computational-3 data type	Used for packed numeric data from mainframe operating systems that stores two numeric digits per byte. The rightmost byte contains a sign indication in the lower half of the byte, usually hexadecimal C for positive and hexadecimal D for negative. (Using hexadecimal B to indicate negative numbers is not supported.) The upper half of the rightmost byte and each half of all other bytes contain one hexadecimal digit that represents the numeric digit of that

Analytics data type (Data Definition Wizard)	Analytics data type (Table Layout dialog box)	Analytics data category	External data source	Additional information
				position in the number. The length of this data type is a maximum of 12 bytes (23 digits); however, Analytics generates an error message if it encounters a number larger than 22 digits. Consequently, when you define a packed numeric field in the **Table Layout** dialog box, the number of decimals that you specify in the **Dec** text box must not result in numbers longer than 22 digits. For example, if your data contains seven-digit figures, you cannot specify more than 15 decimal places (22 digits - 7 digits). Packed Numeric fields can also be used to store date information in numeric form.
PC DOS Text	PCASCII	Character	Windows	Similar to the ASCII data type. You can use it when the data in a file is created by a DOS application. The PCASCII characters available in Analytics are specified by code page 437. The maximum length of a PCASCII field is 32767 bytes.

Analytics data type (Data Definition Wizard)	Analytics data type (Table Layout dialog box)	Analytics data category	External data source	Additional information
				Note Do not use the PCASCII data type when the ASCII data type is required. The extended character sets of the two data types are different.
Numeric (Formatted)	PRINT	Numeric	Windows ASCII or Unicode printable numeric data, or z/OS or OS/400 EBCDIC data that uses the COBOL display data type	Used for printable numeric data that corresponds to the COBOL display type. This field type can include any punctuation, but most commonly includes leading or trailing blanks, an optional leading or trailing sign, embedded commas, and an explicit decimal point. This data type can contain a maximum of 22 digits plus 18 characters of punctuation, for a total length of 40 bytes, and leading zeros are treated as blanks. This data type should be used instead of the Numeric (Unformat-ted)/NUMERIC type when the decimal digits are not included for every numeric value. For example, if you specify 2 decimal places and the values $500.50 and $399 are read, this data type will correctly interpret both values (500.50 and 399.00). If the specified decimal places differ from the explicit decimals in the field, the field is rounded to the appropriate number of

Analytics data type (Data Definition Wizard)	Analytics data type (Table Layout dialog box)	Analytics data category	External data source	Additional information
				decimals. Analytics correctly interprets parentheses and "CR" as negative, but ignores commas and other punctuation, such as dollar signs ($). Signs can be leading or trailing, fixed, or floating.
Unicode	UNICODE	Character	Unicode data	Used for Unicode character data. For Unicode data, Analytics uses the UTF-16LE character encoding. This data type is only available in the Unicode edition of Analytics.
UNISYS Packed	UNISYS	Numeric	Unisys/Burroughs applications	Used to read Unisys/Burroughs byte-aligned packed data. Signed numbers must follow the Unisys/Burroughs convention, and unsigned Unisys packed data should use the Unsigned Packed/UNSIGNED data type. The maximum length of this field type is 12 bytes, or 22 digits.
Unsigned Packed	UNSIGNED	Numeric	IBM z/OS and OS/400 applications	Used for unsigned packed data, which is a data type that stores two decimal digits per byte. The length of this data type is a maximum of 11 bytes or 22 decimal digits. The number of decimal places cannot exceed the maximum number of digits possible for this field.
VAX Floating Point	VAXFLOAT	Numeric	DEC VAX applications	Used for type-D floating-point data from Digital Equipment Corporation's VAX systems. The length of

Analytics data type (Data Definition Wizard)	Analytics data type (Table Layout dialog box)	Analytics data category	External data source	Additional information
				this data type is either 4 or 8 bytes.
Zoned Numeric	ZONED	Numeric	IBM, DEC, or Honeywell mainframe applications	Used for zoned numeric fields that store one digit per byte, and can be encoded using ASCII, EBCDIC, or Unicode (if you are using the Unicode edition of Analytics). Leading zeros are retained, and the upper half of the rightmost byte of the field includes the minus sign. The maximum length of a zoned field is 22 bytes. Analytics automatically detects and adjusts for zoned fields conforming to the IBM, Honeywell, and DEC formats.

Custom data type

The Analytics Custom data type allows you to process data source fields that contain non-standard character data. For example, you can use the Custom data type to read data from foreign-language applications that implement certain characters in a non-standard or unsupported way.

The Custom data type stores ASCII values. However, you can create a file named `custom.dat` that maps non-standard or unsupported character values to standard ASCII character values.

Custom.dat file

`Custom.dat` is a standard text file with two values on each line. The first value is the non-standard or unsupported character to be replaced, and the second value is the ASCII character to replace it with. The values can be specified using any of the following methods, or combination of methods:

- Character codes are specified using numeric values, such as 65 for the character 'A'.
- Hexadecimal values are specified using the two-character hexadecimal value preceded by an X, such as X41 for the character 'A'.
- Literal character values are specified using the character preceded by a C, such as CA for the character 'A'.

The `custom.dat` file is read when you open Analytics, so you cannot edit the file while Analytics is running. None of the values specified in the `custom.dat` file should exceed 255, which is the largest value that can be stored in a byte. You can use any text editor to create or edit the `custom.dat` file.

Example

The data source field uses the hexadecimal values A4 for a dollar sign and A5 for a comma, and the character code 5 for a decimal point. You create a `custom.dat` file to substitute the required values. The file includes the following lines:

```
XA4 C$
XA5 C,
5 C.
```

- **first line** - substitutes the dollar sign ($) wherever the hexadecimal value A4 is encountered.
- **second line** - substitutes a comma wherever the hexadecimal value A5 is encountered.
- **third line** - substitutes a decimal point wherever the character code 5 is encountered.

Configure substitution rules for the Custom data type

Configure substitution rules for the Custom data type by creating a file called `custom.dat` that contains a list of characters that need to be replaced and the replacement characters.

Each time the Custom data type is selected for a field definition, the non-standard or unsupported characters listed in `custom.dat` are automatically replaced by the mapped ASCII equivalents. A single `custom.dat` file applies globally to all fields in Analytics projects that are defined using the Custom data type.

1. Open a text editor and create a new file.
2. Enter each substitution rule on a separate line using the following syntax:

   ```
   <type> character_to_replace <type> replacement_character
   ```

   - `type` - Specify `C` for character values or `X` for hexadecimal values. Do not specify a type for numeric ASCII character codes.
   - `character_to_replace` - Specify the character, digit, or hexadecimal value that you want to replace.
   - `replacement_character` - Specify the character, digit, or hexadecimal value that you want to substitute for the `character_to_replace` value.
3. Save the file as `custom.dat` in the folder where the Analytics executable (ACLWin.exe) is installed.

   The default location is `C:\Program Files (x86)\ACL Software\ACL for Windows <version>`.

   The Custom data type can be used the next time you open Analytics. Whenever you use the Custom data type, it automatically applies the substitution rules that you have defined.

Modifying fields in table layouts

You can modify the field definitions in a table layout if the data source they connect to changes, or if you want to change how a field is formatted.

Analytics restricts changes to field definitions if they are referenced by a computed field. In this case, you are prevented from changing the field name or data type, but you can change other properties, such as start position or length.

To modify a field in a table layout:

1. Select **Edit > Table Layout**.
2. In the **Edit Fields/Expressions** tab, double-click the field you want to modify.
3. Make the necessary changes to the field definition and click **Accept Entry**  .

   For more information about field definitions, see "Define a physical field" on page 717.

Rename a field in a table layout

You can rename one or more fields in a table layout, if required. Renaming a field changes the name of the data element that is acted upon by Analytics operations. It is not the same as renaming a column in a view, which is changing only a display name.

When you rename a field, the error message "<*previous field name*> is undefined" appears when you return to the view, if the view contained the field, or when you open any view that contains the field.

1. Select **Edit > Table Layout**.
2. In the **Edit Fields/Expressions** tab, double-click the field you want to rename.
3. Change the field name in the **Name** field.
4. (Optional) If you want to rename the column in the view at the same time, change the column name in the **Alternate Column Title** field.
5. Click **Accept Entry**  .

   The field is renamed in the table layout.
6. Click **Close**  to exit the **Table Layout** dialog box.
7. If the error message "<*previous field name*> is undefined" appears, click **OK**.

   If you updated the **Alternate Column Title**, the column name is updated.

> **Note**
>
> If the renamed field is in any other views, it is removed from those views and must be manually re-added.

Deleting fields from table layouts

To delete one or more fields:

1. Select **Edit > Table Layout**.
2. In the **Edit Fields/Expressions** tab, select the field you want to delete, or **Ctrl+click** to select multiple fields.
3. Click **Delete Fields**  .
4. Click **Delete** in the confirmation prompt.

Shifting fields in table layouts

Shifting fields allows you to correct field definitions that are displaced by a fixed number of bytes. This is only required if the table layout was incorrectly defined, or if changes have been made to the data source and you do not want to redefine your Analytics table.

If a data source has minor changes, such as an increased field length for one of the fields you have defined, you can shift the position of the subsequent fields and keep using the updated table layout. By shifting a field's start position in a table layout, you automatically shift the start positions of all of the fields defined to the right of that field. Shifting fields only affects physical data fields.

Adjusting record length

When you shift fields in a table layout you may also need to adjust the **Record Length** value in the **Table Layout Options** tab if any of the following situations apply:

- **Changed field length** – If the length of a field in the data source has changed, you will first need to manually adjust the length of the corresponding field in the table layout, and you may need to adjust the record length in the table layout before shifting fields.
- **Field added** – If a new field has been added in the data source, you may first need to increase the record length in the **Table Layout** dialog box to accommodate the new field. Analytics does not allow you to shift fields if one of the shifted fields will extend beyond the current record length. Once the fields have been shifted, you can add the new field definition.
- **Field removed** – If a field has been removed from the data source, you need to delete the corresponding field in the table layout. After shifting fields, you may need to manually decrease the record length.

Shifted field definitions must remain within the record length

When you shift one or more field definitions right or left, the fields cannot exceed the record length in either direction.

Shifting fields moves both the specified field definition, and any field definitions to the right of the specified definition. If the shifted block of definitions would exceed the record length in either direction, an error message appears and the shift operation is not performed.

> **Tip**
> If the error message is appearing because you are exceeding the end of the record, try removing the final field definition to make room for the field definitions being shifted.

Shift fields in a table layout

1. Select **Edit > Table Layout**.
2. Click the **Edit Fields/Expressions** tab.
3. Optional. If a data filter is defined for the table layout, you can choose to shift fields for a particular data filter only. Select the appropriate data filter from the drop-down list above the list of fields. Analytics displays only fields associated with the selected data filter, and these are the only fields that are shifted.
4. Optional. If you want to start shifting fields from a particular field, select the field in the list of fields.

   The starting byte position of the field you select is prefilled in the **Shift Fields** dialog box.
5. Click **Shift Fields**.
6. In the **Start Shifting Fields from position** text box, keep the prefilled value, or specify the starting byte position of the first field definition you want to shift.

   All field definitions to the right of the specified field definition are also shifted.

   If you specify a non-starting byte position, the next starting byte position is used.

   > **Note**
   >
non-Unicode Analytics	1 byte = 1 character
Unicode Analytics, extended ASCII (ANSI) data	1 byte = 1 character
Unicode Analytics, Unicode data	2 bytes = 1 character
   >
   > For Unicode data, typically you should specify an odd-numbered starting byte position. Specifying an even-numbered starting position can cause characters to display incorrectly.
7. In the **Shift fields by the following number of bytes** text box, enter the number of bytes to shift the field definition.

   Enter a positive number to shift a field definition to the right. Enter a negative number to shift a field definition to the left.

Note

non-Unicode Analytics	1 byte = 1 character
Unicode Analytics, extended ASCII (ANSI) data	1 byte = 1 character
Unicode Analytics, Unicode data	2 bytes = 1 character

For Unicode data, specify an even number of bytes only. Specifying an odd number of bytes can cause characters to display incorrectly.

8. Click **OK**, and then click **Yes** in the confirmation dialog box.

Dumping data

The **Dump** feature allows you to view all the printable and non-printable characters in a record or a file in one or more of the following encodings:

- Hex
- ASCII
- EBCDIC
- Unicode (Unicode Analytics only)

You can use the Dump feature to troubleshoot problems with viewing or processing data, or to identify the data fields in a file.

1. If you want to dump data for a particular record in an Analytics table, open the view and select the record.
2. Select **Tools > Hex Dump**.
3. If you want to dump data from a file, select **File** and then select the file in the **Open** dialog box and click **Open**.

   You can select an Analytics source data file (.fil), or another file type.
4. Optional. In **Skip Bytes**, enter a value greater than zero to skip the specified number of bytes from the start of the file before dumping data.
5. Optional. In **Columns**, specify the width of the output columns in bytes.

   > **Note**
   >
   > The value you specify refers to the bytes contained by the Analytics record or table.
   >
   > The encoded characters in the output may not have a one-to-one relation with the characters in the record or table. For example, the hexadecimal encoding for the number 1 is `31`.

   The default is 16 bytes for each column in a vertical display, and 64 bytes for the single column in a horizontal display. The maximum number of bytes you can specify is 255.
6. Optional. Select **Horizontal** to display the character encodings in horizontal rows rather than in side-by-side vertical blocks (the default).
7. Deselect any of the character encodings you do not want display: **HEX**, **ASCII**, **EBCDIC**, or **Unicode** (Unicode Analytics only).
8. If you want to locate a particular value, do the following:
   a. Click **Find**.
   b. Enter the search string in the **Find** text box.
   c. Select the character encoding to search: **ASCII**, **EBCDIC**, or **HEX**.
   d. Select **Ignore case** if you want the search to be case-insensitive.
   e. In the **Search from** panel, select **Top** to search from the top of the file, or **Cursor** to start the search at the current cursor position. The currently selected position is displayed in the **Position** field in the **Dump** dialog box.
   f. Click **Find**.

      If a match is found, the location of the match is highlighted in each character encoding.

9. Click **Close**  to exit the **Dump** dialog box.

Viewing table history

Analytics records information about Analytics tables that are created as output from Analytics commands. The history of a table includes such information as:

- The date and time the table was created
- The name of the original table and new output table
- The command used to create the table
- Control totals

To view history information for a table:

1. Open the table you want to view the history for.
2. Select **Tools > Table History**.
3. To print the history, right-click in the display area and select **Print**.

Using workspaces to share field definitions

An Analytics workspace is an Analytics project item that contains one or more field definitions that have been saved for reuse with other tables. When a workspace is activated, the fields within it are available to the current table. Workspaces let you maintain and reuse definitions of physical fields, computed fields, or filters (which can be selected from the **Filters** list in the **Expression Builder**). Reusing or sharing field definitions and filters ensures consistency, and reduces the repetitive work of defining fields and filters that are used in more than one table.

Workspaces can be shared between tables that contain fields of the same type, with the same names. For example, you might want to associate a workspace with successive tables of a given type, such as accounts receivable tables from different periods, or from different divisions of the same company.

If you are working with multiple-record-type files, you can store the definition for each record type in a separate workspace. You can select the appropriate workspace to process records of a specific type.

Create a workspace

1. If you want to add field definitions from a particular Analytics table to the workspace, you need to open the table before you create the new workspace.
2. Select **File > New > Workspace**.
3. In the **Add Fields to Workspace** dialog box, complete any of the following tasks:
   - Click **Add All** to add all fields to the workspace.
   - Click an individual field in the **Available Fields** list and then click the right-arrow button to add it to the workspace.
   - **Ctrl+click** multiple fields in the **Available Fields** list and then click the right-arrow button to add them to the workspace.
   - Click **Expr** to open the **Expression Builder** and create an expression to add to the workspace.

     > **Note**
     > If you are adding computed fields that reference other computed fields, you need to add the computed fields that do not have dependencies (do not reference computed fields) before you add the computed fields that have dependencies.

4. Click **OK**.

5. In the **Overview** tab, right-click the workspace file and select **Rename**.
6. Type a new name for the workspace and press Enter.

> **Note**
> Workspace names are limited to 64 alphanumeric characters. The name can include the underscore character (_), but no other special characters, or any spaces. The name cannot start with a number.

Edit a workspace

You can edit a workspace by adding additional field definitions, or by modifying or deleting existing field definitions.

1. If you want to add field definitions from a particular Analytics table to the workspace, you need to open the table before you start editing the workspace.
2. Right-click the workspace file in the **Overview** tab in the **Navigator** and select **Edit**.
3. Edit the entries in the **Workspace Editor**. You can modify or delete entries by editing the field definition text.
4. Complete the following steps to add fields to the workspace:
   a. Click **Add Fields to Workspace**  in the Workspace toolbar.
   b. In the **Add Fields to Workspace** dialog box, complete any of the following tasks:
      - Click **Add All** to add all fields to the workspace.
      - Click an individual field in the **Available Fields** list and then click the right-arrow button to add it to the workspace.
      - **Ctrl+click** multiple fields in the **Available Fields** list and then click the right-arrow button to add them to the workspace.
   c. Click **OK**.
5. Right-click the workspace file in the **Overview** tab in the **Navigator** and select **Close**.
6. Click **Yes** in the confirmation dialog box.

Activate a workspace

A workspace can be activated for use with any Analytics table, but you must ensure that any fields referenced in the workspace field definitions are available in the Analytics table. For example, if a workspace includes a computed field named `Value` that is defined using the expression `sale_price` * `quantity`, you must ensure that any table you use the workspace with includes both the `sale_price` and `quantity` fields.

If you activate a workspace that includes a field with the same name as one of the fields in the table, Analytics asks if you want to replace the field in the table. Click **Yes** to temporarily replace the field in the table with the workspace field until you close the table.

If you edit the table layout after you activate a workspace, or make a change that causes the application to automatically save the table layout, Analytics permanently adds the workspace fields to

the table layout. Once the workspace fields are saved in the table layout, you can add the fields to the view.

1. Open the table you want to use the workspace with.
2. In the **Overview** tab of the **Navigator**, right-click the workspace and select **Activate**.

Add a workspace field definition to a table layout

By default, the fields in a workspace are only available for use with your Analytics table when the table is open and the workspace is activated. When the table is closed, the workspace is deactivated automatically, and the workspace fields are no longer available for use unless the workspace is reactivated. If you want the workspace fields to always be available when the table is open, you need to add them to the table layout. When you add the workspace fields to the table layout, you are copying the definitions from the workspace and creating fields in the table layout. This means that any future changes to these fields must be completed in the table layout, not in the workspace.

1. Open the table you want to add the workspace field definitions to.
2. In the **Overview** tab of the **Navigator**, right-click the workspace and select **Activate**.
3. Select **Edit > Table Layout**.
4. Double-click one of the fields listed to edit the properties of the field.
5. Change the value in the **Width** text box by increasing the value by 1.
6. Click **Accept Entry**  .

   Saving the change forces Analytics to save the change made to the field definition and also saves the field definitions in the activated workspace in the table layout.
7. Double-click the field you modified in steps 4 and 5 and restore the original value in the **Width** text box.
8. Click **Accept Entry**  .
9. Click **Close**  to exit the **Table Layout** dialog box.

Copy a workspace from another Analytics project

You can copy a workspace from one Analytics project to another, which allows you to reuse the physical or computed field definitions, or the filters, the workspace contains rather than creating them from scratch. In addition to saving labor, reusing any of these items, or sharing them with other Analytics users, ensures consistency. You can copy a single workspace, or multiple workspaces simultaneously.

If you want to import a workspace that exists as a separate file outside an Analytics project, see "Import a workspace" on the next page.

1. Open the project that will contain the copied workspace or workspaces.
2. In the **Overview** tab of the **Navigator**, right-click the Analytics project entry, or a project folder, and select **Copy from another Project > Workspace**.

   The Analytics project is the top-level folder in the treeview.
3. In the **Locate Project File** dialog box, locate and select the Analytics project you want to copy the workspace or workspaces from and click **Open**.
4. In the **Import** dialog box, complete any of the following tasks to add one or more workspaces to the **To *project_name*** list:
   - Double-click a workspace.
   - **Ctrl+click** multiple workspaces and then click the right-arrow button.
   - Click **Add All** to add all the workspaces.

   You can remove workspaces from the **To *project_name*** list by double-clicking an individual workspace, by using **Ctrl+click** to select multiple workspaces and then clicking the left-arrow button, or by clicking **Clear All**.
5. Click **OK** to copy the workspace or workspaces into the destination project.

   If a workspace with the same name already exists in the project, the copied workspace is given an incrementing numeric suffix.

Import a workspace

You can import a workspace that exists as a separate .wsp file outside an Analytics project, which allows you to reuse the physical or computed field definitions, or the filters, the workspace contains rather than creating them from scratch. In addition to saving labor, reusing any of these items, or sharing them with other Analytics users, ensures consistency. You can import only one workspace at a time.

If you want to import a workspace from another Analytics project, see "Copy a workspace from another Analytics project" on the previous page.

1. In the **Overview** tab of the **Navigator**, right-click the Analytics project entry, or a project folder, and select **Import Project Item > Workspace**.

   The Analytics project is the top-level folder in the treeview.
2. In the **Project** dialog box, locate and select a workspace file (.wsp) and click **Open**.
3. Click **OK** in the confirmation dialog box.

   The workspace is imported into the project. If a workspace with the same name already exists in the project, the imported workspace is given an incrementing numeric suffix.

Export a workspace

You can export a workspace as a separate .wsp file saved outside the Analytics project. A workspace exported as a separate file can later be imported into any Analytics project, which allows you to reuse

the physical or computed field definitions, or the filters, the workspace contains rather than creating them from scratch. You can export only one workspace at a time.

1. Right-click the workspace in the **Overview** tab of the **Navigator** and select **Export Project Item**.
2. In the **Save As** dialog box, choose a location to save the workspace, rename the workspace if required, click **Save**, and click **OK** in the confirmation dialog box.

   The workspace is exported to the location you specified.

> **Note**
>
> Limit the workspace name to 64 alphanumeric characters, not including the .wsp extension, to ensure that the name is not truncated when the workspace is imported back into Analytics.
>
> The name can include the underscore character (_), but do not use any other special characters, or any spaces, or start the name with a number. Special characters, spaces, and a leading number are all replaced by the underscore character when the workspace is imported.

Add or edit a workspace note

You can add a note to a workspace to record any details about the workspace that you want to keep for future reference, or document for other users. You can edit the content of a workspace note at any time.

You do not need to activate the workspace to add, edit, delete, or read the note.

1. Right-click the workspace in the **Overview** tab in the **Navigator**.
2. Select **Properties**.
3. In the **Workspace Properties** dialog box, click the **Notes** tab.
4. Enter a new note or edit the existing note.

   To delete the note, delete all the text.

5. Click **OK** to close the dialog box and save your changes.

About data filters

When defining data sources that contain more than one record type, such as Print Image (Report) files and Multiple Record Type files, you need to be able to identify which type of record you are using at any time. In Analytics you use data filters to identify the distinct record types in a file and, when necessary, to exclude unwanted records.

Data filters are used to identify portions of a data file that can be defined as records and fields for an Analytics table. Data filters are defined using conditions that indicate which parts of a data file should be included in a record or excluded from the record.

When you start to create a new data filter in the **Add a New Data Filter** tab, an Exclude All condition is created by default which excludes all character data in the source data file and highlights all characters in black.

To define your record, you must specify at least one filter condition by selecting a character or set of characters that can be used to uniquely identify each instance of the record you want to include. When you click a data element to select it, Analytics identifies it as a filter condition and adds its description to the filter conditions text box. When you create a filter condition to include a record, the black highlighting is removed from all of the lines that match the filter criteria. If there are portions of the selected record that you do not want to include in the record, you can specify another filter condition to exclude a portion of the record.

You can define more than one data filter in the same data file. A common style of report lists a header record, which could include information about a product class, followed by a number of detail records that are associated with that product class.

In this example, the header record contains additional information associated with the detail records, such as the product class number, and the product class name. To combine this information with the detail records, you need to define the fields in the header record as static fields by selecting the **Static** checkbox for each field in the **Edit Fields/Expressions** tab. When you create a new data filter, you are prompted to indicate whether the information in the record type relates to subsequent records. If you answer **Yes**, the fields you define in the record will have the **Static** checkbox selected by default. You can also select the **Static** checkbox for each field manually.

Create a data filter

Data filters are used to identify the record, or records, in a data file. Creating a data filter involves identifying rules that define the area of the data file that should be included in your Analytics table, and exclude any data that should not be included. Data highlighted in black in the preview table is excluded, and data highlighted in white is included.

If there is an active data filter it is deactivated automatically when you create a new filter.

If you want to edit an existing data filter, you can edit it as a computed field in the **Table Layout** dialog box. To display data filters in the **Table Layout** dialog box, you must select the **Include Filters in Field Lists** option in the **Interface** tab in the **Options** dialog box (**Tools > Options**).

Steps

1. Select **Edit > Table Layout**.
2. Click the **Add a New Data Filter** tab.

   The contents of the data file are displayed in the data preview area in the bottom half of the screen. By default all data displayed is initially excluded. This is indicated by the "Exclude All" condition listed in the data filter list, which cannot be modified or deleted.
3. Select a unique character or sequence of characters in the record to define the filter condition. Click an individual character to highlight it, or click and drag to select more than one character, and click **Include** to select all of the records that match the specified condition.

   For example, a data file might have the decimal point in byte position 71 of each row that should be included in your Analytics table. You need to include this decimal point in your filter. If the decimal point is in the same position in any rows you do not want to include, you will need to create a rule to omit the incorrectly selected rows.
4. If you want to exclude part of a selected record, select a unique character or sequence of characters in the record to define the filter condition and click **Exclude**.
5. Click **Accept Entry**  to create the data filter with the specified conditions.
6. In the **Save Filter As** dialog box, enter a name for the filter and click **OK**.
7. Click **Yes** in the **Keep this filter active?** dialog box to start defining the fields in the record. If you want to define the fields later click **No**, and select the filter from the drop-down list in the **Edit Fields/Expressions** tab when you want to define the fields.

   When the filter is active the areas of the data file that are excluded from the record are highlighted in black in the **Edit Fields/Expressions** preview table. Select the individual fields from the white areas of the preview table to add them to the record.

Activate a data filter

When you create a data filter in the **Add New Data Filter** tab in the **Table Layout** dialog box, you are prompted to activate the data filter. You can also activate an existing filter, change the active filter, or deactivate a selected filter, by selecting the appropriate option from the drop-down list in the **Edit Fields/Expressions** tab in the **Table Layout** dialog box.

When a data filter is activated, only records and fields that meet the data filter criteria can be viewed and processed.

When a data filter is active the following behaviors apply in the **Edit Fields/Expressions** tab:

- Any fields you define are associated with the active data filter.
- The fields in the table are evaluated against the criteria for the data filter. Fields that do not meet the criteria are not displayed in the **fields** list.
- Rows that do not meet the filter criteria are highlighted in black in the data preview table.
- Select the "All fields" option from the drop-down list to deactivate the active filter and display all fields defined for the table.

Steps

1. Select **Edit > Table Layout**.
2. In the **Edit Fields/Expressions** tab, select the data filter to activate from the drop-down list above the fields list.

   The fields list is updated to only show the fields that belong to the selected data filter. To deactivate the data filter select “All fields” from the drop-down list.

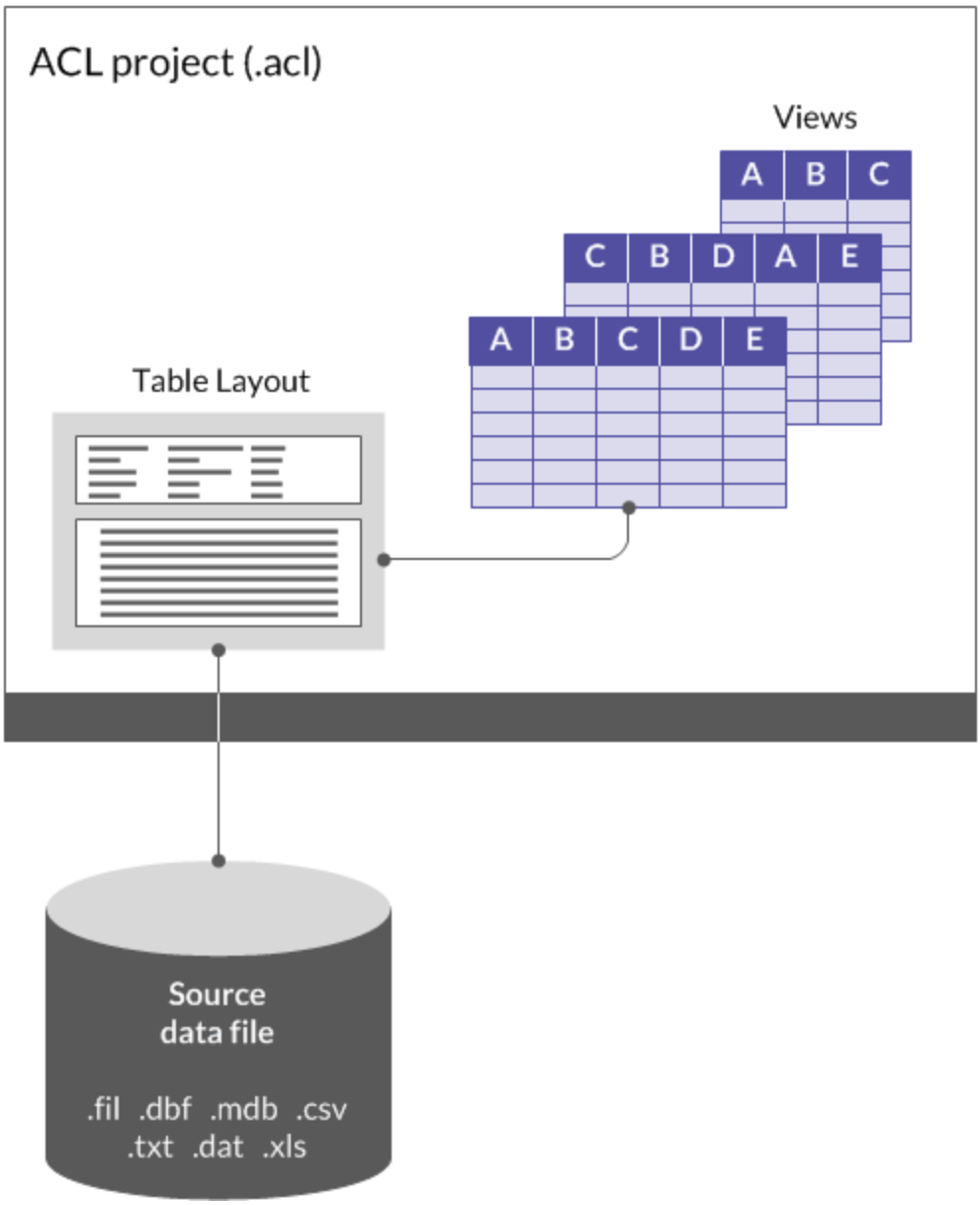
Displaying data with table views

Table views in Analytics let you arrange the way Analytics table data is displayed. Views appear in the Analytics display area and contain named columns and numbered records. The columns in a view represent fields in the associated table layout – either physical data fields or computed fields.

How table views relate to Analytics tables

The diagram below illustrates views in relation to the other components of an Analytics table.

For more information about how views relate to Analytics tables generally, see "The structure of Analytics tables" on page 115.



Views provide flexibility

Views are a flexible tool that allow you to display just the columns of data you want to see in the associated Analytics table, and to arrange the columns in the manner most effective for your analysis.

You can create multiple views for the same table, each of them with a different selection and arrangement of fields from the table. For example, if you are interested in only five fields in a table that contains a hundred fields, you can create a view with only the five relevant fields, and put them in the order you find most convenient.

You can also use views to format the data for Analytics reports.

The Analytics Default_View

When you open a new table for the first time in Analytics, a view called “Default_View” is automatically created. The “Default_View” includes all the fields defined in the associated table layout, in the order in which they appear in the layout.

You can use the default view without modification, modify it to better suit your needs, modify it and save it as a new view, or create new views that are not based on the default view.

The relation between views and table layouts

Although views are associated with an underlying table layout, **changes to the view do not change the table layout**. For example, if you remove a column from a view, the corresponding field is not deleted from the table layout.

Changes to the table layout are not automatically displayed in existing views. For example, if you create a new computed field or define a new physical data field in a table layout, you need to manually add it to any views where it is required.

Deleting a field from the table layout does automatically remove the column from any associated views.

Working with views

The sections below explain the various operations that you can perform with views. In a number of cases, you can perform the operation in more than one location in Analytics.

Locations where view operations can be performed:

- The View tab
- The **Overview** tab of the **Navigator**
- The **Table Properties** dialog box
- The **Project Properties** dialog box

Create a view

The first time you open an Analytics table a view named **Default_View** is automatically created that includes all the fields defined in the table layout with their default properties.

You can modify this default view and save it with a different name to create a new view, or you can create a new empty view and add columns and define settings.

The procedure below outlines the steps for creating a new empty view and adding columns.

Show me how

1. Open the table that you want to create the view for.
2. Right-click a view button at the bottom of the View tab and select **New**.
3. In the **Add View** dialog box, enter a name for the view and click **OK**.

> **Note**
> View names are limited to 64 alphanumeric characters. The name can include the underscore character (_), but no other special characters, or any spaces. The name cannot start with a number.

4. In the **Add Columns** dialog box, complete any of the following tasks:
   - Click **Add All** to add all the table layout fields to the view.
   - Click an individual field in the **Available Fields** list and then click the right-arrow button to add it to the view.
   - **Ctrl+click** multiple fields in the **Available Fields** list and then click the right-arrow button to add them to the view.
   - Click **Expr** to open the **Expression Builder** and create an expression or computed field to add to the view.
   - If you want to use an expression to modify a field after you add it to the **Selected Fields** list, select the field and click **Edit** to open the **Expression Builder**.
5. (Optional) In the **Selected Fields** list, select one or more fields, and use the Up arrow  or the Down arrow  to reposition the fields.

6. You can add fields from any related tables by selecting the related table in the **From Table** drop-down list, and adding the fields you want to include to the **Selected Fields** list.
7. Click **OK**.

Open a view

The first time you open an Analytics table, a view named **Default_View**, containing all the fields defined in the table layout, is automatically created and opened.

When you open a table that has multiple views associated with it, the view that was last active is opened automatically.

You can also manually open any view associated with a table.

Show me how

1. In the **Navigator**, double-click the table with the view that you want to open.

   Analytics displays the view that was last active.
2. To open a different view, click the appropriate view button at the bottom of the View tab.

   The view is opened and the button for the active view is highlighted.

Saving changes to a view

When you modify a view you can either save the changes in the existing view, or save the modified view as a new view and leave the original view unchanged.

Save changes in the existing view

Show me how

1. Select **File > Save Project**.
2. Click **Yes** in the confirmation dialog box.

   You are also prompted to save changes to an existing view when you perform any of the following actions:

   - switch to another view
   - close the table
   - open another table

Save changes as a new view

Show me how

1. Right-click the highlighted view button at the bottom of the View tab and select **Save as**.
2. In the **Save View As** dialog box, enter the name for the new view and click **OK**.

   The new view is created and opened automatically. The original view is not modified.

Rename a view

You can rename a view to make it easier to understand the purpose of the view or the information displayed in the view.

You can use either the View tab, or the **Overview** tab in the **Navigator**, to rename views. If you need to rename a large number of views, the **Overview** tab is more convenient.

> **Note**
> View names are limited to 64 alphanumeric characters. The name can include the underscore character (_), but no other special characters, or any spaces. The name cannot start with a number.

Rename a view using the View tab

Show me how

1. Open the table with the view that you want to rename.
2. Right-click the view button at the bottom of the View tab and select **Rename**.
3. Enter the new name in the **Rename View** dialog box and click **OK**.

Rename a view using the Overview tab

Show me how

1. Open the table with the view that you want to rename.
2. In the **Overview** tab of the **Navigator**, right-click the table and select **Properties**.
3. Click the **Views** tab.
4. Select the view to rename and click **Rename**.
5. Enter the new name in the **Rename View** dialog box and click **OK**.
6. (Optional) Repeat the process for any other views that you want to rename.
7. Click **OK** to close the **Table Properties** dialog box.

Copy a view

You can copy a view to associate an identical view with the same Analytics table, and then subsequently modify the copied view. You can also copy views between tables in an Analytics project.

> **Tip**
> Copying and modifying a view may be easier than creating a new view from scratch.

You can use either the View tab, or the **Overview** tab in the **Navigator**, to copy views. If you need to make multiple copies, the **Overview** tab is more convenient.

Copy a view using the View tab

Show me how

1. Open the table with the view that you want to copy.
2. Right-click the view button at the bottom of the View tab and select **Save As**.
3. In the **Save View As** dialog box, enter a name for the copied view and click **OK**.

Copy a view using the Overview tab

Show me how

1. Open the table with the view that you want to copy.
2. In the **Overview** tab of the **Navigator**, right-click the table and select **Properties**.
3. Click the **Views** tab.
4. Select the view that you want to copy and click **Copy**.

   A copy of the view with an incrementing numeric suffix is created.
5. (Optional) Repeat the process to create multiple copies of the same view, or to copy other views.
6. Click **OK** to close the **Table Properties** dialog box.

Copy a view from another table in the project

> **Note**
>
> If you copy views between tables and the destination table does not include all the fields specified in the view, you get an error message listing one or more fields as "undefined". You may still be able to use the view, but view columns associated with undefined fields do not appear.

Show me how

1. Open the table that will contain the copied view.
2. In the **Overview** tab of the **Navigator**, right-click the Analytics project and select **Properties**.

   The Analytics project is the top-level folder in the treeview.
3. Click the **Views** tab.
4. Select the view that you want to copy, click **Apply**, and click **OK**.

Copy a view from another Analytics project

You can copy a view from one Analytics project to another, which is useful if you have similar tables in both projects and you want to reuse a view rather than create a new view from scratch. You can copy a single view, or multiple views simultaneously.

If a table is open in the destination project, the copied view is automatically associated with the open table. If no table is open, the copied view is added to the project and can be associated with a table at a later point.

If a copied view has the same name as an existing view in the project, the copied view is given an incrementing numeric suffix.

> **Note**
>
> If you copy views between tables and the destination table does not include all the fields specified in the view, you get an error message listing one or more fields as "undefined". You may still be able to use the view, but view columns associated with undefined fields do not appear.

Show me how

Copy the view

1. Open the project that will contain the copied view or views.
2. Do one of the following:
   - If you want to immediately associate a copied view or views with a table, open the table.
   - If you want to associate a copied view or views with a table at a later point, make sure all the tables in the destination project are closed.

     Use this second method if you are copying multiple views that you intend to associate with different tables.
3. In the **Overview** tab of the **Navigator**, right-click the Analytics project entry, or a project folder, and select **Copy from another Project > View**.

   The Analytics project is the top-level folder in the treeview.
4. In the **Locate Project File** dialog box, locate and select the Analytics project that you want to copy the view or views from and click **Open**.
5. In the **Import** dialog box, complete any of the following tasks to add one or more views to the **To *project_name*** list:
   - Double-click a view.
   - **Ctrl+click** multiple views and then click the right-arrow button.
   - Click **Add All** to add all the views.

   You can remove views from the **To *project_name*** list by double-clicking an individual view, by using **Ctrl+click** to select multiple views and then clicking the left-arrow button, or by clicking **Clear All**.
6. Click **OK** to copy the view or views into the destination project.

   If a table is open, the view or views are associated with the table.

Associate the view

If you copied one or more views without associating them with a table, do the following to associate a view with a table:

1. Open the appropriate table.
2. Right-click the Analytics project in the **Overview** tab in the **Navigator** and select **Properties**.
3. Click the **Views** tab.
4. Select the view that you want to associate with the table, click **Apply**, and click **OK**.

Import a view

You can import a view that exists as a separate `.rpt` file outside an Analytics project, which allows you to reuse the view rather than create a new view from scratch. You can import only one view at a time.

If a table is open in the project when you import the view, the imported view is automatically associated with the open table. If no table is open, the imported view is added to the project and can be associated with a table at a later point.

If an imported view has the same name as an existing view in the project, the imported view is given an incrementing numeric suffix.

> **Note**
> If the table you are associating an imported view with does not include all the fields specified in the view, you get an error message listing one or more fields as "undefined". You may still be able to use the view, but view columns associated with undefined fields do not appear.

Show me how

Import the view

1. In the **Overview** tab of the **Navigator**, right-click the Analytics project entry and select **Import Project Item > View**.

   The Analytics project is the top-level folder in the treeview.
2. In the **Project** dialog box, locate and select a view file (`.rpt`) and click **Open**.
3. Click **OK** in the confirmation dialog box.

   The view is imported into the project. If a table is open, the view is associated with the table.

Associate the view

If you imported a view without associating it with a table, do the following to associate the view with a table:

1. Open the appropriate table.
2. Right-click the Analytics project in the **Overview** tab in the **Navigator** and select **Properties**.
3. Click the **Views** tab.
4. Select the imported view, click **Apply**, and click **OK**.

Export a view

You can export a view as a separate `.rpt` file saved outside the Analytics project. A view exported as a separate file can later be imported into any Analytics project, which allows you to reuse the view rather than create a new view from scratch. You can export only one view at a time.

Show me how

1. Open the table associated with the view.
2. In the **Overview** tab of the **Navigator**, right-click the table and select **Properties > Views**.
3. Select the view that you want to export and click **Export**.
4. In the **Save View As** dialog box, choose a location to save the view, rename the view if required, click **Save**, and click **OK** in the confirmation dialog box.

   The view is exported to the location you specified.

   > **Note**
   >
   > Limit the view name to 64 alphanumeric characters, not including the `.rpt` extension, to ensure that the name is not truncated when the view is imported back into Analytics.
   >
   > The name can include the underscore character (_), but do not use any other special characters, or any spaces, or start the name with a number. Special characters, spaces, and a leading number are all replaced by the underscore character when the view is imported.

5. Click **OK** to close the **Table Properties** dialog box.

Delete a view

You can delete individual views associated with a table whenever necessary, but there must always be one view associated with a table. Analytics prevents you from deleting the last view associated with a table.

Deleting a view has no effect on the table layout or an associated data file. You can use either the View tab, or the **Overview** tab in the **Navigator**, to delete views.

Delete a view using the View tab

Show me how

1. Open the table with the view that you want to delete.
2. Right-click the view button at the bottom of the View tab and select **Delete**.
3. Click **Delete** in the confirmation dialog box.

Delete a view using the Overview tab

Show me how

1. Open the table with the view that you want to delete.
2. In the **Overview** tab of the **Navigator**, right-click the table and select **Properties**.
3. Click the **Views** tab.
4. Select the view to delete and click **Delete**.

   You cannot delete a view if it is currently active.
5. Click **Delete** in the confirmation dialog box.
6. Click **OK** to close the **Table Properties** dialog box.

Customizing columns in views

You can customize how individual columns in Analytics views are displayed on screen, and displayed and processed in Analytics reports.

You can specify:

- which columns from the table layout are included in the view
- the order of columns
- how numeric values are displayed in individual columns
- column display names
- column properties in print reports

Add columns to a view

You can add any of the physical data fields or computed fields defined in a table layout as columns in a view.

You can also add columns to a view based on ad hoc expressions created using the **Expression Builder**.

Show me how

1. Do one of the following:
   - Right-click a column header in the view to insert one or more columns to the left of the selected column
   - Right-click in the display area without a column selected to add one or more columns after the last column
2. Select **Add Columns**.
3. In the **Add Columns** dialog box, complete any of the following steps:
   - Click **Add All** to add all of the columns to the view.
   - Click an individual column in the **Available Fields** list and then click the right-arrow button to add it to the view.
   - **Ctrl+click** multiple columns in the **Available Fields** list and then click the right-arrow button to add them to the view.
   - Click **Expr** to open the **Expression Builder** and create an expression to add to the view.
   - If you want to use an expression to modify a column after you add it to the **Selected Fields** list, select the column and click **Edit** to open the **Expression Builder**.
4. If the table has one or more related tables associated with it, you can add fields from any of the related tables to the view by selecting a related table in the **From Table** drop-down list, and adding the fields you want to include to the **Selected Fields** list.
5. Click **OK**.

Remove columns from a view

You can remove unwanted columns from a view. The physical data fields or computed fields the columns are based on are still present in the table layout, and the columns can be re-added whenever necessary and remain available for use in other views.

Show me how

1. In the View tab, click the column header of the column you want to remove.

   You can **Shift+click** to select multiple adjacent column headers, and **Ctrl+click** to select multiple non-adjacent column headers.

2. Right-click in the data area of the view and select **Remove Selected Columns**.

   > **Tip**
   > Do not right-click in the column header row if you have selected multiple columns, or only the column you right-click will be selected.

3. Click **Remove** in the **Remove Columns** dialog box.
4. To save your changes to the view, select **File > Save Project** and click **Yes** in the confirmation dialog box.

Resize columns in a view

You can resize one, several, or all columns in a view.

Show me how

Do one of the following:

To manually resize a single column	Position the cursor over the right-hand separator between column headers and drag the column to the required size.
To automatically resize one or more columns to the width of the column contents	Select the column header(s) and double-click one of the right-hand column separators. You can **Shift+click** to select multiple adjacent column headers, and **Ctrl+click** to select multiple non-adjacent column headers.
To automatically resize all columns in a view to the width of the column contents	Right-click in the data area of the view and select **Resize All Columns**.

Reorder columns in a view

You can modify the order in which columns are displayed in a view. If you want to reorder multiple columns, you need to select each column individually and move it to the appropriate position.

Show me how

1. Click the header of the column you want to move and drag it to a new location.

   You need to drag the column near the line separator between the two columns where you want to move the selected column.
2. Reposition any other columns you want to move, and then select **File > Save Project** to save your changes to the view.

Rename columns in a view

You can rename one or more columns in a view, if required. Renaming a column changes only the display name and has no effect on the underlying name of the field in the table layout.

You can rename either the default column name used by all views associated with the table, or you can rename only the column name used by a particular view.

> **Note**
> Column names specified within individual views override default column names.

Rename the default column name

Show me how

1. Select **Edit > Table Layout**.
2. In the **Edit Fields/Expressions** tab, double-click the column name you want to rename.
3. Change the column name in the **Alternate Column Title** field.
4. Click **Accept Entry** and click Close to exit the **Table Layout** dialog box.

   All columns that use the default name in all views associated with the table are automatically updated. Columns that have a name specified within an individual view are not updated.

Rename the column name used by a particular view

Show me how

1. Right-click the column header and select **Properties**.
2. Change the column name in the **Alternate Column Title** field, and click **OK**.

Change a column from a view-level name to the default name

Show me how

1. Right-click the column header and select **Properties**.
2. Delete the name from the **Alternate Column Title** field, and click **OK**.

3. Switch back and forth between views, or close and reopen the table, to refresh the column name.
4. Click **Yes** when you are prompted to save the changes to the table.

Format numeric values in a view

The formatting applied to numeric values displayed in Analytics views and reports can be configured at three different levels:

- application level (global)
- field level
- column level

> **Note**
>
> These formatting options apply to numeric values in fields that use the numeric data type. They do not apply to numbers in fields that use the character data type.

Level	Overrides	Location to set formatting	Description
Application level	n/a	a. Select **Tools > Options > Numeric**. b. Select or specify a format in **Default Numeric Format**.	Specifies the formatting for all numeric fields and columns in Analytics that do not have field-level or column-level formatting specified
Field level	Application level	a. Open a table. b. Select **Edit > Table Layout**. c. Double-click a field name. d. In the **Edit Fields/Expressions** tab, select or specify a format in **Format**.	Specifies the formatting for an individual numeric field in a table layout, and the associated column in all views that use the table layout, unless column-level formatting is specified For a change in field-level formatting to take effect in a view, you must remove the associated column and re-add it to the view, or create a new view containing the column
Column level	Application level Field level	a. Open a table. b. In the view, right-click a column header and select **Properties**. c. In the **Modify Column** dialog box, select or specify a format in **Format**.	Specifies the formatting for an individual numeric column in an individual view If you have more than one view of a table, you can format the same column differently in the different views. For example, you could display a column with dollar signs in a view you use for printed reports, and omit the dollar signs in a view you use for analysis.

Numeric format syntax

The formatting applied to numeric values in Analytics is specified using a format mask that defines the required layout for each numeric value. For example, the format mask `$99` could be used to display any value less than $100, because each 9 indicates that any digit between 0 and 9 can be displayed, with the dollar sign displayed next to the value.

Default format masks

The table below lists the default format masks that are available in Analytics. It also shows how each mask displays the number **-100234.56**.

You can use the default format masks as defined, modify them to suit your requirements, or define you own format masks.

Default format mask	Displays -100234.56 as:
-999999.99	-100234.56
-9,999,999.99	-100,234.56
(9,999,999.99)	(100,234.56)
-$9,999,999.99	-$100,234.56
($9,999,999.99)	($100,234.56)
9,999,999.99-	100,234.56-

Format mask components

Format masks are defined using the following components:

Component	Description
Digit placeholder	The number 9 is used to specify each place where a single digit between 0 and 9 can be displayed. If you specify more placeholders for digits than required, the extra digits and anything between them, such as commas, are not displayed. For example, if the format mask was specified as $9,999, a value of 310 would be displayed as $310. You should specify formatting for the maximum number of digits that may appear in the column if you are using special formatting because any extra digits in your data are added immediately to the left of the leftmost 9, with no additional punctuation. For example, if your format mask is `-9,999.00` a value of `1000000.00` will be incorrectly formatted as `1000,000.00` (with no thousands separator after the `1`).
Negative value indicator	The negative value indicator can be:

Component	Description
	◦ a minus sign before or after the number: `-100.00` or `100.00-` ◦ `CR` before or after the number: `CR100.00` or `100.00CR` ◦ parenthesis: `(100)` If no negative value indicator is specified, a minus sign before the number is used by default.
Thousands separator (if any)	Large numbers often have formatting applied between groups of digits to make them easier to read. The most common separators are commas (100,000.00) or spaces (100 000.00). Some regional settings use a period as the separator and a comma to indicate decimal values. The default value used in Analytics is specified in the **Thousands Separator** text box in the **Numeric** tab in the **Options** dialog box (**Tools > Options**).
Decimal point indicator	A period is most often used to indicate decimal values, but a comma is used in some regional settings. The default value used in Analytics is specified in the **Decimal Place Symbol** text box in the **Numeric** tab in the **Options** dialog box (**Tools > Options**).
Value indicator sign (if any)	A dollar sign, percentage sign, and so on, can be added to the format to identify the type of value being displayed.

Modify column properties

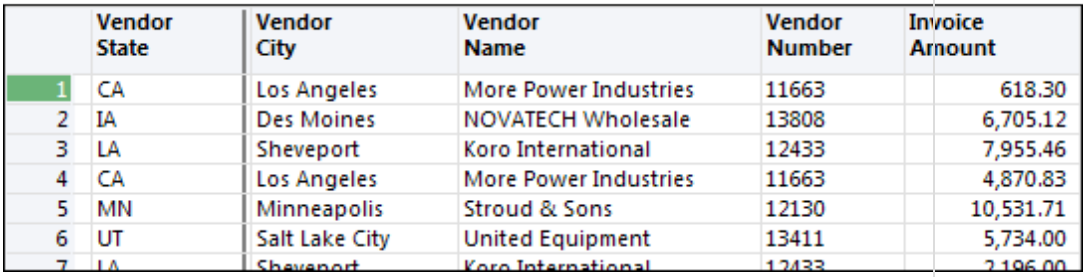
Each column in an Analytics view has a number of properties that can be configured to modify how data is displayed on screen and in reports generated from the view. The properties configured for columns in a view do not alter the settings in the table layout or in other views associated with the table.

If you create a copy of a view, using the **Save As** command, the properties of columns in the view are copied, but subsequent changes to column properties in either view only apply to the view in which the changes are made.

1. In the View tab, right-click the column title and select **Properties**, or double-click the column title.
2. Make one or more changes in the **Modify Column** dialog box, using the table below as a guide, then click **OK**.

Column type	Option	Description
Any	**Column Contents**	Allows you to modify the values displayed in the column. Click **Column Contents** and use the **Expression Builder** to create or edit an expression. Only the display of the values is modified, not the physical data. The expression in **Column Contents** must return the correct data type for the column. For example, an expression for a numeric column must return a numeric value.

Column type	Option	Description
		Any fields referenced in the expression must exist in the table layout.
	Alternate Column Title	The text for the display name of the column in the view. **Note** You are changing the column title in the current view only, which overrides the default column title specified in the **Table Layout** dialog box. For more information, see "Rename columns in a view" on page 780.
	Width	The display width of the column on screen or in a report. Enter the width in characters. For numeric columns, ensure that the column is wide enough to display the values with the largest number of digits. If a full numeric value cannot be displayed on screen, a string of number signs (######) is displayed to indicate an error.
	Sort Key Column	Report output only The column is used to sort data in report output. Select **Sort Ascending** or **Sort Descending** to choose the sort order. **Note** You must also select **Presort** in the **Report** dialog box.
Numeric	**Format**	The format used to display numbers in the column. Select the appropriate format from the **Format** drop-down list, or enter a custom format. If no format is specified, the **Default Numeric Format** is used, which is specified in the **Numeric** tab in the **Options** dialog box (**Tools > Options**).
	Suppress Totals	Report output only Prevents the values in the column from being totaled. By default, Analytics automatically totals numeric fields in reports. You can suppress this behavior if the field contains data, such as unit prices, for which calculating the total is not meaningful.
	Blank if Zero	Report output only Substitutes blank entries for zero values in the column. A report that includes a large number of zero values in a column is more readable if only the non-zero values are displayed.
Character (must be the leftmost column or	**Break Column**	Report output only Creates a subsection each time the value in the column changes, and calculates a subtotal for all numeric fields. You can specify multiple break columns to create nested subsections. All break columns must be positioned together, on the left side of the view, before the first

Column type	Option	Description
columns in the view)		non-break column. This option is enabled only if the **Sort Key Column** checkbox is selected. **Note** You must also select **Presort** in the **Report** dialog box. A heavy, dashed separator appears to the right of the break column or columns in the view. 
	Page Break	Report output only Inserts a page break each time the **Break Column** value changes. This option is enabled only if the **Break Column** checkbox is selected.
	Suppress Duplicates	Report output only Substitutes blank entries for repeated values in the **Break Column**. For example, if the customer name is listed for each invoice record, you can make the report more readable by listing only the first instance of each customer name. This option is enabled only if the **Break Column** checkbox is selected.

	Vendor State	Vendor City	Vendor Name	Vendor Number	Invoice Amount
1	CA	Los Angeles	More Power Industries	11663	618.30
2	IA	Des Moines	NOVATECH Wholesale	13808	6,705.12
3	LA	Sheveport	Koro International	12433	7,955.46
4	CA	Los Angeles	More Power Industries	11663	4,870.83
5	MN	Minneapolis	Stroud & Sons	12130	10,531.71
6	UT	Salt Lake City	United Equipment	13411	5,734.00
7	LA	Sheveport	Koro International	12433	2,196.00

Copying data from views

You can copy data from an Analytics view and paste it into another application. Copying data from a view is an alternative to exporting the data from Analytics. Copying the data may be a faster and simpler option in some situations. For example, when the view is sorted using the **Quick Sort Ascending** or **Quick Sort Descending** menu command and you want to select a subset of the data.

To copy data from a view:

1. Do one of the following:
   - Click and drag in the View tab to highlight the group of values you want to copy.
   - Click the column header of the column you want to copy and, if you want to copy multiple columns, **Ctrl+click** the other column headers to select them.
2. Right-click in the data area of the view and select **Copy**. Do not click in the column header if you have selected multiple columns or highlighted a group of values, or only the column you right-click on will be selected.

   The data is copied to the clipboard. You can open the application you want to copy the data to, such as a spreadsheet or text file, and use the Paste command in that application to insert the data.

Generating graphs from views

You can select data in an Analytics view and generate a graph. You can select either multiple numeric columns, or a single character or datetime column and one or more numeric columns.

To graph data from a view:

1. Open the view that contains the data you want to graph.
2. Do one of the following:
   - Click and drag in the View tab to select the data to include.
   - Select multiple adjacent columns by clicking the first column header to include and then **Shift+click** to select the last column to include. Any columns between the first column and the last column are included in the selection.
   - **Ctrl+click** in the column headers to select multiple non-adjacent columns to include.
3. Right-click in the view and select **Graph Selected Data**.

   > **Note**
   > Do not right-click a column header if you have selected multiple columns or a group of values, because only the column you right-click will be selected.

   If the menu item is disabled it means that the selected data area or columns cannot be graphed.

   The graph is displayed in a new graph window. You can modify any of the graph properties in the graph window, but the Drill-down command is disabled because the subset of data used to create the graph is already highlighted in the view.

How Analytics displays invalid data in views

Analytics uses the following distinct character sequences to highlight invalid data in views:

- ###### – A series of number signs is used to identify numeric values where the width of the column is less than number of digits to be displayed.

  These values are highlighted as invalid so that an incorrect value is not displayed on screen (e.g., 100 being displayed instead of 10100), and because the value will cause reports that include the value to be misaligned. The full value is printed in the report which will result in the adjacent columns being misaligned.

  You can correct this problem by increasing the **Width** property of the column to a value large enough to display all of the digits.

- ### ERR ### – This character sequence is displayed when the numeric value in a column is invalid or undefined.

  Numeric overflow is caused by operations such as division by zero or division by a very small number. You need to correct the value or remove the column from the view before you can generate a report from the view.

If the numeric value is larger than the column width, and the value is invalid or undefined, the number sign character sequence (######) will be displayed until the width of the column is increased, and then the ### ERR ### character sequence will be displayed.

Opening multiple tables

You can have multiple tables open simultaneously in Analytics, which provides a convenient method for visually comparing two or more sets of data. To make comparisons easier, you can apply filtering and quick sorting to the open tables, and the results of these operations are preserved as you switch between tables. Opening multiple tables is an interface-only feature, and does not apply to Analytics scripting.

To open multiple tables in Analytics you pin the View tab for a table you want to keep open, and then open one or more additional tables. You keep any of the additional tables open by also pinning them. If a table is not pinned, it closes automatically whenever you open another table.

An additional View tab opens for each additional table you open. You can switch between tables by clicking the individual View tabs, or by clicking the table name in the **Navigator**. The table currently selected for display is the active table and the active table name appears in bold on the View tab and in the **Navigator**.

Steps

You can open multiple tables simultaneously in Analytics. Each table appears in a separate View tab, allowing you to switch back and forth between different tables. The tables must all be contained in the same Analytics project.

1. In the **Navigator**, double-click the first table to open it.
2. Click the pin icon to pin the View tab.
3. Open one or more additional tables and click the pin icon for each table before opening the next table.

   To move back and forth between tables, click the individual View tabs, or click the table name in the **Navigator**, where open tables are indicated with a green dot. The name of the active table (currently displayed table) appears in bold in the **Navigator**.
4. To close a pinned table, click on the appropriate View tab.
5. To close all open tables, click **Close All Tabs** to the right of the View tabs.

The benefit of opening multiple tables

The general benefit of being able to open multiple tables simultaneously is that in some circumstances you can perform useful analysis without being required to first process the tables using Analytics operations such as joining or relating.

For example, consider a review of T&E transactions for banned vendors (or suspicious merchant category codes). You have two tables, one containing T&E transactions, and another containing banned vendors. You could join the two tables on vendor name, an operation that could require a lot

of preliminary work to manually harmonize the two key fields before performing the join. Or you could open both tables, and using the banned vendor list as a reference, create filters in the transactions table that isolate any occurrences of the banned vendors. If appropriate, you could build a fuzzy search component into the filtering of the T&E transactions. The filters below provide two examples:

- `ISFUZZYDUP(vendor_name,"Diamond Casino", 5, 99)` isolates exact occurrences and fuzzy duplicates of Diamond Casino.
- `MATCH(mc_code, "7298", "7995", "9222")` isolates exact occurrences of merchant category codes 7298 (Health and Beauty Spas), 7995 (Betting), and 9222 (Fines).

For more information, see "ISFUZZYDUP( ) function" on page 2198 and "MATCH( ) function" on page 2231.

Additional information about opening multiple tables

Functional area	Details
General table behavior	Tables behave the same way whether only one table is open, or multiple tables are open.
Table state preserved	The state of each table is preserved as you switch between tables. Table state includes the following elements: ◦ filtering ◦ quick sorting ◦ record selection ◦ column, record, or field highlighting ◦ scroll bar position ◦ active view selection ◦ active index selection
Analytics operations	Analytics operations are performed on the active table only.
Closing the View tab and closing tables	◦ Closing the View tab containing a table also closes the table. You cannot perform any operations on a closed table. Prior to version 10.0 of Analytics, closing the View tab removed the table data from view, however the table remained in an open state and you could continue to perform operations on the data.
Primary and secondary tables (Applies to the Analytics interface only, including the command line. Does not apply to Analytics scripting.)	◦ One or more open tables can be primary tables with associated secondary tables. Each primary-secondary association is unique and has no effect on any other primary-secondary association. ◦ Different primary tables can have the same secondary table. ◦ A table that is open in the View tab cannot be opened as a secondary table. It must first be closed before being opened as a secondary table. ◦ A table open as a secondary table cannot be opened in the View tab. It must first be closed before being opened in the View tab.

Functional area	Details
	Note Opening a secondary table means associating it with a primary table and making it available for processing. Secondary tables are not opened in the View tab. ◦ Unassociated or 'free-floating' secondary tables are not permitted: • If you close a primary table, the associated secondary table is automatically closed. • Another table must already be open before you can open a table as secondary. • This restriction does not apply to Analytics scripting, which still allows unassociated secondary tables.
Analytics projects	Multiple open tables must all be contained in the same Analytics project. If an Analytics project has multiple tables open when you close the project, only the active table is automatically opened when you reopen the project. The open state of the other tables, and any primary-secondary associations, are not preserved.
Server tables	Multiple-table mode is supported for local tables only. You cannot open more than one server table at a time. If you have one or more local tables open when you open a server table, all the local tables are automatically closed.
Analytics command log	In multiple-table mode, every time you switch to a different table and perform an action or an operation on the table that causes an entry to be written to the Analytics command log, the log entry is preceded by an **OPEN <table name>** log entry. When multiple tables are open, this preceding log entry identifies which table is the target of an operation. Pinning a table, or switching between tables without performing any subsequent operation, does not create a log entry.
Data Definition Wizard	When you use the Data Definition Wizard to import data and define a table all open tables, whether pinned or not, are automatically closed. Closing all open tables ensures that a table can be overwritten, if necessary. Running the IMPORT command from the command line does not close any open tables. If you attempt to overwrite a table using this method, and the table is open and active, the command fails and an error message appears.
Scripting	◦ You cannot open multiple tables using an Analytics script. Multiple-table mode is an interface-only feature and does not have an equivalent in ACLScript. ◦ Pinning a table to keep it open in the Analytics interface does not prevent the table from being closed with the CLOSE command. ◦ When you run an Analytics script, all open tables, whether pinned or not, are automatically closed. The active table, and an associated secondary table, if one exists, are then automatically reopened, unless the script specifies opening a different table. Automatically reopening the active table and any associated secondary table supports the use of scripts that assume manually opening a table prior to running a script.

Formatting records to span multiple rows

You can customize the layout of views so that each record is displayed on more than one row. You can use this feature to display all of the information you are interested in on screen, and avoid having to scroll to see additional information. For example, if you have a table with a number of columns you are interested in and a column that includes notes that provide more information, you could display the data columns in one row and the notes on a separate row. You can also use this feature to optimize the layout of reports generated from the view. For example, if you have two address fields in a table (address_1 and address_2) you can align the address_2 field on the row below the address_1 field in the report, to make it more readable than if these fields were side by side.

To format records in a view to span multiple rows:

1. Position the cursor over the bottom line for any record in the record number column. The record number column is the gray area to the left of the first column in the view.
2. Click and drag the cursor down to add rows to the record, or drag up to remove blank rows. When you modify the number of rows for an individual record, the change is applied to all records in the table and the column header is modified by an equivalent number of rows to reflect the change.
3. To move data in a column to a different row, click and drag the column in the column header to the row in the column header that you want to add it to. The column is added to the leftmost position in the row, after any columns already in the row.
4. Modify the position of individual columns in the row by positioning the mouse cursor over the vertical lines that mark the start and end of the column and click and drag the line to the new position.
5. If you do not want to display all of the lines in the column header in the view, and you want to suppress some of the column headers when the view is printed, you can modify which column headers are visible. For example, if the first line has an entry for an “Address” header and the second line has an entry for an “Address_2” header, you will typically want to suppress the second line. To change the visible lines, position your mouse cursor over the black line above the scroll bar on the right-hand side of the view and click and drag the cursor up to hide column headers you do not want to include. The columns in the top row of the header cannot be hidden because the column header must include at least one row.
6. After you have set the position of the columns and column headers, save the changes to the existing view or to a new view.

Preparing data for analysis

You may need to do some initial, preparatory work on the data that you intend to analyze. In some cases, you can start analyzing data as soon as you have imported it. But often you need to perform one or more preparatory tasks in order to:

- shape the data set that you will ultimately analyze
- ensure that the results are reliable

Think of preparation as the foundation upon which your analysis will be built. A good foundation is critical to effective, reliable results.

Common preparation tasks

Here are some of the more common preparation tasks:

- **convert** – convert the data type of fields so that they can be used as input for specific Analytics commands, or harmonize with other fields
- **clean and standardize** – clean and standardize input data so that output results are reliable
- **combine** – combine data from multiple tables
- **sample** – draw a sample of records because you may not have the time or the budget to examine every record in a data set

Any of these tasks could be an absolutely necessary first step before you perform your intended analysis.

Verifying data

In addition to preparing data, you should also verify the completeness and validity of any data that you are going to analyze. Even a small amount of invalid data can invalidate all your subsequent analysis, and waste valuable time and resources.

Using expressions

Analytics expressions are combinations of values and operators that perform a calculation and return a result.

Expressions are a valuable and flexible tool. You can use them to:

- perform a wide range of calculations
- create filters
- prepare data for analysis
- create computed fields

The content of expressions

Expressions can include data fields, functions, literals, constants, and variables, which can be combined using arithmetic or logical operators.

You can enter expressions manually, or you can build them using the **Expression Builder**, which provides a standard utility throughout Analytics for creating expressions.

The complexity of expressions

Expressions can be as simple or as complex as needed.

A simple expression could return the result of a basic arithmetic operation.

For example:

```
Quantity * Cost
```

A more complex expression could reference a number of fields and functions and use operators to combine the parts of the expression.

For example:

```
PROPER(first_name) + " " + PROPER(last_name)
```

converts all the names in the `first_name` and `last_name` fields to proper case (initial capital letter followed by lowercase), and joins the first and last names with a single space between them.

Types of expressions

Analytics supports four types of expressions, which correspond with the four supported data categories, or data types:

- character
- numeric
- datetime
- logical

For example:

- `Amount + 1` is a numeric expression because it performs an operation on numbers and returns a numeric result.
- `Amount > 1` is a logical expression because it makes a comparison and returns a logical result of True or False.

The content of any expression you create must correspond with the expression type:

Expression type	Required content	Example
Character	Contains any of the following: ◦ character fields ◦ variables that contain character data ◦ functions that return character values ◦ quoted character strings (character literals)	Extract the digits from a product code and discard the three-character prefix: ◦ `SUBSTR(Product_code, 4, 10)`
Numeric	Contains any of the following: ◦ numeric fields ◦ variables that contain numeric data ◦ functions that return numeric values ◦ literal numeric values, without quotation marks – limited to digits, a minus sign if needed, and a decimal point if needed	Calculate sale price plus tax: ◦ `Sale_price * 1.07` Find the maximum value across three fields: ◦ `MAXIMUM(Min_Qty, Qty_on_hand, Qty_on_order)`
Datetime	Contains any of the following: ◦ datetime fields ◦ variables that contain datetime data ◦ functions that return datetime values ◦ quoted datetime values (datetime literals) The Datetime data type encompasses three subtypes: date, datetime, and time. Quoted datetime values require backquotes – for example, `` `20141231` `` or `` `20141231.235959` ``. The backquote (`) is the lowercase key at the upper left corner of the keyboard.	Calculate the elapsed days between the two dates: ◦ `` `20141231` - `20141130` `` Calculate the elapsed time between values in two time fields: ◦ `Finish_Time - Start_Time`
Logical	Contains any of the following: ◦ an operation that produces a logical result of True or False (`T` or `F`) ◦ functions that return logical values	Find all records with a payment date past the due date: ◦ `Payment_date > Due_date`

Expression type	Required content	Example
	If `T` or `F` are part of the expression, they must be entered without quotation marks. **Note** A logical expression can reference fields, variables, or literals of any data type.	Filter records in a table on three cities: ◦ `MATCH(Vendor_City, "Phoenix", "Austin", "Los Angeles")`

How Analytics evaluates expressions

Analytics evaluates expressions according to the following rules:

Operator precedence	Arithmetic and logical precedence dictates the order in which operators are evaluated. See "Operators in Analytics expressions" below. Use parentheses **( )** to modify the order in which the operators are evaluated.
Operand data type	Each operator works only if its operands are of a compatible data type.
Function parentheses	All Analytics functions require parentheses. Everything inside a function's parentheses is evaluated first, before any other parts of an expression outside the function's parentheses.
Comparing character strings	By default, when character strings of different lengths are compared, the shorter of the two lengths is used. If the **Exact Character Comparisons** option is selected in the **Tables** tab in the **Options** dialog box, the longer of the two lengths is used. For more information, see "Table options" on page 122.
Decimal precision	If numbers with different decimal precision are mixed in a numeric expression, the result retains the number of decimal places of the operand with the largest number of decimal places. (This default behavior can be changed with the SET MATH command.) For example: ◦ 4 + 5.0 = 9.0 ◦ 6 * 2.000000 = 12.000000 ◦ 1.1 * 1.1 = 1.2 ◦ 1.1 * 1.10 = 1.21 For more information, see "Controlling rounding and decimal precision in numeric expressions" on page 803.

Operators in Analytics expressions

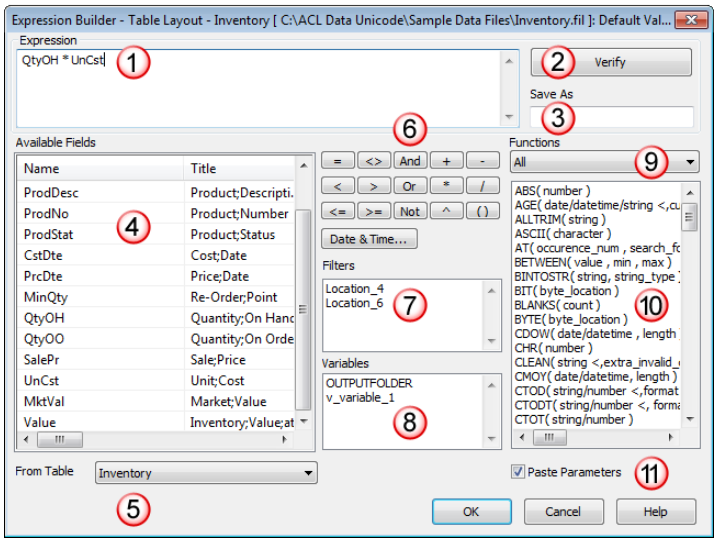
The table below lists the operators that are available for use when you create an expression.

The operators are listed in decreasing order of precedence. When operators of equal precedence appear in an expression, they are evaluated from left to right – unless you use parentheses to specify a particular order of evaluation.

Operators in order of precedence	Description
( )	Parentheses – modify operator precedence, or enclose function parameters
NOT **-**	Logical NOT Unary minus – minus sign, indicates a negative number
^	Exponentiation – raises a number to a power
***** **/** Operators have equal precedence and are evaluated from left to right	Multiply Divide
+ **-** Operators have equal precedence and are evaluated from left to right	Add Subtract
+	Concatenate character strings
> **<** **=** **>=** **<=** **<>** Operators have equal precedence and are evaluated from left to right **Note** Operators with two symbols must not contain a space. For example, type `>=` not `> =`.	Greater than Less than Equal to Greater than or equal to Less than or equal to Not equal to
AND (or **&**)	Logical AND
OR (or **|**)	Logical OR

Expression Builder overview

The **Expression Builder** is an Analytics utility that helps you create expressions. The **Expression Builder** is available wherever a user-defined expression can be created.



What is an expression?

An expression is a statement that combines elements such as data fields, operators, functions, filters, and variables that Analytics evaluates and returns a value for.

Validating expressions based on context

Depending on the context where the **Expression Builder** is displayed, Analytics checks that the expression evaluates to the required type of value. For example, if you are creating an expression used to filter the records in a view, the expression must evaluate to a logical value of True or False.

Expression Builder user interface elements

1	Expression text box	Text box for creating a new expression or editing an existing expression You can type the required syntax for the expression, or use the lists and buttons in the **Expression Builder** to enter required information. When you use the lists and buttons, the information is added at the current cursor position in the **Expression** text box.

2	Verify button	Automatically verify the syntax of the expression in the **Expression** text box Clicking **Verify** saves time by allowing you to quickly check the validity of an expression without leaving the **Expression Builder**.
3	Save As text box	Field for entering a name for a permanently saved expression
4	Available Fields list	Lists all data fields and computed fields in the selected table Double-click a field entry to add it to the **Expression** text box.
5	From Table drop-down list	Drop-down list for selecting any related tables An expression can contain fields from one or more related tables, even if the fields do not appear in the current view. You can also build filters that include fields from related tables and see the results in the current view, whether or not you add the fields from the related tables to the view.
6	Operator buttons, and Date & Time Selector	Buttons for adding operators, or dates, datetimes, or times, to the **Expression** text box
7	Filters list	Lists the named filters associated with the selected table Double-click a filter entry to add it to the **Expression** text box.
8	Variables list	Lists the variables associated with the selected table Double-click a variable entry to add it to the **Expression** text box.
9	Functions drop-down list	Lists function categories that can be used to filter the functions displayed in the **Functions** list
10	Functions list	Lists the functions available in Analytics and their required syntax. Optional parameters are enclosed in angle brackets (< >). Double-click a function to add it to the **Expression** text box. Depending on the type of function, the parameters can be constants, literals, field names, or expressions, which may include other functions. **Tip** For information about how to use the individual functions in the **Functions** list, click the **Help** button below the list.
11	Paste Parameters checkbox	◦ **Selected** - the function and the parameter placeholders are copied to the **Expression** text box ◦ **Not selected** - only the function, including opening and closing parentheses, is copied to the **Expression** text box

Creating expressions using the Expression Builder

The **Expression Builder** provides a standard interface for creating and modifying expressions. The **Expression Builder** dialog box is displayed as needed when you work in Analytics. It opens when you perform any of the following actions:

- Click **If** or **Expr** in any dialog box.
- Click **Edit View Filter**  in the View tab in the display area.
- Select **Edit > Table Layout**, click **Add a New Expression**  in the **Edit Fields/Expressions** tab, and then click **Expression**  .
- Select **Edit > Filters**, or **Edit > Variables**, and click **New**.
- Click **Insert Expression**  in the **Script Editor** toolbar.

When you open the **Expression Builder** by clicking **If**, or by editing a filter, you are restricted to creating logical expressions.

To create an expression:

1. In the **Expression Builder** dialog box, complete one or more of the steps listed below to build an expression.

   When you click or double-click to insert components of an expression, the components are inserted at the cursor location in the **Expression** text box.

   - Enter all or part of the expression manually in the **Expression** text box.
   - Double-click a field name in the **Available Fields** list. If you are working with a table that has related tables, you can select the table to add fields from in the **From Table** drop-down list.
   - Click a logical or mathematical operator button to insert an operator.
   - Click **Date & Time** and use the **Date & Time Selector** dialog box to insert a valid date, datetime, or time.
   - Double-click a filter name in the **Filters** list, or a variable name in the **Variables** list.
   - Double-click a function name in the **Functions** list.

     You can use the **Functions** drop-down list to filter the functions by type. If the **Paste Parameters** checkbox is selected the parameter placeholders are copied, and need to be replaced with appropriate fields or values. You also need to remove the angle brackets (< >) around any optional parameters you are including, or remove the optional parameters if you are not using them.

> **Tip**
> For information about how to use the individual functions in the **Functions** list, click the **Help** button below the list.

2. Click **Verify** to validate the expression you have built. The expression is checked to ensure that the syntax is correct and that the expression returns the correct value type, if a particular type is required.
3. Complete one of the following steps to create the expression:
   a. If you want to save the expression permanently, enter a name in the **Save As** text box and click **OK**. The name of the expression is copied into the text box or the location from which you accessed the **Expression Builder**. If you are creating a variable you must provide a name.
   b. Click **OK** to create the expression without saving it permanently. The expression is copied into the text box or the location from which you accessed the **Expression Builder**.

Controlling rounding and decimal precision in numeric expressions

In calculations that involve multiplication or division, Analytics rounds the result to the larger number of decimal places in the two operands. This method of rounding is associated with the fixed-point arithmetic that Analytics uses for evaluating most numeric expressions.

For multi-operand expressions (such as `a*b/c`), rounding is applied at each stage in the expression, starting with the first evaluated stage, and repeating until the expression is completely evaluated.

> **Note**
> Not being aware how rounding works in Analytics is one of the most common causes of computational errors.

Fixed-point arithmetic

The rounding behavior in Analytics is associated with the fixed-point arithmetic that Analytics uses for numeric operations (with the exception of financial functions). Analytics uses fixed-point arithmetic for two reasons:

- It improves processing speed
- It allows user control over decimal places and rounding

Rounding in multiplication

Consider the expression 1.1 * 1 .1. The correct answer is 1.21. However, Analytics rounds the result to one decimal place because that is the larger number of decimal places in the operands.

```
1.1 * 1.1 = 1.2
```

If one of the operands has two decimal places, Analytics rounds the decimal portion of the result to the larger number of decimal places in the operands. In this particular example, no rounding is required:

```
1.10 * 1.1 = 1.21
```

Rounding in division

Consider the expression 7/3. The correct answer is 2.333... However, Analytics rounds the result to zero decimal places because neither operand has any decimal values.

```
7/3 = 2
```

If one or both operands have decimal places, Analytics rounds the decimal portion of the result to the larger number of decimal places in the operands:

```
7/3.00 = 2.33
7.000/3.00 = 2.333
```

Adding decimal places to control rounding

The easiest way to control rounding, and achieve a desired decimal precision, is to multiply an expression by 1, followed by the number of decimal places of precision that you want in the result. For example, multiply by 1.0000 to ensure that a result is accurate to four decimal places.

Example

Problem

In the following expressions, Analytics rounds the result to two decimal places, which is not precise enough for your requirements:

```
7.21 * 2.33 = 16.80
```

```
7.21 / 2.33 = 3.09
```

Solution

To increase the precision of the result, you multiply by 1 followed by the number of decimal places of precision that you want:

```
1.0000 * 7.21 * 2.33 = 16.7993
```

```
1.000000 * 7.21 / 2.33 = 3.094421
```

Caution

Position 1 at the beginning of an expression. If you position 1 anywhere else, the precision adjustment may not work because the precision of the first two operands to be evaluated has already caused rounding to occur:

```
7.21 * 2.33 * 1.0000 = 16.8000
```

```
7.21 / 2.33 * 1.000000 = 3.090000
```

Be careful when using parentheses

Be careful when using parentheses to specify the order of mathematical operations. If you do use parentheses, the precision adjustment may not work because the precision of the operands in parentheses has already caused rounding to occur:

```
1.0000 * (7.21 * 2.33) = 16.8000
```

Incorporating the 1 inside the parentheses can solve the problem:

```
(1.0000 * 7.21 * 2.33) = 16.7993
```

Rounding behavior in multi-operand expressions

Rounding behavior and decimal precision work the same way regardless of how many operands an expression contains. Analytics rounds the result to the larger number of decimal places in two operands as it evaluates expressions in a pairwise manner.

However, because of the way cumulative rounding works in a multi-operand expression, the decimal precision established by the first two operands to be evaluated is the precision that applies to the expression result.

Another characteristic of cumulative rounding is that the loss of decimal precision increases at each stage of a multi-operand expression.

Example of two-decimal-place precision

The table below illustrates how Analytics applies rounding as it calculates the following multi-operand expression:

```
1.1 * 1.12 * 1.123 * 1.1234 = 1.5514
```

The larger number of decimal places in the first evaluated stage of the expression is two (`1.1 * 1.12`). This two-decimal-place precision persists throughout the remainder of the multi-operand expression (indicated by the digits in red).

The **Result difference** column shows how the cumulative loss of precision increases at each successive stage of evaluation.

Analytics calculations (in evaluation order)	Analytics result (rounded)	Unrounded calculations (in evaluation order)	Unrounded result	Result difference
1.1 * 1.12	1.23	1.1 * 1.12	1.232	0.002
1.23 * 1.123	1.381	1.232 * 1.123	1.383536	0.002536
1.381 * 1.1234	1.5514	1.383536 * 1.1234	1.5542643424	0.0028643424

A closer look at precision

In the **ACL result (rounded)** column, all decimal places after the first two are imprecise when compared to the corresponding **Unrounded result**. The degree of imprecision is likely your main concern when performing numeric calculations as part of data analysis.

The rounded results are not imprecise in the context of the specific pairwise calculations that produce them. For example, `1.23 * 1.123 = 1.38129`, which rounded to three decimal places, in accordance with the rule, is `1.381`. However, `1.23` was previously rounded from `1.232`, which means that the specific pairwise calculation already contains a degree of imprecision.

Example of five-decimal-place precision

The table below illustrates how Analytics applies rounding after the addition of 1.00000 to specify five decimal places of precision:

```
1.00000 * 1.1 * 1.12 * 1.123 * 1.1234 = 1.55427
```

The larger number of decimal places in the first evaluated stage of the expression is five (`1.00000 * 1.1`). This five-decimal-place precision persists throughout the remainder of the multi-operand expression (indicated by the digits in red).

Analytics calculations (in evaluation order)	Analytics result (rounded)	Unrounded calculations (in evaluation order)	Unrounded result	Result difference
1.00000 * 1.1	1.**10000**	1.00000 * 1.1	1.**10000**	0.00000
1.10000 * 1.12	1.**23200**	1.10000 * 1.12	1.**23200**	0.00000
1.23200 * 1.123	1.**38354**	1.23200 * 1.123	1.**38353**6	0.000004
1.38354 * 1.1234	1.**55427**	1.383536 * 1.1234	1.**55426**43424	0.0000056576

Specifying the order of operations

Placement of parentheses plays an important role in determining the level of precision obtained in a calculation. Be careful when using parentheses to override the normal order of mathematical operations.

Calculating one day's interest

The scenario

You need to calculate one day's interest on $100,000 at 12% per annum.

One approach

You could first calculate the interest rate per day, and then multiply the daily rate by 100,000. However, problems with rounding arise with this approach.

```
100000 * (0.12/365) = 0.00
```

Analytics first divides 0.12 by 365, which based on the rules of rounding in Analytics results in 0.00. The true result is 0.00032876712..., but because the result is rounded to two decimal places all the subsequent digits are lost.

The rounded result is then multiplied by 100000, yielding 0.00, even though the correct answer is 32.876712...

An alternate approach

You could first calculate the total amount of interest for the year, and then divided the amount by 365. This alternate approach avoids problems with rounding.

```
100000 * 0.12/365 = 32.88
```

With the parentheses removed, the results at each stage of the calculation remain greater than 1, avoiding any decimal rounding issues, and the answer is correct to the penny (two decimal places).

Change how decimal precision works

You can use the SET MATH command to change how decimal precision works in numeric expressions. By default, Analytics uses the larger or maximum number of decimal places when evaluating two operands. Using the maximum number preserves the greatest amount of precision at each stage of an expression.

Using SET MATH in the command line or in an Analytics script changes the default behavior for the duration of the Analytics session. In the summary of options below, the different results for the expression `1.275 * 1.3` are shown. The actual unrounded result is `1.6575`.

Command	Description	Result of `1.275 * 1.3`
`SET MATH FIRST`	use the number of decimal places of the first operand in a pair of operands	1.658
`SET MATH LAST`	use the number of decimal places of the last operand in a pair of operands	1.7
`SET MATH MIN`	use the minimum number of decimal places in a pair of operands	1.7
`SET MATH MAX`	use the maximum number of decimal places in a pair of operands	1.658

Command	Description	Result of `1.275 * 1.3`
Default		
For detailed information about SET MATH, see "SET command" on page 1960.		

Controlling rounding in financial functions

Unlike other numeric operations in Analytics, financial functions are evaluated using floating-point arithmetic.

Floating-point arithmetic affects your calculations in two ways:

- Any function that returns an amount will do so to two decimal places, for example, 1250.00
- Any function that returns an interest rate will do so to eight decimal places, for example, 0.01676584 or 1.676584%

In financial functions, you often divide interest rates by constants such as 12 or 365, to create monthly or daily interest rates. For example, 0.08/365 gets the daily interest factor based on 8% per annum. The normal Analytics rules for division would result in significant rounding errors in most situations. For example, 0.08/365 will result in an interest rate of 0.00, instead of the actual rate of 0.000219178. For this reason, the normal division rules are suspended when evaluating interest rate parameters.

When the interest rate parameter in a financial function is the division of two quantities (for example, 0.08/365), then the two quantities are evaluated separately. They are then divided, maintaining the full precision of the result. Each of the two sides are still evaluated using the standard Analytics rules for rounding, so in the case of very complex calculations, this solution may still not provide total protection from rounding errors.

When the interest rate is neither a specific rate, nor the division of two simple quantities, (for example, 08/1/365) then a warning message is included in the log (and displayed in an alert dialog, if not in script mode) warning that rounding may have affected the interest rate calculation. When this occurs, you should ensure that the result returned has not been rounded excessively.

When the interest rate is calculated in a computed field, Analytics cannot determine whether the rate has been rounded, so you must ensure that the result is correct.

Avoiding overflow errors in numeric expressions

Calculation results that exceed twenty-two digits, including decimal places, can create overflow errors if the **Stop on Numeric Overflow** option is turned on. When overflow occurs, Analytics stops processing and displays ###ERR### in the view.

Overflow occurs with calculations that involve very large numbers, or numbers with many decimal places, such as financial calculations, percentages, and present values. Overflow also occurs when you use an invalid parameter such as a negative payment period in a financial function, and when you divide by zero in a calculation.

You can turn off the **Stop on Numeric Overflow** option. Analytics then continues to process commands, but it also truncates excess digits, starting from the left, which makes the result inaccurate.

You can create conditional fields to avoid division by zero. For example, in the calculation Gross pay/Hours worked, division by zero occurs if an employee works 0 hours during this particular period. To ensure that Analytics evaluates only those fields that contain values other than 0, create a conditional computed field with these values and conditions:

- **Default value:** 0
- **Condition:** Hours worked <> 0
- **Value:** Gross pay/Hours worked

The safer alternative is to leave the **Stop on Numeric Overflow** option turned on and to exercise caution with values that contain large numbers or numbers with many decimal places. Also, take care with financial function parameters and calculations where division by zero can occur.

When obviously invalid function parameters are used in a financial function, such as negative periods, the function handles the invalid parameter in one of the following ways:

- If the **Stop on Numeric Overflow** option is turned on, Analytics stops processing
- If the **Stop on Numeric Overflow** option is turned off, the function returns a value of -1.

Two common errors when using expressions

Users new to expressions in Analytics often make two common errors:

- Creating an invalid mix of data type and operation, resulting in an “Expression type mismatch” error message
- Creating a character, numeric, or datetime expression in a situation or location in which only a logical expression is valid, resulting in an “Expression type invalid” or “Logical expression is required” error message

“Expression type mismatch”

In order for an expression to be valid, the data type of the values in the expression must match the calculation or operation you are performing. For example, you cannot multiply two character fields, or divide two date fields. If you try, an “Expression type mismatch” error message is displayed and no processing occurs.

Any of the following methods can allow you to avoid this error, although not all of them may be an appropriate solution for your particular situation:

- **Keep the values the same, but change the operation**

  For example, you cannot add two time fields, so `Finish_Time + Start_Time` is not valid. However, you can subtract one time field from another, so `Finish_Time - Start_Time` is valid.

- **Keep the operation the same, but change one or more of the values**

  For example, you cannot subtract a number from a character value, so `DATE( ) - 2` is not valid, because the DATE( ) function without any parameters returns the current operating system date as a character value. However, you can subtract a number from a date value, so `TODAY( ) - 2` is valid because the TODAY( ) function returns the current operating system date as a date value.

- **Keep the operation the same, but change the data type of one or more of the values**

  For example, you cannot compare a date to a number, so `Finish_Date > 20141231` is not valid. However, you can compare a date to a date, so `Finish_Date > `20141231`` is valid. The addition of the backquotes (`) changes the numeric literal to a date literal.

  In some cases, you can use functions to convert the values or fields in an expression to a data type appropriate for the expression. For more information about conversion functions, see "Harmonizing fields" on page 851.

“Expression type invalid”, or “Logical expression is required”

You must ensure that the return value of an expression is the correct data type for the situation. In many areas of the application where an expression is required, the return value must be a particular data type. If it is not the required data type, an “Expression type invalid” or “Logical expression is required” error message is displayed and no processing occurs. Even if an expression is valid, Analytics displays an error message if the expression returns the wrong data type for the situation.

For example, the expression `Quantity_on_Hand * Unit_Cost` could be generally valid, and work as expected, providing a numeric total, if you use it to create a computed field. However, the same expression would return an error if you used it when creating a view filter or an IF statement, which require that expressions return a logical value of True or False (T or F). If you altered the expression to `Quantity_on_Hand * Unit_Cost > 5000` then it would work correctly for a filter or an IF statement.

The key point to be aware of is the location from which you open the **Expression Builder**. Clicking **Edit View Filter**  beside the Filter text box, or clicking the **If** button in various locations, requires the creation of a logical expression in the **Expression Builder**.

Using datetimes in expressions

You can use expressions to perform calculations with dates, datetimes, and times:

- **Calculate elapsed days, elapsed days and time, or elapsed time**

  For example, `` `20141231` - `20141130` `` returns `31`, the number of days between the two dates.
- **Make positive or negative adjustments to dates, datetimes, or times**

  For example, `` `20141231` - 15 `` returns `16 Dec 2014`, the date 15 days earlier.
- **Compare dates, datetimes, or times**

  For example, `` `20141231 183000` > `20141231 171500` `` returns `T` (True), because the first datetime is more recent than the second datetime.

Date and time functions

To help you work with datetime data, Analytics provides a number of date and time functions that perform a variety of useful tasks. You can use these functions when constructing datetime expressions.

The datetime functions, grouped by task, appear in the following table. You can see the same list of functions if you filter the **Functions** drop-down list in the **Expression Builder** by **Date & Time**.

Task performed	Functions
Returns the number of elapsed days (the age) between a date and a cutoff date, or the current date, or the number of elapsed days between any two dates.	AGE( )
Calculates a date, or a month end, a specified number of months before or after a date	GOMONTH( ), EOMONTH( )
Identifies the day of the week, or the month of the year, for a date	CDOW( ), CMOY( )
Returns a numeric value (1 to 7) representing the day of the week for a date	DOW( )
Extracts the date or the time from a datetime value	DATE( ), TIME( )
Extracts the day, month, year, hour, minutes, or seconds from a datetime value	DAY( ), MONTH( ), YEAR( ), HOUR( ), MINUTE( ), SECOND( )
Converts serial datetime values, or character or numeric datetime values, to regular datetime values with a Datetime data type	STOD( ), STODT( ), STOT( ), CTOD( ), CTODT( ), CTOT( )

Task performed	Functions
Returns the current operating system date, datetime, or time	TODAY( ), DATETIME( ), NOW( )

An amount of time versus a point in time

As you work with datetimes in expressions, it is important to differentiate between amounts of time, and points in time, because the difference requires that you construct different types of expressions.

Times

The time value **08:30:00** could refer to an amount of time – 8 hours and 30 minutes – or a point in time – 08:30:00 AM in the morning.

> **Note**
> The **Time Display Format** (**Tools > Options > Date and Time**) in the first example is **hh:mm:ss** and in the second example is **hh:mm:ss PM**. Both examples involve serial datetime calculations, which are explained in subsequent sections.

Amount of time

If you subtract one time from another time the result is an elapsed time, which is an amount of time.

Returns `01:15:00` (1 hour, 15 minutes, 0 seconds):

```
STOT(`T083000` - `T071500`)
```

Point in time

If you add a number to a time, or subtract a number from a time, the result is a positive or negative adjustment that creates another point in time – either later or earlier than the initial time.

Returns `07:00:00 AM`:

```
`T083000` - (1.00000000/24*1.5)
```

Dates

Amount of time

If you subtract one date from another date the result is the number of elapsed days, which is an amount of time.

Returns `31`, the number of days between the two dates:

```
`20141231` - `20141130`
```

Point in time

If you add a number to a date, or subtract a number from a date, the result is another point in time – either more recent or earlier than the initial date.

Returns `30 Nov 2014`, the date 31 days earlier:

```
`20141231` - 31
```

Valid and invalid datetime expressions

Datetime expressions encompass a number of possible combinations of datetime subtype (date, datetime, and time) and operator. Not all combinations are valid expressions. For example, you can subtract one date from another to find out the number of elapsed days, but you cannot add two dates because the operation is illogical. You can, however, add a number to a date to produce a subsequent date.

The following rules apply to datetime expressions:

- **Subtract or compare datetimes** – Any combination of date, datetime, or time values can be used in a subtraction operation or a comparison operation.

- **Add or subtract numbers and datetimes** – Whole numbers, mixed numbers, and fractional numbers can be subtracted from or added to date, datetime, or time values.
- **Add datetimes** – Date, datetime, or time values cannot be added to one another.

  If you need to add amounts of time, such as the hours worked in a week, you can use Analytics functions to extract the hour, minutes, and seconds portions of times as numeric values. You can then perform calculations on those numeric values. For more information, see "Using functions to add amounts of time" on page 819.
- **Compare datetimes and numbers** – Date, datetime, or time values cannot be compared to numbers.

The table below summarizes the combinations that are possible with datetime expressions and indicates whether each combination is valid or invalid – that is, whether it can be processed by Analytics.

> **Note**
> Even if an expression is valid, it may not always serve a useful analytical purpose. For example, Analytics will process the expression `Finish_Date > Start_Time`, but the result is always True (T) and comparing a date to a time serves no logical purpose.

	Date value	Datetime value	Time value	Number
Date value	**Valid:** Subtract Compare	**Valid:** Subtract Compare	**Valid:** Subtract Compare	**Valid:** Subtract Add
	Invalid: Add	**Invalid:** Add	**Invalid:** Add	**Invalid:** Compare
Datetime value	**Valid:** Subtract Compare	**Valid:** Subtract Compare	**Valid:** Subtract Compare	**Valid:** Subtract Add
	Invalid: Add	**Invalid:** Add	**Invalid:** Add	**Invalid:** Compare
Time value	**Valid:** Subtract Compare	**Valid:** Subtract Compare	**Valid:** Subtract Compare	**Valid:** Subtract Add
	Invalid: Add	**Invalid:** Add	**Invalid:** Add	**Invalid:** Compare

Data types returned by datetime expressions

The data type of the result returned by a datetime expression depends on the values and operator in the expression:

Datetime expression	Data type of result
Subtraction (datetime values only) Any combination of date, datetime, or time values used in a subtraction operation	**Numeric** A serial date, serial datetime, or serial time For more information, see "Serial datetimes" on page 827.
Addition or subtraction (datetime values and numbers) Whole number, mixed number, or fractional number subtracted from or added to a date, datetime, or time value	**Datetime** A date, datetime, or time subtype of the Datetime data type
Comparison (datetime values only) Any combination of date, datetime, or time values used in a comparison operation	**Logical** T (True) or F (False)

Format of datetime literals

- **Datetime values** – you can use any combination of the date, separator, and time formats listed in the table below

  The date must precede the time, and you must use a separator between the two. Valid separators are a single blank space, the letter 't', or the letter 'T'.

- **Time values** – you must specify times using the 24-hour clock

  Offsets from Coordinated Universal Time (UTC) must be prefaced by a plus sign (+) or a minus sign (-).

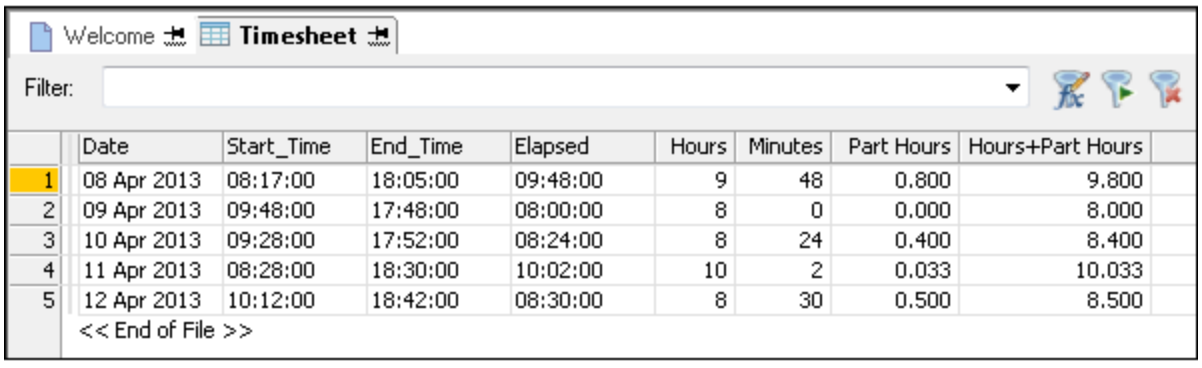
Example formats	Example literal values
YYYYMMDD	`20141231`
YYMMDD	`141231`
YYYYMMDD hhmmss	`20141231 235959`

Example formats	Example literal values
YYMMDDthhmm	`141231t2359`
YYYYMMDDThh	`20141231T23`
YYYYMMDD hhmmss+/-hhmm (UTC offset)	`20141231 235959-0500`
YYMMDD hhmm+/-hh (UTC offset)	`141231 2359+01`
thhmmss	`t235959`
Thhmm	`T2359`
Note Do not use hh alone in the main time format with data that has a UTC offset. For example, avoid: hh+hhmm. Results can be unreliable.	

Using functions to add amounts of time

You cannot directly add time values to one another in Analytics. However, you can use Analytics functions to extract the hour, minutes, and seconds portions of times as numeric values and then perform calculations on those numeric values.

The example of timesheet data below illustrates this approach using hours and minutes.

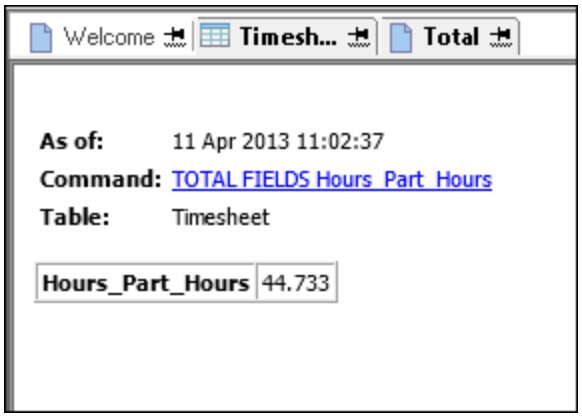


	Date	Start_Time	End_Time	Elapsed	Hours	Minutes	Part Hours	Hours+Part Hours
1	08 Apr 2013	08:17:00	18:05:00	09:48:00	9	48	0.800	9.800
2	09 Apr 2013	09:48:00	17:48:00	08:00:00	8	0	0.000	8.000
3	10 Apr 2013	09:28:00	17:52:00	08:24:00	8	24	0.400	8.400
4	11 Apr 2013	08:28:00	18:30:00	10:02:00	10	2	0.033	10.033
5	12 Apr 2013	10:12:00	18:42:00	08:30:00	8	30	0.500	8.500
	<< End of File >>							

Several computed fields are required to produce the calculations:

Computed field name	Expression	Description
Elapsed	STOT(End_Time - Start_Time)	A subtraction operation that calculates the hours worked for the day. The STOT( ) function converts the results from serial time values to regular time values.
Hours	HOUR(Elapsed)	The hour portion extracted as a numeric value from the Elapsed value.
Minutes	MINUTE(Elapsed)	The minutes portion extracted as a numeric value from the Elapsed value. (For display purposes. Not required for the calculation.)
Part Hours	MINUTE (Elapsed)/60.000	The minutes portion extracted as a numeric value from the Elapsed value, and calculated as a decimal fractional portion of 60 minutes.
Hours+Part Hours	Hours + Part_Hours	The numeric hours and part hours added together.

As a final step, you can total the **Hours+Part Hours** field to calculate the total hours for the week:



Making positive or negative adjustments to dates, datetimes, or times

You can make a positive or negative adjustment to date, datetime, or time values – for example, adding or subtracting 15 days, or adding or subtracting 3 hours.

Make a positive or negative adjustment to a date

Making a positive or negative adjustment to a date is straightforward. You add or subtract the required number of days to calculate one date based on another.

Returns `15 Jan 2015`:

```
`20141231` + 15
```

Returns `16 Dec 2014`:

```
`20141231` - 15
```

Make a positive or negative adjustment to a datetime or a time

Making a positive or negative adjustment to a datetime or a time value is somewhat more involved than making an adjustment to a date.

You cannot directly add time values to each other, or a time value to a datetime value. For example, if you try to make a positive adjustment of 3 hours using either of the following expressions, you get an error.

Returns `Expression type mismatch` error:

```
`t120000` + `t030000`
```

Returns `Expression type mismatch` error:

```
`20141231 235959` + `t030000`
```

Make a positive adjustment using the serial time equivalent

You can create a valid expression by adding the serial time equivalent of 3 hours (0.125), but manually creating such expressions can be awkward because it requires that you know what the serial time equivalent is.

Returns `15:00:00`:

```
`t120000` + 0.125
```

Returns `01 Jan 2015 02:59:59`:

```
`20141231 235959` + 0.125
```

Make a negative adjustment with a serial datetime or time result

Making a negative adjustment to a datetime or time value is easier to do, however the result is a serial datetime value or a serial time value that must be converted to a regular datetime or time value in order to be more human readable.

Returns `0.37500000000000`:

```
`t120000` - `t030000`
```

Returns `09:00:00`:

```
STOT(`t120000` - `t030000`)
```

Create a computed field to make positive or negative adjustments more easily

The following method allows you to make positive or negative adjustments to datetime or time values more easily:

1. Create a computed field that calculates the serial time equivalent of the amount of time you want to add or subtract.

   Returns `0.10416668`, the serial time equivalent of 2-1/2 hours:

   ```
   (1.00000000/24*2.5)
   ```

   You need to specify '1' with the multiple zeros in the decimal places to ensure Analytics does not round the result.

   You can change the number you multiple by to get an appropriate number of hours: `(1.00000000/24*1)`, `(1.00000000/24*8)`, `(1.00000000/24*10.25)`, and so on.

2. In the same computed field, add or subtract the calculated serial time to or from the source time or datetime value.

   Returns values in the field + 2-1/2 hours:

```
<time or datetime field> + (1.00000000/24*2.5)
```

3. If you want to add or subtract both days and time to or from a datetime value, include an appropriate number of days in the calculation.

   Returns values in the field + 2 days and 2-1/2 hours:

   ```
   <datetime field> + 2 + (1.00000000/24*2.5)
   ```

Examples of datetime expressions

The tables below provide examples of valid datetime expressions:

- "Calculate elapsed days, days and time, or time" below
- "Make positive or negative adjustments to dates, datetimes, or times" on the next page
- "Compare dates, datetimes, or times" on page 825
- "Datetime expressions that use conversion functions" on page 826

> **Note**
>
> In a number of the examples, the results are returned as serial datetimes – that is, a date, datetime, or time value represented as an integer, or a decimal fractional portion of 24 hours.
>
> You can use the STOD( ), STODT( ), and STOT( ) functions to convert serial datetime results into regular datetime values. For more information, see "Serial datetimes" on page 827.

Calculate elapsed days, days and time, or time

Expression	Result
`20141231` - `20141130`	31 The elapsed days between the two dates
Finish_Date - Start_Date	The elapsed days between Finish_Date and Start_Date values
`20141231 235959` - `20141130 114530`	31.51005787037037 The elapsed days and time between the two datetimes, with the time expressed as a serial time
STRING(INT(`20141231 235959` - `20141130 114530`),5) + " " + TIME(STOT(MOD(`20141231	31 12:14:29 The elapsed days and time between the two datetimes

Expression	Result
235959` - `20141130 114530`, 1)))	in the example above, expressed as days, hours, minutes, and seconds Assumes a current Analytics time display format of hh:mm:ss
(`20141231 235959` - `20141130 114530`) * 24	756.24138888888888 The elapsed hours between the two datetimes in the example above, expressed as hours and a decimal fractional portion of an hour
Finish_Datetime - Start_Datetime	The elapsed days and time between Finish_Datetime and Start_Datetime values, with the time expressed as a serial time
STRING(INT(Finish_DateTime - Start_DateTime), 5) + " " + TIME(STOT(MOD(Finish_DateTime - Start_DateTime, 1)))	The elapsed days and time between Finish_Datetime and Start_Datetime values, expressed as days, hours, minutes, and seconds
`T235959` - `T114530`	0.51005787037037 The elapsed time between the two times, expressed as a serial time
STOT(0.51005787037037)	12:14:29 The serial time in the example above converted to a time value, using the current Analytics time display format
STOT(`T235959` - `T114530`)	12:14:29 The elapsed time between the two times, expressed as a time using the current Analytics time display format
Finish_Time - Start_Time	The elapsed times between Finish_Time and Start_Time values, expressed as a serial time

Make positive or negative adjustments to dates, datetimes, or times

Expression	Result
Due_date + 15	The values in the Due_date field incremented by 15 days
`20141231` - 15	16 Dec 2014 The date decremented by 15 days. Assumes a current

Expression	Result
	Analytics date display format of DD MMM YYYY
`20141231 235959` + (1.00000000/24*1.5)	01 Jan 2015 01:29:59 The datetime plus 1.5 hours
`20141231 235959` - (1.00000000/24*1.5)	31 Dec 2014 22:29:59 The datetime minus 1.5 hours
STODT(`20141231 235959` - `T013000`)	31 Dec 2014 22:29:59 The datetime minus 1.5 hours
`20141231 235959` + 2 + (1.00000000/24*1.5)	03 Jan 2015 01:29:59 The datetime plus 2 days and 1.5 hours
`20141231 235959` - 2 - (1.00000000/24*1.5)	29 Dec 2014 22:29:59 The datetime minus 2 days and 1.5 hours
`t235959` + (1.00000000/24*1.5)	01:29:59 The time plus 1.5 hours
`T173000` - (1.00000000/24*1.5)	16:00:00 The time minus 1.5 hours
STOT(`T173000` - `T013000`)	16:00:00 The time minus 1.5 hours
STOT(STOT(`T173000` - `T013000`) - `T010000`)	15:00:00 The time minus 1.5 hours, minus another 1 hour

Compare dates, datetimes, or times

Expression	Result
`20141231` > `20141230`	T (True)
Due_date <= `20141231`	All values in the Due_date field on or before 31 Dec 2014
Payment_date > Due_date	All values in the Payment_date field past their due date
CTOD(DATE(Payment_timestamp, "YYYYMMDD"), "YYYYMMDD") > Due_date	All values in the Payment_timestamp field past their due date

Expression	Result
	To compare datetime and date values, the date is first extracted as a character value from the datetime values in the Payment_timestamp field, and then converted back to a date value to compare it with the due date. To ensure date formats match, identical formats are specified for the DATE( ) format parameter (the output format) and the CTOD( ) format parameter (the input format).
Login_time > `t100000`	All values in the Login_time field later than 10:00:00

Datetime expressions that use conversion functions

Expression	Result
STOT(CTOT("t120000") - CTOT("t090000"))	03:00:00 The elapsed time between the two character time values The character time values are first converted to regular time values so they can be used in a subtraction operation. The numeric serial time resulting from the subtraction operation is then converted to a regular time value.
CTOT(TIME(`20141231 125959`)) < `T235959`	T (True) The time is first extracted as a character value from the datetime value, and then converted back to a time value to compare it with 23:59:59.

Serial datetimes

Analytics uses serial datetime values to store dates, datetimes, and times, and to perform datetime calculations.

You may encounter a serial datetime value when working with datetime expressions. For example, subtraction operations that involve time values return results in the form of serial times.

What is a serial datetime?

Serial datetimes are numbers that use integers to represent dates, and a decimal fractional portion of 24 hours to represent times. The portion before the decimal point is the date, and the portion after the decimal point is the time.

Serial datetime	Regular datetime equivalent
42004	01 January 2015
42004.5000000	01 January 2015 12:00:00
0.7500000	18:00:00
42004.74618055555556	01 January 2015 17:54:30

The date portion

The date portion is the number of days that have elapsed since 01 January 1900. A serial date of '1' is equivalent to 02 January 1900. A serial date of '0' (zero) is not counted. A serial date of '42004' is equivalent to 01 January 2015.

The time portion

The time portion of serial datetimes uses the 24-hour clock. The serial time value is calculated as follows:

```
1 / 86,400 (seconds in a day) * (a specific time value expressed in seconds)
```

> **Tip**
> Another way of thinking of a serial time value is that it represents a percentage of a 24-hour day.

Regular time	Serial time
01:00:00	0.04166666666667 (1 hour, 1/24th of a 24-hour day)
08:00:00	0.3333333 (one third of a 24-hour day)
12:00:00	0.5000000 (half of a 24-hour day)
17:54:30	0.74618055555556 (17 hours, 54 minutes, 30 seconds)
18:00:00	0.7500000 (three quarters of a 24-hour day)

Converting serial datetimes to regular datetime values

Three conversion functions allow you to convert serial datetimes to regular datetime values with a Datetime data type:

- STOD( ) – "Serial to Date"
- STODT( ) – "Serial to Datetime"
- STOT( ) – "Serial to Time"

You can convert serial datetimes to make results of some datetime expressions more human-readable, or to convert a Numeric serial datetime value to a Datetime data type for use in another expression that requires a Datetime data type.

The table below shows examples of the three functions.

Expression	Results
STOD(42004)	01 Jan 2015
STODT(42004.5000000)	01 Jan 2015 12:00:00
STOT(0.7500000)	18:00:00
STODT(42004.74618055555556)	01 Jan 2015 17:54:30
STOT(`T173000` - `T093000`)	08:00:00

Converting regular datetime values to serial datetimes

Normally, there is no need to convert regular datetime values to serial datetimes. Serial datetimes are used internally by Analytics for datetime storage and computation.

If you do want to see the serial datetime value that corresponds to a regular datetime, you can use the following methods:

Regular datetime value	Conversion expression	Corresponding serial datetime
01 Jan 2015	`20150101`-`19000101`	42004
17:54:30	1.0000000000*((HOUR(`t175430`)*3600)+(MINUTE (`t175430`)*60)+SECOND(`t175430`))/86400	0.7461805556

How UTC offsets affect datetime expressions

What is UTC?

UTC is Coordinated Universal Time, the time at zero degrees longitude, which is used as a global time standard to regulate time and time zones. UTC is closely associated with Greenwich Mean Time (GMT), and for many purposes the two can considered to be identical.

What is a UTC offset?

A UTC offset is the difference in hours and minutes between a particular time zone and UTC, the time at zero degrees longitude. For example, New York is UTC-05:00, which means it is five hours behind London, which is UTC±00:00.

In Analytics, time data with a UTC offset uses this format: `hh:mm:ss±hh:mm`. For example: `23:59:59-05:00`

How Analytics processes a UTC offset

When Analytics processes datetime expressions that include local time data with a UTC offset, the UTC offset is reconciled and the expression performs the calculation on the UTC equivalent of the local time. Datetime functions also reconcile the UTC offset. For example, if an expression or function encounters the local time `23:59:59-05:00`, it actually performs the calculation on the UTC equivalent, which is `04:59:59`.

The reason that datetime expressions and functions work in this manner is that internally Analytics stores local times with UTC offsets as their UTC equivalent.

Dates can be affected

Reconciliation of the UTC offset can affect dates in datetime data if reconciliation moves the time forward or backward through the boundary of midnight. For example, the UTC equivalent of `31 Dec 2014 23:59:59-05:00` is `01 Jan 2015 04:59:59`.

UTC display option in Analytics

By default, Analytics displays local times with a UTC offset as their reconciled UTC equivalent, so you see the actual times that are being used in calculations. You also have the option of seeing the local time with the UTC offset displayed. For more information about UTC display, see "Date and Time options" on page 133.

Working with UTC-based data

If you are working with UTC-based data, you might think the results are incorrect if you do not account for the UTC reconciling process.

The examples in the table below illustrate the effect of UTC offsets on datetime expressions. To assist with the illustration, a version of the expression using the UTC equivalent is also shown. This UTC-equivalent version is not visible in Analytics when you process the expression.

Datetime expression	Expression with UTC equivalent	Result
`T235959` > `T230000` `T235959-0500` > `T230000`	n/a `T045959` > `T230000`	T (True) F (False)
`20131231 235959` + 1 `20131231 235959-0500` + 1	n/a `20140101 045959` + 1	01 Jan 2014 23:59:59 02 Jan 2014 04:59:59
CDOW(`20141231T235959`, 9) CDOW(`20141231T235959-0500`, 9)	n/a CDOW(`20150101T045959`, 9)	Wednesday Thursday
MONTH(`20141231T235959`) MONTH(`20141231T235959-0500`)	n/a MONTH(`20150101T045959`)	12 1
STOT(`T235959` - `T225959`) STOT(`T235959-0500` - `T225959-0400`)	n/a STOT(`T045959` - `T025959`)	01:00:00 02:00:00

Verifying audit data

You should always verify the completeness and validity of the data you are going to work with before performing any analytical work. Even a small amount of invalid data in a table can invalidate all your subsequent data analysis.

Operation	Description
Verify	Tests the validity of values in character, numeric, and datetime fields
Count	Counts the number of records in a table You can compare the result to a control count
Total	Totals one or more numeric fields You can compare the result to control totals

Verifying data

Verifying data checks for data validity errors in the active table. Verification ensures that data in a table conforms to the table layout and is consistent with the specified data types.

Checks performed by verifying

Verifying performs the following checks:

- **Character fields** – contain only valid, printable characters, such as letters, numbers, and symbols
- **Numeric fields** – contain only valid numeric characters, such as numbers, decimal points, and currency symbols
- **Datetime fields** – contain only valid dates, datetimes, or times

If errors are found, the relevant record number, field name, and the hexadecimal code for the invalid data are included in the output and recorded in the log.

Steps

You can verify that data conforms to the table layout, including the specified data types, and output any validity errors.

Show me how

1. Select **Data > Verify**.
2. On the **Main** tab, do one of the following:
   - Select the field(s) to verify from the **Verify Fields** list.
   - Click **Verify Fields** to select the field(s), or to create an expression.

     The order in which you select the fields is the order in which the columns appear in the results.
3. If there are records in the current view that you want to exclude from processing, enter a condition in the **If** text box, or click **If** to create an IF statement using the **Expression Builder**.

   > **Note**
   > The **If** condition is evaluated against only the records remaining in a table after any scope options have been applied (**First**, **Next**, **While**).

4. Click the **Output** tab.

5. Select the appropriate output option in the **To** panel:
   - **Screen** – Select this option to display the results in the Analytics display area.

     > **Tip**
     > You can click any linked result value in the display area to drill down to the associated record or records in the source table.

     If the output table contains a large number of records, it is faster and more useful to save the results to a file than to display the results on the screen.
   - **Print** – Select this option to send the results to the default printer.
   - **Graph** – Select this option to create a graph of the results and display it in the Analytics display area.
   - **File** – Select this option to save or append the results to a text file. The file is saved outside Analytics.

     > **Note**
     > Output options that do not apply to a particular analytical operation are disabled.

6. If you selected **File** as the output type, specify the following information in the **As** panel:
   - **File Type** – **ASCII Text File** or **Unicode Text file** (depending on which edition of Analytics you are using) is the only option. Saves the results to a new text file, or appends the results to an existing text file.
   - **Name** – Enter a file name in the **Name** text box. Or click **Name** and enter the file name, or select an existing file in the **Save** or **Save File As** dialog box to overwrite or append to the file. If Analytics prefills a file name, you can accept the prefilled name, or change it.

     You can also specify an absolute or relative file path, or navigate to a different folder, to save or append the file in a location other than the project location. For example: `C:\Results\Output.txt` or `Results\Output.txt`.
   - **Local** – Disabled and selected. Saving the file locally is the only option.
7. Click the **More** tab.
8. Select the appropriate option in the **Scope** panel:
   - **All**
   - **First**
   - **Next**
   - **While**

   Show me more

All	This option is selected by default. Leave it selected to specify that all records in the view are processed.
First	Select this option and enter a number in the text box to start processing at the first record in the view and include only the specified number of records.
Next	Select this option and enter a number in the text box to start processing at the currently selected record in the view and include only the specified number of records. The actual record number in the leftmost column must be selected, not data in the row.

While	Select this option to use a WHILE statement to limit the processing of records in the view based on a particular criterion or set of criteria. You can enter a condition in the **While** text box, or click **While** to create a WHILE statement using the **Expression Builder**. A WHILE statement allows records in the view to be processed only while the specified condition evaluates to true. As soon as the condition evaluates to false, the processing terminates, and no further records are considered. You can use the **While** option in conjunction with the **All**, **First**, or **Next** options. Record processing stops as soon as one limit is reached.

> **Note**
> The number of records specified in the **First** or **Next** options references either the physical or the indexed order of records in a table, and disregards any filtering or quick sorting applied to the view. However, results of analytical operations respect any filtering.
> If a view is quick sorted, **Next** behaves like **First**.

9. In the **Error Limit** text box, specify the maximum number of invalid records to list, or keep the default of 10.

   If the limit is reached, Analytics stops processing and outputs the invalid records found to that point.

   > **Note**
   > You can change the default error limit by selecting **Tools > Options**, **Command** tab and updating the **Error Limit** value.

10. If you selected **File** as the output type, and want to append the output results to the end of an existing text file, select **Append To Existing File**.
11. Click **OK**.
12. If the overwrite prompt appears, select the appropriate option.

Enable automatic verifying

You can configure Analytics to automatically verify data every time a table is opened. If this option is enabled, it applies to all Analytics tables.

Show me how

1. Select **Tools > Options.**
2. Click the **Numeric** tab.
3. Select **Verify Data**.
4. If you want fields that contain invalid data to appear blank, select **Blank Invalid Data**. If you do not choose this option, `###ERR###` is displayed in fields that contain invalid data.
5. Click **OK**.

Counting records

You can count all records in a table, or only those that meet a specified condition. The results are displayed in the Analytics display area.

If no condition is specified, the total number of records in the table is displayed in the status bar. If a global filter has been applied to a view, the number of records remaining in the table after the filter has been applied is displayed in the status bar.

Steps

1. Select **Analyze > Count.**
2. On the **Main** tab, you can optionally do one of the following:
   - Enter a condition in the **If** text box.
   - Click **If** to create an IF statement using the **Expression Builder**.

   The IF statement considers all records in the view and filters out those that do not meet the specified condition.
3. Click the **More** tab.
4. Select the appropriate option in the **Scope** panel:
   - **All**
   - **First**
   - **Next**
   - **While**

Show me more

All	This option is selected by default. Leave it selected to specify that all records in the view are processed.
First	Select this option and enter a number in the text box to start processing at the first record in the view and include only the specified number of records.
Next	Select this option and enter a number in the text box to start processing at the currently selected record in the view and include only the specified number of records. The actual record number in the leftmost column must be selected, not data in the row.
While	Select this option to use a WHILE statement to limit the processing of records in the view based on a particular criterion or set of criteria. You can enter a condition in the **While** text box, or click **While** to create a WHILE statement using the **Expression Builder**. A WHILE statement allows records in the view to be processed only while the specified condition evaluates to true. As soon as the condition evaluates to false, the processing terminates, and no further records are considered. You can use the **While** option in conjunction with the **All**, **First**, or **Next** options. Record processing stops as soon as one limit is reached.

> **Note**
> The number of records specified in the **First** or **Next** options references either the physical or the indexed order of records in a table, and disregards any filtering or quick sorting applied to the view. However, results of analytical operations respect any filtering.
> If a view is quick sorted, **Next** behaves like **First**.

5. Click **OK**.

Totaling fields

You can total numeric fields or expressions in the active table. Totaling finds the arithmetic sum of one or more numeric fields, and is typically used to prove the completeness and accuracy of the data, and to produce control totals. The results are displayed in the Analytics display area.

> **Tip**
> Some character fields, such as invoice numbers, may contain numbers. To total this type of data, create a computed field that uses the VALUE( ) function to convert character data to numeric data, and then total the computed field.

Steps

1. Select **Analyze > Total**.
2. On the **Main** tab, do one of the following:
   - Select the field(s) to total from the **Total Fields** list.
   - Click **Total Fields** to select the field(s), or to create an expression.

     The order in which you select the fields is the order in which the columns appear in the results.
3. If there are records in the current view that you want to exclude from processing, enter a condition in the **If** text box, or click **If** to create an IF statement using the **Expression Builder**.

   > **Note**
   > The **If** condition is evaluated against only the records remaining in a table after any scope options have been applied (**First**, **Next**, **While**).

4. Click the **More** tab.
5. Select the appropriate option in the **Scope** panel:
   - **All**
   - **First**
   - **Next**
   - **While**

Show me more

All	This option is selected by default. Leave it selected to specify that all records in the view are processed.
First	Select this option and enter a number in the text box to start processing at the first record in the view and include only the specified number of records.
Next	Select this option and enter a number in the text box to start processing at the currently selected record in the view and include only the specified number of records. The actual record number in the leftmost column must be selected, not data in the row.
While	Select this option to use a WHILE statement to limit the processing of records in the view based on a particular criterion or set of criteria. You can enter a condition in the **While** text box, or click **While** to create a WHILE statement using the **Expression Builder**. A WHILE statement allows records in the view to be processed only while the specified condition evaluates to true. As soon as the condition evaluates to false, the processing terminates, and no further records are considered. You can use the **While** option in conjunction with the **All**, **First**, or **Next** options. Record processing stops as soon as one limit is reached.

> **Note**
> The number of records specified in the **First** or **Next** options references either the physical or the indexed order of records in a table, and disregards any filtering or quick sorting applied to the view. However, results of analytical operations respect any filtering.
> If a view is quick sorted, **Next** behaves like **First**.

6. Click **OK**.

Combining data

Analytics allows you to analyze data in only one table at a time. For this reason, you may have to combine data from two or more tables into one table before performing your analysis.

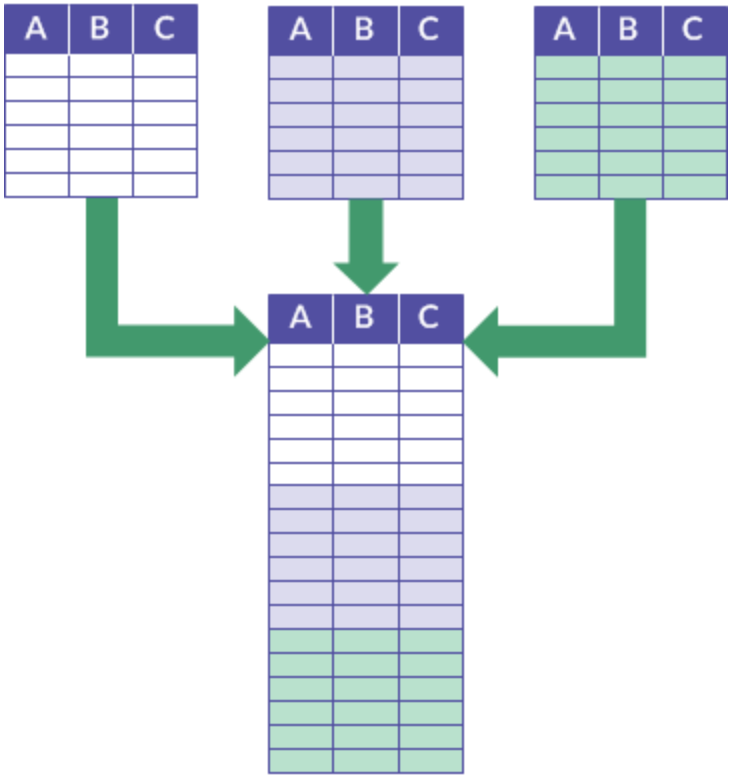
Analytics provides the following methods for combining data:

Combine records	Combine fields
◦ Append ◦ Extract/Append ◦ Merge	◦ Join ◦ Relate

The nature of the source data, or your analysis goal, dictates which method of combining data you should use. The five methods are described briefly below.

Append

When you append tables, you combine records from two or more tables into a new table that contains all the records from the appended tables. You have the option of including all the fields from the appended tables, or only the common fields.



Example

Scenario

You want to perform analysis on an entire year's worth of data but the data is spread among twelve monthly transaction tables.

Approach

You append the data from the twelve monthly tables into a single annual table containing all the data, and then perform the analysis.

Detailed information

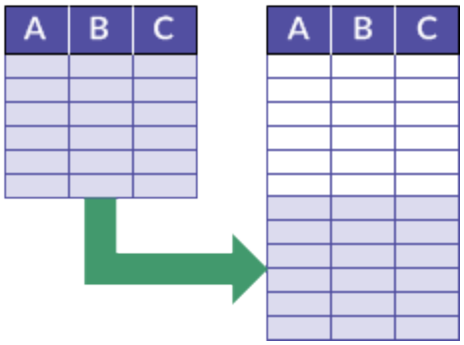
For detailed information, see "Appending tables" on page 858.

Extract/Append

When you extract and append data, you extract records from one table and append them to the end of another table. Extracting is the same as copying, and appending is the same as adding.

You also have the option of extracting a subset of the fields in a record rather than the entire record.

The table you append to (the target table) is increased in size. A new table is not created.



Example

Scenario

You want to perform analysis on an entire set of employee records but records for new employees are not yet contained in the Employee master table.

Approach

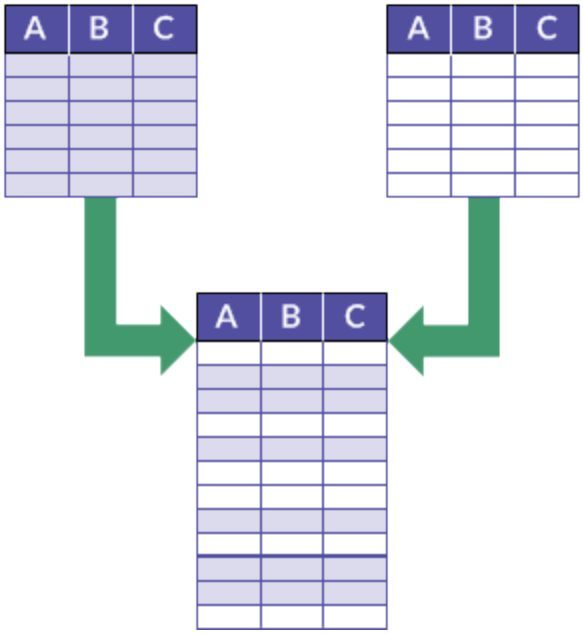
You extract the records for new employees and append them to the end of the Employee master table, and then perform the analysis.

Detailed information

For detailed information, see "Extracting and appending data" on page 870.

Merge

When you merge tables, you interfile records from two sorted tables into a new, third table, which is also sorted. Interfiling means to combine records in accordance with their existing sort order.



Example

Scenario

You want to perform analysis on an entire set of employee records but the records are split between two divisional Employee tables.

Both tables are sorted by last name and you want to avoid the overhead of resorting the records once they are combined.

Approach

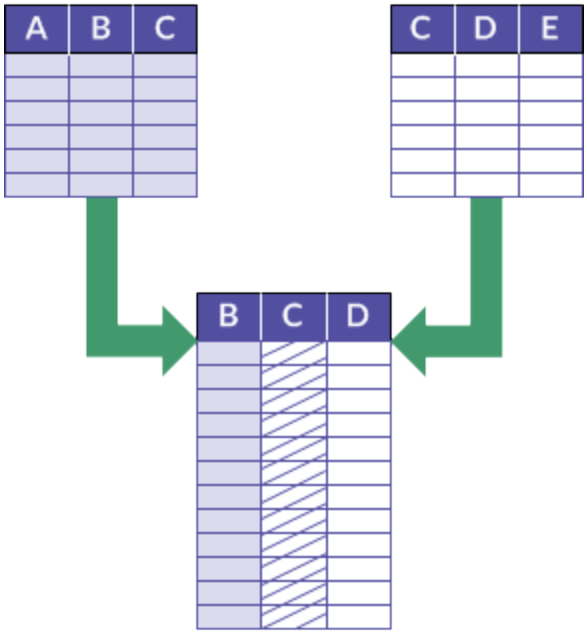
You merge the records from the two tables into a new third table. Merging preserves the sorting by last name.

Detailed information

For detailed information, see "Merging tables" on page 881.

Join

When you join tables, you use a common key field to incorporate records, or a selection of fields, from two tables into a new, third table. A common key field is an identifying field, such as Employee ID, that appears in both of the tables being joined.



Example

Scenario

You want to identify any vendors who are also employees as one way of analyzing data for possible improper payments.

Approach

You join the Vendor master table with the Employee table, using the common key field of Address.

The joined output table contains any vendors and employees with the same address.

Detailed information

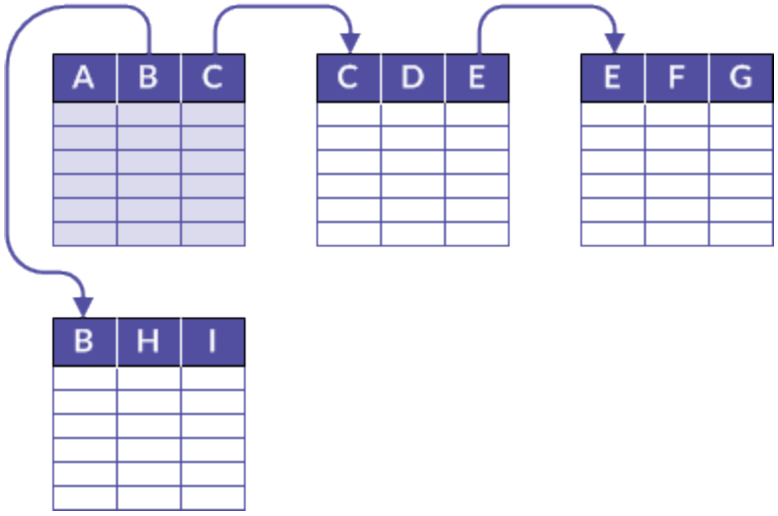
For detailed information, see "Joining tables" on page 889.

Relate

When you relate tables, you virtually join up to 18 tables. You use a common key field to relate each table pair.

Relating, or virtually joining, means to create a temporary programmatic association between tables that allows you to access the data in the tables as if it existed in a single physical table. However, no physical table is created, and you can unrelate the source tables at any point.

A common key field is an identifying field, such as Employee ID, that appears in each table pair being related. Typically, you use a different common key field for each table pair.



Example

Scenario

You want to create a sales report that contains details about the customers, and the products sold, for the month of March, but the data is spread across three tables.

Approach

You relate the Customer master table with the Orders table, and the Orders table with the Product master table, to create a temporary association of tables that contains all the information you need for the report:

- **customer name and location** – from the Customer master table
- **order details** – from the Orders table
- **product details** – from the Product master table

Detailed information

For detailed information, see "Relating tables" on page 927.

Which data combining method should I use?

There can be more than one consideration when it comes to selecting a data combining method. You can use the guidelines below as a starting point.

Use...	If...
Append	◦ You want to combine multiple tables using the least amount of labor. ◦ The records in the source tables are similar or exactly identical in structure.
Extract/Append	◦ The records or fields in the two source tables are exactly identical in structure.
Merge	◦ The records in the two source tables are exactly identical in structure. ◦ Both source tables are sorted, using an identical sort order. **Tip** Merging can be tricky to perform correctly. You can get the same result by appending, or by extracting and appending, and then sorting. If the two source tables are already sorted, merging is more efficient and can execute more quickly.
Join	◦ The records in the two source tables have different record structures. ◦ You want to include or exclude records based on matched or unmatched values in a common key field. ◦ You are doing investigative analysis that requires a physical, joined table.
Relate	◦ You want to relate, or virtually join, up to 18 tables with different record structures. ◦ You want to include or exclude records based on matched or unmatched values in common key fields. ◦ You do not need to output the combined data to a new table. ◦ You are doing informational work, like reporting, that requires only a temporary association between tables.

Use...	If...
	Tip If required, after relating tables you can perform a separate operation and extract any combination of fields from the related tables to a new physical table.

Data structure

When you combine data, the method you choose is often dependent on how the source data is structured. Data structure, or record structure, refers to the data elements contained in a record, their data type, the length of fields, and the number and order of columns.

For detailed information about data structure, see "Data structure and data format requirements" on page 848.

You may need to experiment

In some situations, it may not be immediately obvious which method to use for combining data. You may need to experiment with a small subset of the data to determine which method is best suited to the task you want to perform.

Using a small subset allows you to avoid longer processing times associated with larger tables, and may also make it easier to see patterns.

Using more than one method of combining data to achieve your goal

You may be able to address more complex data combining situations, involving multiple tables, by first using one method of combining data, and then using a second method with the output results of the first method.

Example

1. You first compile an annual transaction table by combining monthly transaction tables.
2. You then use a common key field such as Customer ID to join the annual transaction table with a master table containing data such as Customer Name.

Alternative methods for combining data

In some instances it may be easier or more practical to combine data using a method other than one of the native data combining methods in Analytics.

> **Note**
> The suitability of an alternative method depends on your particular data analysis workflow and the nature of your source data. A method may be appropriate in some cases and not others.

Alternative method	Description
Join tables using the Data Access window	Join up to ten tables when you import data into Analytics using the Data Access window. For more information, see "Joining tables in the Data Access window" on page 368.
Combine data using a third-party application	Use the native features of source applications such as Excel and Access to combine data, then import the combined data into Analytics.
Round-trip data	If you are having difficulty appending or merging data in Analytics because of inconsistencies between fields containing the same data element, you can try round-tripping the data: a. Export one of the Analytics tables to a delimited flat file. b. Export and append additional Analytics tables to the first table. c. Once you have combined all the data in a single flat file, re-import the flat file to Analytics. Delimited flat files are less stringent about data structure inconsistencies than Analytics tables: ◦ Data type and field length do not need to be the same. ◦ Only the data elements, and the number and order of columns, need to be identical. This approach may require less labor than harmonizing data and fields inside Analytics.

Data structure and data format requirements

What is data structure?

Data structure, or **record structure**, refers to:

- the fields (data elements) contained in a record
- the number and order of the fields
- the data type and length of the fields

Fields are the individual units of data in a record, such as first name, last name, address, vendor ID, and so on.

> **Tip**
> For information about using the DISPLAY command to compare the data structures of two tables, see "Comparing data structures" on page 202.

What is data format?

Data format refers to characteristics of the values contained in fields, such as:

- justification
- case
- the format of dates

Requirement for identical data structure or format

Depending on the data combining method, Analytics requires that records or fields in the tables being combined have an identical data structure. In some cases, field values must be identically formatted.

> **Tip**
> If the data structure or the data format of fields differ, you may be able to use Analytics functions to harmonize the fields. For more information, see "Harmonizing fields" on page 851.

Data combining method	Data structure requirement	Data format requirement
Append	To be directly appended to one another, fields must have identical physical names and belong to the same data category.	No requirement
Extract/Append	Entire records, all fields in a view, or all selected fields in tables being extracted and appended must be exactly identical in structure.	Datetime format must be identical
Merge	Entire records in tables being merged must be exactly identical in structure.	Datetime format must be identical
Join Relate	The common key fields in tables being joined or related must be exactly identical in structure.	Values in common key fields must be identically formatted in order for Analytics to correctly match values.

Data structure requirements in detail

Requirement for tables being combined	Append (common fields)	Extract/Append (entire record, fields in view, or selected fields)	Merge (entire record)	Join (common key field)	Relate (common key field)
Fields (data elements)	must be the same data elements	must be the same data elements	must be the same data elements	must be the same data elements	must be the same data elements
Field name	must be identical	can differ	can differ	can differ	can differ
Number and order of fields	number must be identical order can differ	must be identical	must be identical	not applicable	not applicable
Data type of corresponding fields	data category must be identical datetime subtypes must be identical automatic harmonization of character and numeric data subtypes	must be identical	must be identical	must be identical automatic harmonization of character-numeric joins	must be identical

Requirement for tables being combined	Append (common fields)	Extract/Append (entire record, fields in view, or selected fields)	Merge (entire record)	Join (common key field)	Relate (common key field)
Field length of corresponding fields	can differ	must be identical	must be identical	must be identical, or recommended to be identical, depending on data type automatic harmonization of the length of character key fields	recommended to be identical

Data format requirements in detail

Requirement for corresponding fields	Append (common fields)	Extract/Append (entire record, fields in view, or selected fields)	Merge (entire record)	Join (common key field)	Relate (common key field)
Date/Datetime format	automatic harmonization of date, datetime, or time formats	must be identical	must be identical	can differ	can differ
Justification	can differ	can differ	can differ differences affect sort order of key field	must be identical	must be identical
Case	can differ	can differ	can differ differences affect sort order of key field	must be identical	must be identical
Format of field value (Standardized hyphenation, street type abbreviation, etc.)	can differ	can differ	can differ differences affect sort order of key field	must be identical	must be identical

Harmonizing fields

In order to successfully combine tables in Analytics, you may need to first harmonize one or more fields in the two tables being combined.

What is harmonizing?

Harmonizing is the process of making the data structure of corresponding fields in separate tables identical – for example, standardizing the data type of the fields.

Harmonizing can also mean making the format of the values in two corresponding fields identical – for example, standardizing the use of hyphenation in ID numbers.

If the structure of corresponding fields, or the format of values in the fields, is not identical, jumbled data can result, the combining operation may not execute, or joins or relations may not match values correctly.

Using functions and computed fields to harmonize fields

Using Analytics functions to create computed fields is the primary technique for harmonizing fields. For example, conversion functions allow you to convert fields from one data type to another. Other functions allow you to alter field length, justification, and case, and standardize the format of values in fields.

Depending on the degree of discrepancy between two fields, you may have to use a series of functions to successfully harmonize the fields.

Once you have harmonized fields, you can combine data using any of these methods:

- **Join or relate** – for the common key field, use a harmonized field and an original key field, or two harmonized fields.
- **Append, extract and append, or merge** – create one or more harmonized fields and then extract by fields to convert the harmonized computed fields to physical fields populated with the actual computed values. Use the extracted tables with the physical fields in the data combining operation.

  For more information, see "Extracting data" on page 194 and "Extracting and appending computed fields" on page 879.

Analytics functions for harmonizing fields

Analytics functions that you can use for harmonizing fields are outlined below. For more information about using a specific function, see the "Functions overview" on page 2016.

Analytics function	Category	Purpose
STRING( )	Data type conversion (N to C)	Converts numeric data to character data.
ZONED( )		Converts numeric data to character data (ASCII zoned data format) and adds leading zeros to the data.
VALUE( )	Data type conversion (C to N)	Converts character data to numeric data.
CTOD( )	Data type conversion (C or N to D)	Converts character or numeric dates to date data.
CTODT( )		Converts character or numeric datetimes to datetime data.
CTOT( )		Converts character or numeric times to time data.
DATE( )	Data type conversion (D to C)	Converts date data to character data.
DATETIME( )		Converts datetime data to character data.
TIME( )		Converts time data to character data.
STOD( )	Data type conversion (serial N to D)	Converts serial dates to date data.
STODT( )		Converts serial datetimes to datetime data.
STOT( )		Converts serial times to time data.
SUBSTRING( )	Length adjustment	Extracts the specified portion of a string (which can be equivalent to the entire existing string). Can be used for shortening or lengthening field length. If the specified length is longer than the existing string, trailing spaces are added.
BLANKS( )		Creates a blank character string of the specified length. Can be used to add leading or trailing spaces to character data.
LTRIM( )	Length adjustment/Justification	Removes leading spaces from character data.
TRIM( )		Removes trailing spaces from character data.
ALLTRIM( )		Removes leading and trailing spaces from character data.
RJUSTIFY( )		Right justifies character data, with any trailing spaces converted to leading spaces.

Analytics function	Category	Purpose
UPPER( )	Case conversion	Converts alphabetic characters to uppercase.
LOWER( )		Converts alphabetic characters to lowercase.
PROPER( )		Converts the first character of each word to uppercase, and the rest of the word to lowercase.
INCLUDE( )	Format modification	Extracts the specified characters from a string. For example, you could extract just numbers from alphanumeric data.
REMOVE( )		Extracts the specified characters from a string, and retains the original string length by adding trailing spaces.
EXCLUDE( )		Removes the specified characters from a string. For example, you could remove numbers from alphanumeric data, or remove hyphens from “123-45-4536” and output the string “123454536”.
OMIT( )		Removes the specified characters or substrings from a string. For example, you could remove “Corporation”, “Inc.”, or “Ltd.” from vendor names.
INSERT( )		Inserts the specified characters into a string. For example, you could insert hyphens into “123454536” and output the string “123-45-4536”.
SPLIT( )		Breaks character data into segments based on separators such as spaces or commas and extracts a specified segment.
CLEAN( )		Removes invalid characters such as tabs and carriage returns, and any specified characters, from a string, and all subsequent characters, and replaces removed characters with spaces.
REPLACE( )		Replaces every instance of an existing string with a new string. For example, you could replace “Rd.” with “Road”.
DEC( )		Specifies the number of decimal places for a numeric field.

Comparison of data combining methods

The advantages and disadvantages of the different data combining methods in Analytics are outlined below.

> **Note**
>
> Appending, extracting and appending, and merging are compared because these methods combine tables with identical or similar record structures.
>
> Joining is compared with relating because these two methods combine tables with different record structures.
>
> For more information, see "Data structure and data format requirements" on page 848.

Appending, extracting and appending, and merging

Requirement/Capability	Appending	Extracting and Appending	Merging
Tables being combined must have an identical data structure	No	Yes	Yes
Resulting combined table is sorted	No Records extracted from source tables are appended as groups in the output table.	No Records extracted from the source table are appended as a group to the end of the target table.	Yes Records from both tables are inserted into a new, third table based on a sort order.
Access and analyze data from two tables	Yes	Yes	Yes
Access and analyze data from more than two tables	Yes	Not supported by a single extract and append operation. Requires multiple operations.	Not supported by a single merge operation. Requires multiple operations.
Outputs results to a new, physically separate	Yes	No	Yes

Requirement/Capability	Appending	Extracting and Appending	Merging
Analytics table			
Key fields in both tables must be: ◦ sorted ◦ the same data type ◦ the same length	Not applicable Appending does not use key fields.	Not applicable Extracting and appending does not use key fields.	Yes
Number of key fields	Not applicable Appending does not use key fields.	Not applicable Extracting and appending does not use key fields.	One or more key fields can be selected from each table.

Joining and relating

> **Note**
> If the tables that you want to combine have identical record structures, you should probably use appending, extracting and appending, or merging.

Comparison of capabilities

Capability	Joining	Relating
Use case	Good for as a preliminary step for investigative work because it outputs a permanently joined new third table.	Good for informational work because it creates a virtual table without any requirement that it be made permanent.
Access and analyze data from two tables simultaneously	Yes	Yes
Access and analyze data from more than two tables simultaneously	No Not supported by a single join operation. Requires multiple join operations.	Yes A single relation operation supports access/analysis of up to 18 tables simultaneously.
Outputs results to a new, physically separate Analytics table	Yes	No If required, you can perform a separate operation and extract any combination of fields from the related tables to a new table.
Number of key fields	One or more key fields can be	Limited to one key field per table

Capability	Joining	Relating
	selected from each table.	pair If more than one key field is required to establish an accurate relation between a table pair, create a computed field in each table to concatenate the required key fields.
Execution speed of operation	Slower The duration of the join operation varies depending on the complexity of the join, and whether or not the primary table is sorted.	Faster No actual record matching is done during the relation operation. For this reason, it takes considerably less time than joining.
Subsequent processing of a file	Faster The results of joining are stored in a flat file (.fil source data file). Flat files can be processed very quickly.	Slower Record matching between related tables is done at the time of subsequent processing, which adds to processing time.
Updateable from source data files	No Join results are sent to a new, third table with a new source data file no longer associated with the source data files involved in the join.	Yes Related tables remain associated with, and can be refreshed from, the source data files involved in the relation.
Matched primary and secondary records (1st secondary match)	Yes	Not directly supported After relating tables, use filters to isolate primary records that have matching secondary records.
Matched primary and secondary records (all secondary matches) Also called many-to-many matching	Yes	No
Unmatched primary records	Yes	Not directly supported After relating tables, use filters to isolate primary records that do not have matching secondary records.
All primary records and matched secondary records	Yes	Yes
All secondary records and matched primary records	Yes	No

Capability	Joining	Relating
All primary and secondary records, matched and unmatched	Yes	No

Comparison of requirements

Requirement	Joining	Relating
Tables being combined must have an identical data structure	No	No
Key field data types must be identical for each table pair	Varies Not required for character-numeric or numeric-character joins. Automatically harmonized by Analytics. Required for all other possibilities.	Yes
Key field lengths must be identical for each table pair	Recommended (not enforced) Lengths of two character key fields automatically harmonized by Analytics.	Recommended (not enforced)
Required disk space for processing	More Joining creates a new, third table that can be larger than both original tables combined, depending on the nature of the join.	Less Minimal disk space is required to create an index for the child table (s).
Tables must be sorted or indexed	Sort, Presort, or Index required for secondary table, optional for primary table.	Index required for child tables (created automatically when relating tables), Sort or Index optional for primary table.

Appending tables

Appending tables combines records from two or more Analytics tables into a new table. You may need to append multiple tables into a single table before you can perform analysis.

For example, you want to perform analysis on an entire year's worth of data but the data is spread among twelve monthly Excel worksheets. After importing the individual worksheets into Analytics, you can then append them to create a single annual table for analysis.

How does appending work?

Appending adds one group of records to the bottom of another group of records. The records from each source table are appended in the order in which you select the tables. The new table contains the records from the first selected table, followed by the records from the second selected table, and so on.

The source tables can have different or identical record structures, and can be sorted or unsorted.

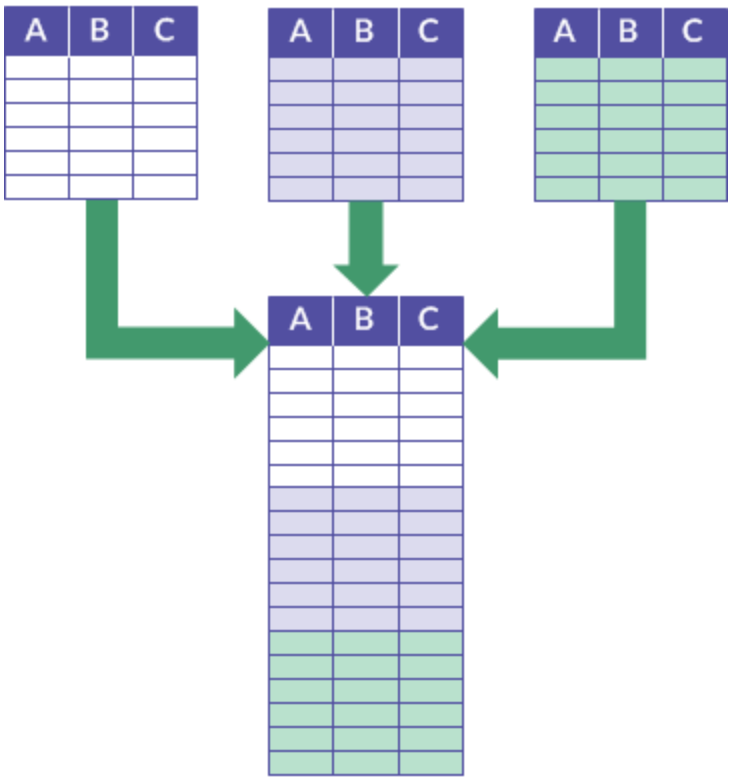
How fields are appended

When you append records from multiple tables, how the individual fields within the records are appended depends on whether the fields have identical names or unique names.

Fields with identical names

Source table fields with identical physical names and identical data categories are directly appended to one another.

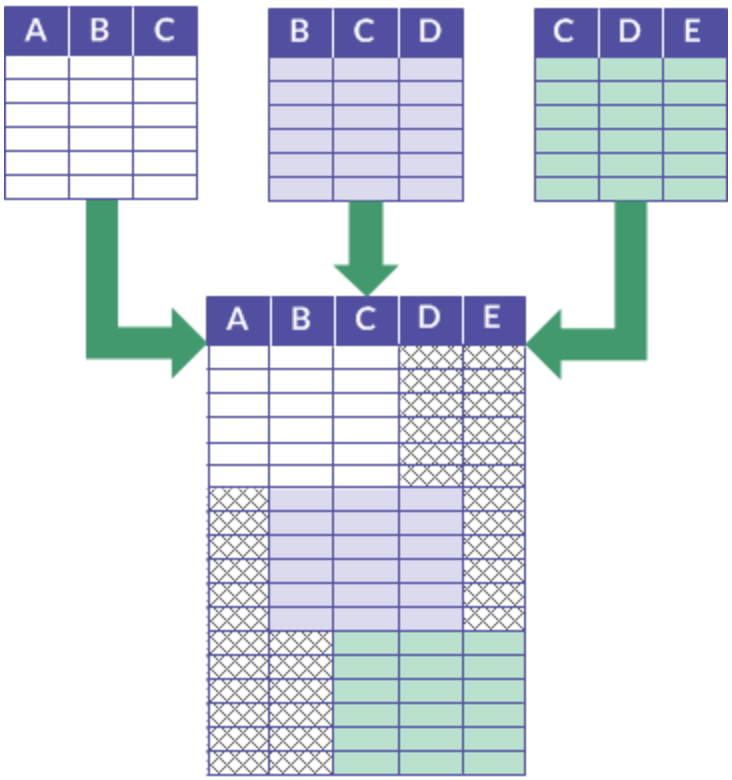
In the diagram below, fields **A**, **B**, and **C** are all directly appended.



Fields with unique names

Fields with physical names that are unique across all the source tables are added to the output table but not directly appended to any other field.

In the diagram below, fields **A** and **E** are examples of this method of appending fields.



> **Tip**
> If you want to directly append inconsistently named fields, standardize the physical names of the fields in the table layouts before appending. (Assumes that the fields belong to the same data category, or that you harmonize the data category of the fields.) For more information, see "Define a physical field" on page 717.

When to append

Use appending when you want to combine data from multiple tables with an identical or similar structure. For example, appending is a good choice for combining monthly or quarterly tables into an annual table.

> **Tip**
> A single execution of the append operation can replace multiple executions of the extract and append operation.

Example

Scenario

You want to perform analysis on an entire year's worth of data but the data is spread among twelve monthly transaction tables.

Approach

You append the data from the twelve monthly tables into a single annual table containing all the data, and then perform the analysis.

When appending is less suitable

Appending is generally not a substitute for joining or relating because it does not allow you to include or exclude records based on matched or unmatched values in a common key field. With appending, all records from each source table are included in the output table.

Appending completely dissimilar tables

You can append completely dissimilar tables – that is, two or more tables that do not have any fields in common. While not the primary intended use of appending, there may be instances in which appending dissimilar tables serves an analytical purpose.

Including all fields or only common fields

When you append tables you have two options:

- include all fields from all source tables
- include only those fields that are **common** to all source tables, meaning that they occur in every table

For fields to be considered common they must have an identical physical name, and belong to the same data category:

- Character
- Numeric
- Datetime
- Logical

Example: Appending three employee tables

You want to append three employee tables into an employee master table before performing analysis on employee data.

The three tables have three common fields, which appear in every table:

- **Employee_number**
- **First_name**
- **Last_name**

and two non-common fields, which appear in one or more tables, but not in every table:

- **Middle_name**
- **Email**

Input

The three tables being appended appear below:

Table name	Fields
Employees_central	
Employees_east	
Employees_west	

	Employee_number	First_name	Middle_name	Last_name
1	190	Sybil	Denise	Johnson
2	170	Catherine	Eleanor	Exelby
3	140	Abed	Aziz	Bhatti
	<< End of File >>			

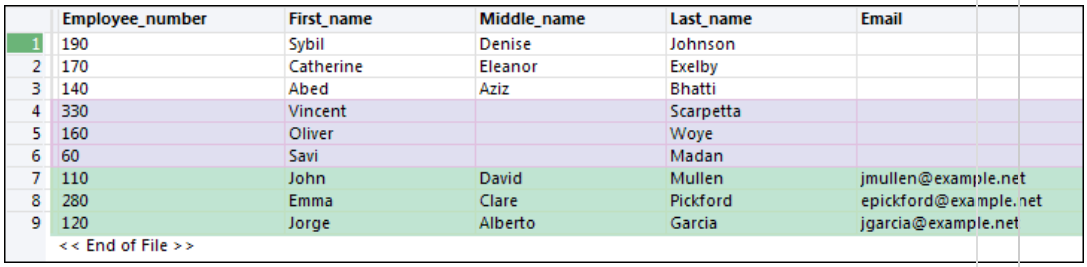
	Employee_number	First_name	Last_name
1	330	Vincent	Scarpetta
2	160	Oliver	Woye
3	60	Savi	Madan
	<< End of File >>		

	Employee_number	First_name	Middle_name	Last_name	Email
1	110	John	David	Mullen	jmullen@example.net
2	280	Emma	Clare	Pickford	epickford@example.net
3	120	Jorge	Alberto	Garcia	jgarcia@example.net
	<< End of File >>				

Output - all fields included

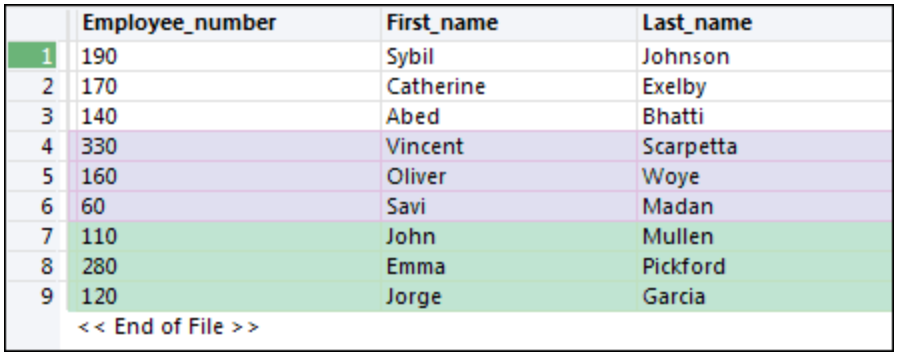
If you include all fields, the output table contains all the records and all the fields from the three appended tables.

Blank values appear in the output table where no fields exist in the source tables.

Table name	Fields
Employees_ master	

Output – only common fields included

If you include only common fields, the output table contains all the records and only the common fields from the three appended tables.

Table name	Fields
Employees_ master	

Automatic harmonization

In some situations Analytics automatically harmonizes fields in order to append them:

Data category of fields	Harmonization performed
Character	◦ Different field lengths are harmonized. ◦ Different character data types such as Custom, PCASCII, and EBCDIC are harmonized by converting the fields to the ASCII or UNICODE data type.
Numeric	◦ Different field lengths are harmonized. The fields are converted to the ACL data type. ◦ A different number of defined decimal places are harmonized. Decimal places are standardized on the greatest number of places, with trailing zeros added to numeric values where necessary. The fields are converted to the ACL data type. ◦ Different numeric data types such as Print, Float, EBCDIC, and Micro are harmonized

Data category of fields	Harmonization performed
	by converting the fields to the ACL data type.
Datetime	◦ Different date, datetime, or time formats in the source data are harmonized by converting the fields to the Analytics default formats: • YYYYMMDD • YYYYMMDD hh:mm:ss • hh:mm:ss

When automatic harmonization is not performed

Analytics does not automatically harmonize fields in the following situations. An error message appears and the append operation is not executed.

- Two fields with an identical name belong to different data categories.
- Two datetime fields with an identical name belong to different datetime subtypes (date, datetime, or time).
- Two datetime fields with an identical name are inconsistent in their use of the time zone indicator.

If you encounter one of these situations, see "User-specified harmonization" below.

User-specified harmonization

Two options in the **Append** dialog box allow you to harmonize identically named fields belonging to different data categories so that the fields can be appended without error. The options work by standardizing identically named fields on the character data category:

- **Use Character data type to harmonize common fields** – converts non-character fields to the character data category only when required for harmonization
- **Convert all fields to Character data type** – converts all non-character fields in all tables being appended to the character data category whether required for harmonization or not

Example

Scenario

You want to append two tables in which the Employee_ID field is character data in one table, and numeric data in the other.

Approach

In the **Append** dialog box, you select **Use Character data type to harmonize common fields**. The numeric Employee_ID field is converted to character data and the two fields are appended without an error.

> **Tip**
>
> If harmonizing on the character data category does not meet your needs, you may be able to manually harmonize the fields using a different approach, or redefine one or more fields. For more information, see "Harmonizing fields" on page 851 and "Define a physical field" on page 717.

Computed fields are not supported

You cannot append computed fields. When you append tables, any computed fields in the source tables are automatically excluded from the output table.

If a computed field in a source table has the same name as a physical field in another source table, an error message appears and the append operation is not executed.

> **Tip**
>
> You can append a computed field by first extracting it to convert the field to a physical field. (For more information, see "Extract and append data" on page 875.) You then use the extracted table in the append operation.
>
> Another approach is to recreate the computed field in the appended output table.

Record Note fields are not supported

You cannot append Record Note fields. When you append tables, any Record Note fields in the source tables are automatically excluded from the output table.

If a Record Note field in a source table has the same name as a physical field in another source table, an error message appears and the append operation is not executed.

A Record Note field is automatically generated by Analytics when you add a note to a record. For more information, see "Add or edit record notes" on page 179.

Additional information about appending

The table below provides additional information about appending.

Functional area	Details
Record length	If you include all fields from all source tables when appending, the record length in the

Functional area	Details
	output table can be longer than the longest record in the source tables. An error message appears if the output record length exceeds the Analytics maximum of 32 KB.
Datetime fields	For two or more datetime fields to be appended, the following conditions must be met: ◦ identical physical names ◦ identical data category (Datetime) ◦ identical data subtypes (date, datetime, or time) ◦ identical use of time zone indicator – either used, or not used, by all fields being appended **Note** You can harmonize dissimilar datetime fields by converting them to the character data category, and then append them. This approach allows you to combine the data in a single table. However, depending on the nature of the source data, you may not be able to convert the combined data back to datetime data.
Decimal places	Specific behavior governs the appending of numeric fields that include decimal places. **The Decimal setting** The append operation uses the number of decimal places defined in the **Dec** setting in the field definition in the table layout. **Note** The **Dec** setting may not be the same as the actual number of decimal places in the source data. Decimal places that exceed the **Dec** setting are undefined, and are rounded in calculations. **Inconsistent Decimal settings** If appended numeric fields have inconsistent **Dec** settings, the fields are converted to the ACL data type and automatically harmonized on the longest **Dec** setting. Any decimal places in source data files that exceed the longest **Dec** setting are **excluded** from the output table generated by the append operation. **Consistent Decimal setting** If appended numeric fields have a consistent **Dec** setting, no data type conversion or harmonization occurs. Any decimal places in source data files that exceed the **Dec** setting are **included** in the output table generated by the append operation.
Sorting	Any existing sort orders in the source tables are separately maintained in the respective record sets in the output table. Even if the records in all source tables are sorted, the output table is considered unsorted because the source records are appended as groups, without consideration of any existing sort order in other source tables. For example, if you append monthly or quarterly tables to create an annual table, any internal sorting of the monthly or quarterly data is retained. If required, you can sort the output table after performing the append operation.

<table>
<tr><th>Functional area</th><th>Details</th></tr>
<tr><td>Field order</td><td>
Common fields
Common fields in source tables do not have to be in the same order to be appended.
For example, these fields are correctly appended even though they are in a different order:
<table>
<tr><th>Table</th><th>Fields</th></tr>
<tr><td>Table 1</td><td>Last_name | First_name | Middle_name</td></tr>
<tr><td>Table 2</td><td>First_name | Middle_name | Last_name</td></tr>
</table>
The first table selected in the Append dialog box dictates the order of the fields in the output table. So in the example above, the order in the output table is:
Last_name | First_name | Middle_name
Non-common fields
Non-common fields in source tables appear in the output table in the order that they appear in the selected group of source tables.
For example, when appending these two tables:
<table>
<tr><th>Table</th><th>Fields</th></tr>
<tr><td>Table 1</td><td>Title | Last_name | First_name | Middle_name</td></tr>
<tr><td>Table 2</td><td>First_name | Middle_name | Last_name | Date_of_birth</td></tr>
</table>
the order in the output table is:
Title | Last_name | First_name | Middle_name | Date_of_birth
</td></tr>
<tr><td>Alternate Column Title</td><td>Alternate column titles in source tables appear in the output table. If more than one source table has an alternate column title for the same field, the title from the first selected table takes precedence.</td></tr>
</table>

Append tables

You can append two or more Analytics tables to create a new table that contains all the data from the source tables, or only the common fields from the source tables.

Based on the order in which you select tables in the **Append** dialog box, records from the tables are appended in vertical blocks in the new table.

Steps

1. From the Analytics main menu, select **Data > Append**.
2. In the **Append** dialog box, in the **Available Tables** list, double-click tables in the order that you want to append them in the new table.

   The tables are added to the **Selected Tables** area. A number before the table name indicates the order of the table in the **Selected Tables** area, which is also the order in which the tables are appended in the output table.
3. (Optional) Drag any of the selected tables to reorder them, and to change the order in which the tables are appended in the output table.

   > **Note**
   > Drag a table by its header, and drop it on top of another table.

4. (Optional) Click **Hide table fields**  to collapse a table field list, or **Delete selected table**  to remove a table from the **Selected Tables** area.
5. (Optional) Select **Common Fields Only** if you want the output table to include only those fields that occur in every selected table.

   If you select this option, the tables being combined must have at least one field in common.

   > **Note**
   > To be considered "common", fields must have identical physical names and belong to the same data category, or be harmonized to belong to the same data category.
   >
   > You cannot append computed fields. For more information, see "Computed fields are not supported" on page 865.

6. (Optional) Select **Add Table Name** if you want the output table to include the **Source Table** field.

For each record in the output table, the **Source Table** field identifies the table from which the record originated.

> **Tip**
> Including the names of the source tables you are appending may provide useful information when you analyze data in the output table.

7. (Optional) If you need to harmonize the data categories of identically named fields, select one of the following:
   - **Use Character data type to harmonize common fields** – converts non-character fields to the character data category only when required for harmonization

     For example, two invoice date fields that use a numeric data type in one table, and a date data type in another table, would both be converted to the character data category.
   - **Convert all fields to Character data type** – converts all non-character fields in all tables being appended to the character data category whether required for harmonization or not
8. (Optional) Select **Use Output Table** if you want the output table to open automatically.
9. In the **To** text box, specify the name of the new table, and click **OK**.

   You can also specify an absolute or relative file path, or click **Browse** to navigate to a different folder, to save the new table in a location other than the project location. For example: `C:\Data\Annual_table.fil` or `Data\Annual_table.fil`.

   If a notification about harmonizing fields appears, click **Yes**, unless you have a reason for not wanting to convert and harmonize fields. For more information, see "Automatic harmonization" on page 863.

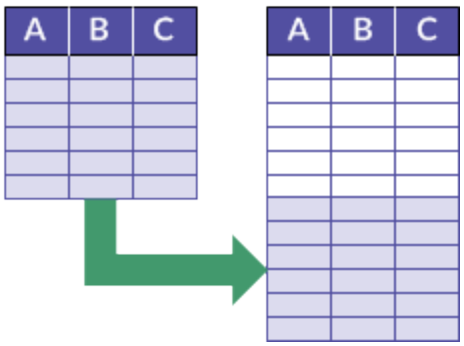
   If the overwrite prompt appears, select the appropriate option.

Extracting and appending data

Extracting and appending data allows you to extract records or fields from one Analytics table and append them as a group to the end of another Analytics table. Extracting is the same as copying, and appending is the same as adding. The two tables can be sorted or unsorted.

The table you append to (the target table) is increased in size. A new table is not created.

You can use multiple iterations of the extract and append operation to perform useful tasks such as combining monthly or quarterly tables into an annual table.



Example

Scenario

You want to perform analysis on an entire set of employee records but records for new employees are not yet contained in the Employee master table.

Approach

You extract the records for new employees and append them to the end of the Employee master table, and then perform the analysis.

> **Tip**
> A single execution of the append operation can replace multiple executions of the extract and append operation.
> For more information, see "Appending tables" on page 858.

Extract and append best practice

When you extract and append data, a best practice is to never append records to an original data file.

You should create a new target table first, by extracting the records from the original table into a new table. Then extract the records from the source table or tables, and append them to the new table.

This method preserves the original data file in the event you encounter any problems with the extract and append process.

Different options when extracting and appending data

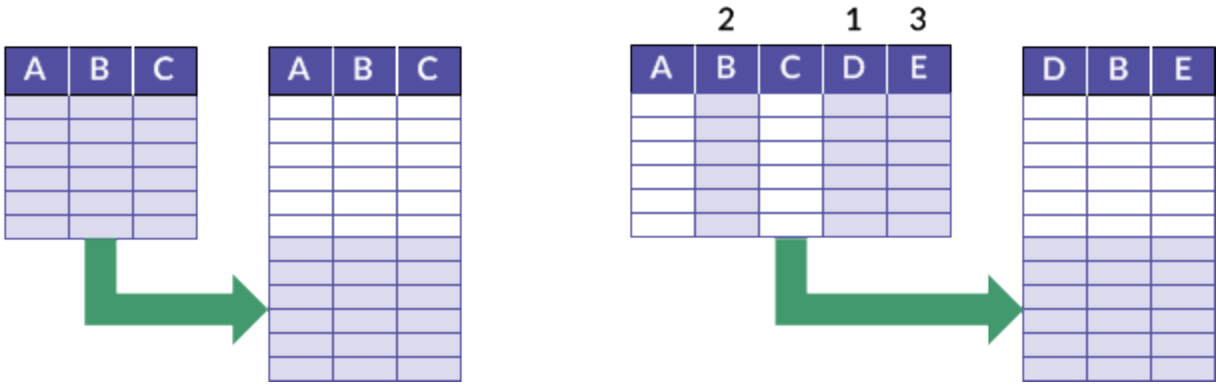
There are three different options you can choose from when extracting and appending data:

Option	Description
Extract by record	Extracts entire records. ◦ The fields are extracted in the order they appear in the table layout. ◦ The data structure of the source and target tables must be exactly identical.
Extract by view	Extracts all the fields in the current view. ◦ The fields are extracted in the order they appear in the view. ◦ The data structure of the corresponding fields in the source and target tables must be exactly identical.
Extract by fields	Extracts a selection of individual fields. ◦ The fields are extracted in the order you select them. ◦ The data structure of the corresponding fields in the source and target tables must be exactly identical.

The key difference between the options

The figure below illustrates the key difference between extracting and appending by record, and by view or by fields.

Extracting and appending by record	As shown in the two tables on the left side of the figure, with fields A, B, C: The number and the order of the fields must be identical in the source and target tables.
Extracting and appending by view or by fields	As shown in the two tables on the right side of the figure: The number and the order of the fields in the source and target tables does not have to be identical. In this situation, you tailor the view in the source table, or select the appropriate fields when you extract, to match the number and the order of fields in the target table. In the example below, you position fields in the view, or select fields, in the order: D, B, E. You omit fields A and C.



How sorting works when extracting and appending

When you extract and append, any existing sort orders in the source and the target tables are separately maintained in the respective record sets in the resulting combined table.

Even if the records in both tables are sorted, the resulting combined table is considered unsorted because the extracted records are appended as a group to the end of the target table, without consideration of any existing sort order in the target table.

For example, if you extract and append monthly or quarterly tables to create an annual table, any internal sorting of the monthly or quarterly data is retained. If required, you can sort the resulting combined table after performing one or more extract and append operations.

Extracting and appending from server tables and local tables

You can extract and append data from both server tables and local tables. Data extracted from a server table can be appended to a table on the server, or on your local computer. Data extracted from a local table can be appended only to a table on your local computer.

Requirements when extracting and appending data

When you extract and append data, the data must meet certain requirements in order for the operation to be successful. If the data does not meet the requirements, jumbled, missing, or inaccurate data can result.

If a difference in data structure at the field level is preventing successfully extracting and appending data, you may be able to harmonize the fields. For more information, see "Harmonizing fields" on page 851.

For tables with different record structures (that is, data elements are not identical), use joining or relating.

> **Tip**
>
> In some instances it may be easier or more practical to combine data outside Analytics. If you have difficulty appending data in Analytics because of inconsistencies between fields, see "Alternative methods for combining data" on page 847.

The table below summarizes the requirements for the different extract and append options.

Requirement	Extract and append by record	Extract and append by view	Extract and append by fields
Fields (data elements) The fields (data elements) in the source and target tables must be the same.	Yes	No The fields in the target table can be a subset of the fields in the source table.	No The fields in the target table can be a subset of the fields in the source table.
Order of fields Corresponding fields in the source and target tables must be in the same order in the table layouts.	Yes	No The fields in the source table view must be in the same order as the fields in the target table's table layout.	No You must select the fields in the source table in the same order as the fields in the target table's table layout.
Number of fields The number of fields in the source and target tables must be the same.	Yes	No The number of fields in the source table view must be the same as the number of fields in the target table's table layout.	No You must select a number of fields in the source table equivalent to the number of fields in the target table's table layout.
Structure of view The data structure of the source table view must be identical to target table's table layout.	No	Yes	No
Record length The overall record length in the source and target tables must be the same.	Yes	No	No
Field length	Yes	Yes	Yes

Requirement	Extract and append by record	Extract and append by view	Extract and append by fields
The length of the corresponding fields in the source and target tables must be the same.			
Field name The names of the corresponding fields in the source and target tables must be the same.	No	No	No
	Target table field names are used in the resulting combined table.		
Start position The start position of the corresponding fields in the source and target tables must be the same.	Yes	No	No
Data type The data type of the corresponding fields in the source and target tables must be the same.	Yes	Yes	Yes
Datetime format The format of dates and datetimes in corresponding fields in the source and target tables must be the same.	Yes	Yes	Yes

Extract and append data

You can extract records or fields from one Analytics table and append them as a group to the end of another Analytics table. The records or fields must have an identical structure in the two tables. The two tables can be sorted or unsorted. The resulting combined table is considered unsorted.

Steps

1. In the Navigator, open the table from which you want to extract records or fields.
2. Select **Data > Extract**.
3. On the **Main** tab, select one of the following:
   - **Record** – extracts entire records.
   - **View** – extracts all the fields in the current view.

     > **Note**
     > The number, selection, and order of the fields in the view must exactly match the number, selection, and order of fields in the target table's table layout.

   - **Fields** – extracts a selection of individual fields.
4. If you selected **Fields**, select the appropriate fields from the **Extract Fields** list.

   > **Tip**
   > You can **Ctrl+click** to select multiple non-adjacent fields, and **Shift+click** to select multiple adjacent fields.

   > **Note**
   > The number, selection, and order of the fields you select must exactly match the number, selection, and order of fields in the target table's table layout.

5. In the **To** text box, specify the name of the target table.
6. On the **More** tab:
   a. (Optional) To specify that only a subset of records are processed, select one of the options in the **Scope** panel.
   b. Click **OK**.

Extract dialog box options

The tables below provide detailed information about the options in the **Extract** dialog box.

Main tab

Options - Extract dialog box	Description
Record View Fields	Specifies the extraction method. ◦ **Record** - extracts entire records. The fields in the record are extracted in the order they appear in the table layout. ◦ **View** - extracts all the fields in the current view. The fields are extracted in the order they appear in the view. ◦ **Fields** - extracts a selection of individual fields. The fields are extracted in the order you select them. If you are extracting one or more computed fields: ◦ select **Record** to preserve the extracted fields as computed expressions ◦ select **View** or **Fields** to convert the extracted fields to physical fields of the appropriate data type and populate them with the actual computed values **Note** You cannot append computed and physical fields to each other. For more information, see "Extracting and appending computed fields" on page 879. If you want to extract data from a child table in a table relation: ◦ select **Fields**, or select **View** if the child table fields have previously been added to the view. You cannot extract child table data using the **Record** option.
Extract Fields	If you selected **Fields**, specifies the fields to extract. ◦ You can select the appropriate fields from the **Extract Fields** list. ◦ You can also click **Extract Fields** to select the appropriate fields, or to create an expression, then click **OK**. If you want to select fields from a child table in a table relation: ◦ Click **Extract Fields**. The **From Table** drop-down list in the **Selected Fields** dialog box allows you to select the appropriate child table.
If	(Optional) Allows you to create a condition to exclude records from processing. You can enter a condition in the **If** text box, or click **If** to create an IF statement using the **Expression Builder**.
To	Specifies the name and location of the target table. ◦ You can specify the name of the target table in the **To** text box. ◦ You can click **To** and specify the name of the target table, or select an existing table in the **Save** or **Save File As** dialog box as the target table. You can also specify an absolute or relative file path, or navigate to a different folder, to append data to a target table in a location other than the project location. For example: `C:\Results\GL_2011.fil` or `Results\GL_2011.fil`. Regardless of where you append data, the target table is added to the open project if it is

Options - Extract dialog box	Description
	not already in the project.
Local	If you are connected to a server table, specifies where to save the output table. ◦ **Local selected** - saves the output table to the same location as the Analytics project, or to a specified path, or location you navigate to. ◦ **Local deselected** - saves the output table to the Prefix folder on AX Server.
Use Output Table	Specifies whether the Analytics table containing the output results opens automatically upon completion of the operation.

More tab

Options - Extract dialog box	Description
Scope panel	Specifies which records in the source table are processed: ◦ **All** - (default) all records in the source table are processed. ◦ **First** - select this option and enter a number in the text box to start processing at the first record in the source table and include only the specified number of records. ◦ **Next** - select this option and enter a number in the text box to start processing at the currently selected record in the source table view and include only the specified number of records. The actual record number in the leftmost column must be selected, not data in the row. ◦ **While** - select this option to use a WHILE statement to limit the processing of records in the source table based on criteria. • You can enter a condition in the **While** text box, or click **While** to create a WHILE statement using the **Expression Builder**. • A WHILE statement allows records to be processed only while the specified condition evaluates to true. • You can use the **While** option in conjunction with the **All**, **First**, or **Next** options. **Note** The number of records specified in the **First** or **Next** options references either the physical or the indexed order of records in a table, and disregards any filtering or quick sorting applied to the view. However, results of analytical operations respect any filtering. If a view is quick sorted, **Next** behaves like **First**.
EOF (End of file processing)	(Optional) Forces the extract operation to execute one more time when the end of a table is reached. The EOF parameter is usually used if you are extracting records as part of a larger analytic process and the Extract command occurs inside a group in a script. If you are extracting records based on a comparison between sequential records, you may need to use EOF to ensure the final record in a table is extracted.
Append To Exist-	Specifies that the output results are appended (added) to the end of an existing Analytics

Options - Extract dialog box	Description
ing File	table. ◦ You can select **Append To Existing File** if you are certain the records or fields and the target table are identical in structure. ◦ You can leave **Append To Existing File** deselected if you want Analytics to compare the record lengths of the output results and the existing table. If the record lengths are not identical, the data structure is not identical, and the append will not work correctly. **Note** Leaving **Append To Existing File** deselected is recommended if you are uncertain whether the output results and the existing table have an identical data structure. For more information about appending and data structure, see "Appending output results to an existing table" on page 200.
OK	Executes the operation. If the overwrite prompt appears, select **Append**. If you are expecting the **Append** option to appear and it does not, click **No** to cancel the operation and see "Appending output results to an existing table" on page 200.

Extracting and appending computed fields

You cannot append computed and physical fields to each other. If the source or target table contains computed fields, the potential exists for a mismatch between corresponding computed and physical fields. The mismatch results in jumbled data if you force the append operation by selecting **Append to Existing File** in the **Extract** dialog box.

Extract by record, view, or fields

The option you select when extracting data – **Record**, **View**, or **Fields** – can either create or reconcile a mismatch between corresponding computed and physical fields.

- Selecting **Record** preserves extracted computed fields as computed expressions.
- Selecting **View** or **Fields** converts extracted computed fields to physical fields and populates them with the actual computed values.

You need to select the appropriate option, and in some cases perform preparatory steps (outlined below), when extracting and appending involves computed fields.

Reconciling computed and physical fields

The tables below summarize the different possible field combinations when extracting and appending, and outline the steps necessary to reconcile corresponding computed and physical fields.

Two corresponding computed fields may also use expressions that differ. In this situation, the target expression takes precedence over the source expression, which may also require that you perform some preparatory steps to ensure the combined data is valid.

Reconciling when using the Extract Record option

	Target: computed field	Target: physical field
Source: computed field	No mismatch when appending. If source and target computed fields use different expressions, the target field expression takes precedence, which may make computed values in the appended source field invalid. Review both expressions to ensure the difference is warranted. If it is, you can use	Mismatch when appending. Remedy: Use the extract **View** or extract **Fields** option to extract data from the source table.

	Target: computed field	Target: physical field
	the extract **View** or extract **Fields** option to convert both the source and target computed fields to physical fields in new tables prior to appending the new tables.	
Source: physical field	Mismatch when appending. Remedy: Use the extract **View** or extract **Fields** option to create a new target table from the existing target table. Extract the required records or fields from the source table and append them to the new target table.	No mismatch when appending.

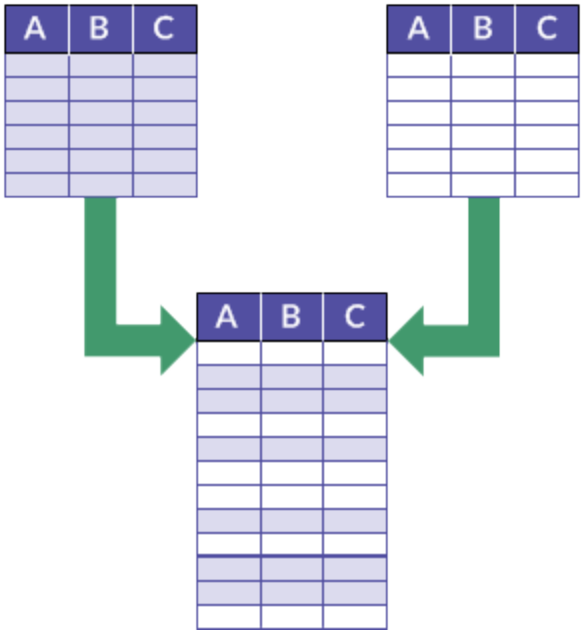
Reconciling when using the Extract View or Extract Fields option

	Target: computed field	Target: physical field
Source: computed field	Mismatch when appending. Remedy: Use the extract **Record** option to extract data from the source table.	No mismatch when appending.
Source: physical field	Mismatch when appending. Remedy: Use the extract **View** or extract **Fields** option to create a new target table from the existing target table. Extract the required records or fields from the source table and append them to the new target table.	No mismatch when appending.

Merging tables

Merging tables allows you to combine two sorted Analytics tables with identical record structures into a new third table that preserves the sort order of the original tables. Merging works by interfiling records, which means to combine records in accordance with their existing sort order.

You can use the merge operation to perform useful tasks such as combining sorted employee tables into a unified table that preserves the sort order.



Example

Scenario

You want to perform analysis on an entire set of employee records but the records are split between two divisional Employee tables.

Both tables are sorted by last name and you want to avoid the overhead of resorting the records once they are combined.

Approach

You merge the records from the two tables into a new third table. Merging preserves the sorting by last name.

Tables require an identical record structure

In order for two tables to be successfully merged, the records in both tables must be exactly identical in structure:

- The data elements, and the number and order of fields, must be identical.
- The data type of corresponding fields must be identical.
- The start position and length of corresponding fields must be identical.
- For datetime fields, the datetime format must be identical.

> **Note**
> Only character fields, and character computed fields, are displayed in the **Merge** dialog box. Fields not displayed must also have an identical data structure between the two tables.

Compare the record structure

Before you attempt to merge two tables, you can compare the corresponding fields in the tables to ensure they have an identical structure. For more information, see "Comparing data structures" on page 202.

If a difference in data structure at the field level is preventing successfully merging tables, you may be able to harmonize the corresponding fields. For more information, see "Harmonizing fields" on page 851.

> **Tip**
> In some instances it may be easier or more practical to combine data outside Analytics. If you have difficulty merging data in Analytics because of inconsistencies between fields, see "Alternative methods for combining data" on page 847.

Merge tables using a common key field

You merge tables using a common key field – that is, a data element such as employee number, vendor ID, or last name, that appears in both tables. Records in the two original tables are positioned in the merged table based on their place in the sort order used by the original tables.

Several requirements apply to the key fields in the tables you are merging:

Key field characteristic	Requirement
Data element	Must be the same. For example, both key fields are last name fields.
Sort order	Must be the same, and must be ascending.

Key field characteristic	Requirement
	Note You can use the **Presort Primary Table** option for sorting the primary key field during the merge operation. If the secondary key field is unsorted, you must first sort it in a separate sort operation before performing the merge.
Data type	Must be character.
Field type	Can be physical fields or computed fields.
Field name	Can be different.
Start position	Must be the same.
Field length	Must be the same.

Primary and secondary tables and key fields

The tables and the key fields in the merge operation are identified as **primary** and **secondary** based on the order in which you open the tables:

- **primary table** – the first table you open
- **primary key field** – the key field you choose from the primary table
- **secondary table** – the second table you open

  Opening a secondary table means associating it with a primary table and making it available for processing. Secondary tables are not opened in the View tab.
- **secondary key field** – the key field you choose from the secondary table

You are free to choose whatever primary and secondary tables and key fields you want. However, the merge will succeed only if the tables and key fields meet the requirements for merging.

For more information, see "About key fields" on page 216.

Merging tables using multiple key fields

If you want to merge two tables using more than one primary and secondary key field (that is, more than one common key), these additional requirements apply:

- All key fields must be sorted in ascending order, which means there must be a nested sorting pattern in each table.

- The order in which you select the key fields in each table must be the same as the order of the nested sorting pattern in each table.
- Both tables must use the same nested sorting pattern.

Additional information about merging

The table below provides additional information about merging.

Functional area	Details
Size of output table	The number of records in the resulting combined table is the sum of the records in the two tables being merged.
Records versus fields	You can merge entire records only.
Data type of key fields	Only character fields, or character computed fields, can be key fields. **Tip** You can use an Analytics function to convert a numeric or datetime field to a character field. For more information, see "Harmonizing fields" on page 851.
Identical key field values	When key field values are identical in primary and secondary table records, primary table records are sorted above secondary table records.
Names of corresponding fields	Corresponding fields in the primary and secondary tables do not need identical names. In the resulting combined table, the primary table field names take precedence.
Corresponding computed fields	It there are corresponding computed fields, the expression in the computed field in the primary table takes precedence over the expression in the secondary table.
Performance tip	When merging two tables of different sizes, using the larger table as the primary table requires less processing.
Indexing instead of sorting	The primary and secondary key fields can be indexed in ascending order instead of sorted. Indexing can provide performance benefits over sorting. Applying an index to the secondary table can only be performed from the command line or in a script.
Scope parameters	The If, While, First, and Next parameters that limit which records are processed apply only to the primary table.
Location of tables	In order to be merged, tables must be in the same Analytics project. Server tables must be on the same server, and must be accessed using the same server profile. You cannot merge a local table with a server table.

Merge tables

Using a common key field from each table, you can merge two sorted Analytics tables with identical record structures into a new third table that uses the same sort order as the original tables.

> **Note**
> To successfully merge tables, the data in both tables must be exactly identical in structure. For more information, see "Merging tables" on page 881.

Steps

1. In the Navigator, open the primary table, and right-click the secondary table and select **Open as Secondary**.

   The primary and secondary table icons update with the numbers 1 and 2 to indicate their relation to each other  .
2. Select **Data > Merge**.
3. On the **Main** tab:
   a. Select the primary key field from the **Primary Keys** list.
   b. Select the secondary key field from the **Secondary Keys** list.
4. In the **To** text box, specify the name of the new, merged table.
5. On the **More** tab:
   a. (Optional) To specify that only a subset of records are processed, select one of the options in the **Scope** panel.
   b. Click **OK**.

Merge dialog box options

The tables below provide detailed information about the options in the **Merge** dialog box.

Main tab

Options - Merge dialog box	Description
Secondary Table	An alternate method for selecting the secondary table.
Primary Keys	Specifies the common key field to use to merge the two tables.

Options - Merge dialog box	Description
Secondary Keys	◦ You can select the common key field directly in the **Primary Keys** and **Secondary Keys** lists. ◦ You can also click **Primary Keys** or **Secondary Keys** to open the **Selected Fields** dialog box where you can select the common key field, or create an expression on the primary key. **Key field guidelines:** ◦ **Data type** - Both key fields must be character fields. ◦ **Data structure** - The following elements must be exactly identical for both key fields: • start position • field length ◦ **Sorting** - Both key fields must be sorted in ascending order. ◦ **Multiple key fields** - If required, the common key can include more than one key field per table. For more information, see "Merging tables using multiple key fields" on page 883.
Presort Primary Table	Sorts the primary table by the key field, or fields. ◦ If the key field or fields are already appropriately sorted or indexed, you can deselect **Presort**. ◦ Presorting increases the length of time it takes to merge tables, so you should use this feature only if you need to. ◦ The secondary key field must already be sorted or indexed in **ascending** order because there is no Presort option for the secondary key field.
Local	If you are connected to a server table, specifies where to save the merged table. ◦ **Local selected** - saves the output table to the same location as the Analytics project, or to a specified path, or location you navigate to. ◦ **Local deselected** - saves the output table to the Prefix folder on AX Server.
Use Output Table	Specifies whether the Analytics table containing the output results opens automatically upon completion of the operation.
If	(Optional) Allows you to create a condition to exclude records from processing. ◦ You can enter a condition in the **If** text box, or click **If** to create an IF statement using the **Expression Builder**. ◦ The condition applies only to the primary table.
To	Specifies the name and location of the output table. ◦ **To save the output table to the Analytics project folder** - enter only the table name. ◦ **To save the output table in a location other than the project folder** - specify an absolute or relative file path, or click **To** and navigate to a different folder. For example: `C:\Results\Output.fil` or `Results\Output.fil`. Regardless of where you save the output table, it is added to the open project if it is not already in the project. If Analytics prefills a table name, you can accept the prefilled name, or change it.

More tab

Options - Merge dialog box	Description
Scope panel	Specifies which records in the primary table are processed: ◦ **All** - (default) all records in the primary table are processed. ◦ **First** - select this option and enter a number in the text box to start processing at the first record in the primary table and include only the specified number of records. ◦ **Next** - select this option and enter a number in the text box to start processing at the currently selected record in the primary table view and include only the specified number of records. The actual record number in the leftmost column must be selected, not data in the row. ◦ **While** - select this option to use a WHILE statement to limit the processing of records in the primary table based on criteria. • You can enter a condition in the **While** text box, or click **While** to create a WHILE statement using the **Expression Builder**. • A WHILE statement allows records to be processed only while the specified condition evaluates to true. • You can use the **While** option in conjunction with the **All**, **First**, or **Next** options. **Note** The number of records specified in the **First** or **Next** options references either the physical or the indexed order of records in a table, and disregards any filtering or quick sorting applied to the view. However, results of analytical operations respect any filtering. If a view is quick sorted, **Next** behaves like **First**.
Append To Existing File	Specifies that the output results are appended (added) to the end of an existing Analytics table. **Note** Leaving **Append To Existing File** deselected is recommended if you are uncertain whether the output results and the existing table have an identical data structure. For more information about appending and data structure, see "Appending output results to an existing table" on page 200.
OK	Executes the operation. If the overwrite prompt appears, select the appropriate option. If you are expecting the **Append** option to appear and it does not, click **No** to cancel the operation and see "Appending output results to an existing table" on page 200.

Common uses of joining or relating

A common use of joining or relating is to match records in a transaction table with those in a master table.

Example

- Compare employee T&E claims to employee T&E limits to identify any employees who have exceeded their reimbursement limit.
- Compare transaction authorizations against a segregation of duties list to identify any employees who are circumventing internal controls.
- Compare customer account balances to customer credit limits to identify any customers who have exceeded their credit limit.

Another common use of joining or relating is to compare the contents of two files.

Example

- Compare employee records to a vendor listing to check for employees who are doubling as vendors.
- Compare physician billing records to insurance claims to ensure that claim amounts are accurate.

Which table should be the primary or parent table?

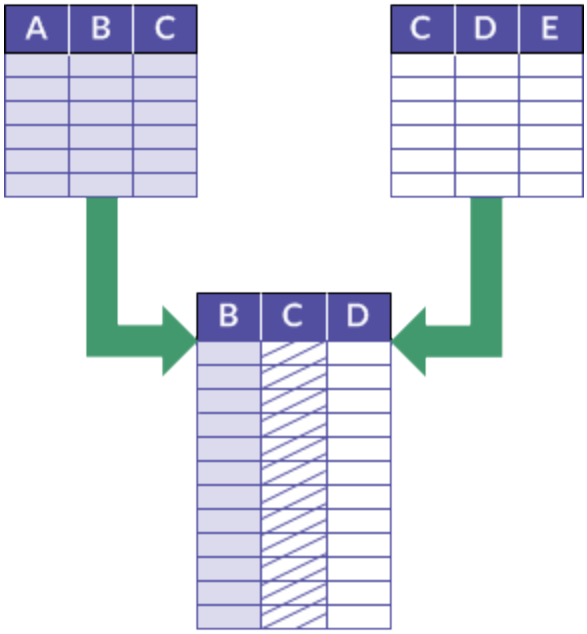
Because Analytics joins are predominantly many-to-one joins, and Analytics relations are always many-to-one relations, the transaction table should usually be the primary or parent table, and the master table the secondary or child table. Multiple transactions ('many'), such as a number of days of T&E claims, can be joined or related to a single master record ('one'), such as a T&E limit.

Carefully identify your primary/parent and secondary/child tables because results can differ if you reverse the order. As a general rule, if you want to analyze all the records in a table, that table should be the primary or parent table.

Joining tables

Joining tables allows you to combine two Analytics tables with different record structures into a new third table. You can select any combination of fields from the two original tables to be included in the new table.

Record structures are different if they have one or more fields (data elements) that differ. Joining is a good choice for investigative work that requires a permanently joined set of data as a starting point for analysis.



Example

Scenario

You want to identify any vendors who are also employees as one way of analyzing data for possible improper payments.

Approach

You join the Vendor master table with the Employee table, using the common key field of Address.

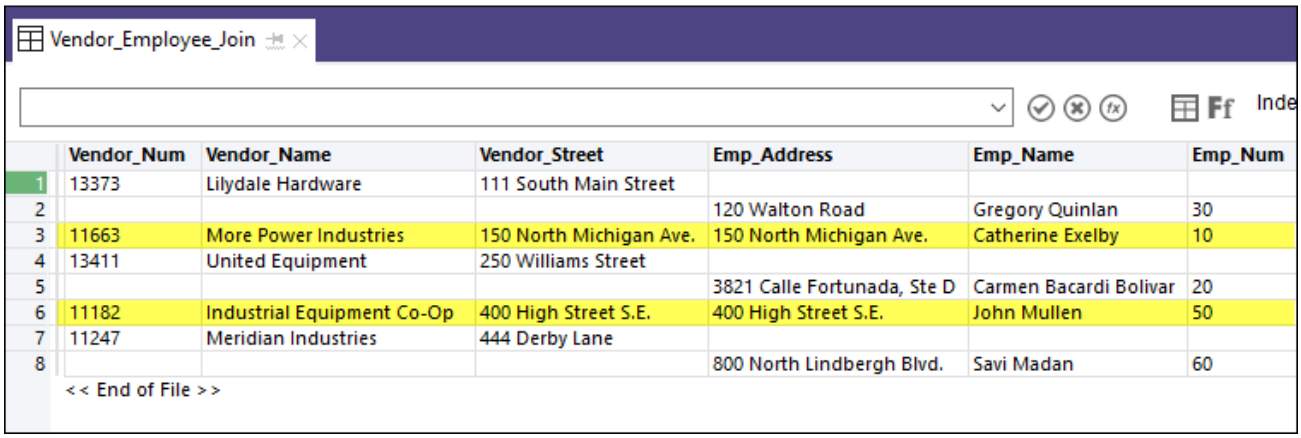
The joined output table contains any vendors and employees with the same address.

> **Note**
>
> For information about joining tables in the Data Access window as part of the data import process, see "Joining tables in the Data Access window" on page 368.
>
> This topic is about joining Analytics tables once they are in Analytics.

Join tables using a common key field

You join tables using a common key field – that is, a data element such as employee number, vendor ID, or address, that appears in both tables. When identical values exist in the two key fields, the result is a match that joins individual records from the separate tables.

In the example below, a vendor master table and an employee master table are joined using the address field in each table as a common key (**Vendor_Street**, and **Emp_Address**). The output table contains two joined records. In the example, the unjoined records from each table are also included in the output table, which is an option you can select.


Vendor_Employee_Join

	Vendor_Num	Vendor_Name	Vendor_Street	Emp_Address	Emp_Name	Emp_Num
1	13373	Lilydale Hardware	111 South Main Street			
2				120 Walton Road	Gregory Quinlan	30
3	11663	More Power Industries	150 North Michigan Ave.	150 North Michigan Ave.	Catherine Exelby	10
4	13411	United Equipment	250 Williams Street			
5				3821 Calle Fortunada, Ste D	Carmen Bacardi Bolivar	20
6	11182	Industrial Equipment Co-Op	400 High Street S.E.	400 High Street S.E.	John Mullen	50
7	11247	Meridian Industries	444 Derby Lane			
8				800 North Lindbergh Blvd.	Savi Madan	60

<< End of File >>

Joining tables using similar or nearly identical key field values

An Analytics fuzzy join use fuzzy matching of key field values to combine two Analytics tables into a new third table. In most respects, a fuzzy join is like a regular Analytics join. The main difference is that in addition to joining records based on exact matching of key field values, a fuzzy join can join records based on approximate matching.

For more information, see "Fuzzy join" on page 913.

Key field requirements

Several requirements apply to the key fields in the tables you are joining.

Show me more

Key field characteristic	Requirement
Data element	Must be the same. For example, both key fields are employee number fields.
Data type	Can be any data type, but key fields must be the same data type as each other. For example, two character fields. The exception is character-numeric and numeric-character data type joins, which Analytics automatically harmonizes. For more information, see "Automatic harmonization when joining tables" on page 925. Datetime subtypes (date, datetime, and time) can only be joined to the same subtype.
Field type	Can be physical fields or computed fields.
Field name	Can be different.
Start position	Can be different.
Field length	◦ **Character key fields** - must be the same. Analytics automatically harmonizes the length of character key fields. For more information, see "Automatic harmonization when joining tables" on page 925. ◦ **Numeric key fields** - recommended that they be the same. ◦ **Datetime key fields** - can be different.
Justification and case in character fields	Must be the same.

Primary and secondary tables and key fields

The tables and key fields in the join operation are identified as **primary** and **secondary** based on the order in which you open the tables:

- **primary table** - the first table you open
- **primary key field** - the key field you choose from the primary table
- **secondary table** - the second table you open

  Opening a secondary table means associating it with a primary table and making it available for processing. Secondary tables are not opened in the View tab.

- **secondary key field** - the key field you choose from the secondary table

You are free to choose whatever primary and secondary tables and key fields you want. However, the join will succeed only if the key fields meet the "Key field requirements" on the previous page.

For more information, see "About key fields" on page 216.

Matched versus unmatched records

When you work with joins, you need to consider both the matched and the unmatched records:

- **Matched records** – primary and secondary records are matched if they have identical values in the primary and secondary key fields.

> **Note**
> Depending on the join type you select, duplicate occurrences of matching secondary key values may be left unjoined. For more information, see "Why are some secondary table records missing from the joined output table?" on the facing page

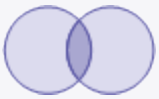
- **Unmatched records** – primary and secondary records are unmatched if they do not have identical values in the primary and secondary key fields.

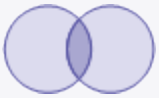
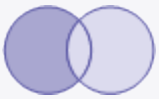
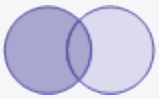
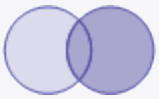
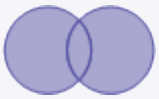
Which records are included in the joined table?

Matched and unmatched key field values, and the type of join you perform, determine which records from the two original tables are included in the new, joined table.

Types of joins

Analytics supports six different types of joins, summarized below. For specific examples, see "Examples of join types" on page 904.

Join type	Records included in joined table			
	Matched primary records	Unmatched primary records	Matched secondary records	Unmatched secondary records
Matched primary and secondary (1st secondary match) 	✔		✔ **Not included:** duplicate occurrences of matching secondary key values	
Matched primary and secondary (all secondary matches)	✔		✔ **Included and joined:** duplicate	

Join type	Records included in joined table			
	Matched primary records	Unmatched primary records	Matched secondary records	Unmatched secondary records
			occurrences of matching secondary key values	
Unmatched primary 		✔		
All primary and matched secondary 	✔	✔	✔ **Not included:** duplicate occurrences of matching secondary key values	
All secondary and matched primary 	✔		✔ **Included but not joined:** duplicate occurrences of matching secondary key values	✔ **Included:** duplicate occurrences of unmatched secondary key values
All primary and secondary 	✔	✔	✔ **Included but not joined:** duplicate occurrences of matching secondary key values	✔ **Included:** duplicate occurrences of unmatched secondary key values

Why are some secondary table records missing from the joined output table?

Five of the six Analytics join types do not join duplicate occurrences of matching secondary key values. Duplicate occurrences of matching primary key values are joined, but they are all joined to the first occurrence of the matching secondary key value. These join types are broadly known as many-to-one joins.

To join all matching secondary key values do one of the following:

- **Reverse the tables** – In the join, reverse the primary and secondary tables. This method is appropriate if the values in the key field in the original primary table are unique. If there are duplicate occurrences of primary key values in both tables, this method may not produce the results you want.
- **Use many-to-many joining** – Use the **Matched primary and secondary (all secondary matches)** join type.

Many-to-one joins and the many-to-many join

You may see Analytics joins referred to as many-to-one joins, and one of the six join types referred to as the many-to-many join. These terms are useful as a way of broadly describing the behavior of Analytics joins. However, the terms are generalizations, and do not entirely represent join behavior.

Analytics many-to-one joins

With one exception, the join types available to you in Analytics are many-to-one joins. They also function as one-to-one joins if all values in the primary key field are unique.

Show me more

In an Analytics many-to-one join:

- **joined** – duplicate occurrences of a matching primary key value are all joined to the first occurrence of the matching secondary key value

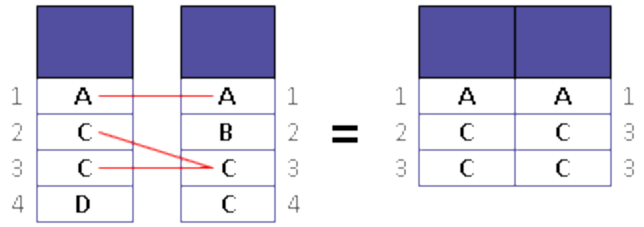
  The duplicate primary key matches, and the first secondary key match, are included in the joined table.

- **not joined** – duplicate occurrences of a matching secondary key value are left unjoined

  The duplicate secondary key matches are excluded from the joined output table, unless you select a join type that includes all secondary records. If you include all secondary records, duplicate secondary key matches appear in the joined output table as unjoined records.

Many-to-one join

In the example below, both occurrences of the primary key value 'C' are joined in the output table, but only the first occurrence of the secondary key value 'C' is joined.



> **Tip**
>
> If the key field values are unique in one of the tables you are joining, make that table the secondary table. For example, if you are joining a transactional table with a master table, make the master table the secondary table.
>
> Structuring the join in this manner ensures that all the matching records are joined and included in the output table.

The Analytics many-to-many join

One Analytics join type – **Matched primary and secondary (all secondary matches)** – includes all matching primary and secondary records. This join type is also known as the many-to-many join.

The many-to-many join also functions as a one-to-many join if all values in the primary key field are unique.

Show me more

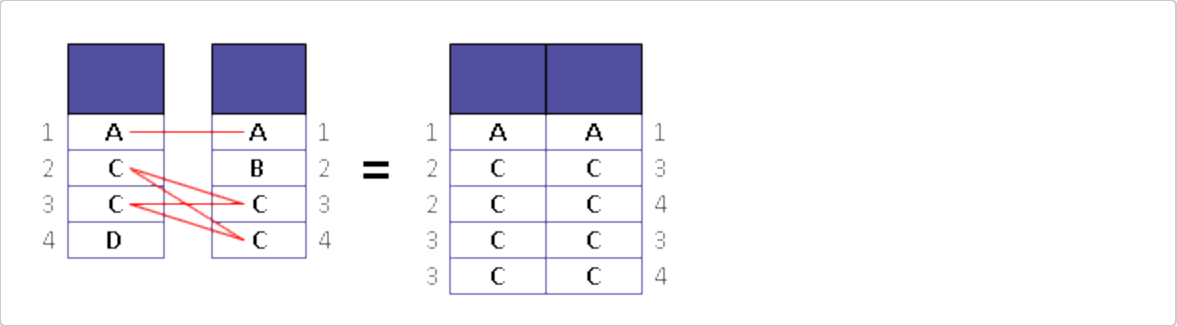
In an Analytics many-to-many join:

- **joined** – all occurrences of a matching primary key value are joined to all occurrences of the matching secondary key value

  The duplicate primary key matches, and the duplicate secondary key matches, are all joined and included in the output table.
- **not joined** – no duplicate occurrences of a matching secondary key value are left unjoined

Many-to-many join

In the example below, both occurrences of the primary key value 'C' are joined in the output table, and both occurrences of the secondary key value 'C' are also joined.



> **Tip**
>
> If you are uncertain whether duplicate matches exist in the secondary key, choose the many-to-many join type. It ensures that you do not exclude any records that should be joined.
>
> If you intentionally want to exclude duplicate secondary key matches, then do not choose the many-to-many join type.

How Analytics joins and SQL joins differ

There is an important difference between the Analytics joins you perform using the **Join** dialog box, and the SQL joins available when importing data into Analytics using the Data Access window:

- **Analytics joins** – duplicate secondary key matching values are left unjoined (with the exception of the many-to-many join)
- **SQL joins** – duplicate secondary key matching values are all joined, regardless of the join type you choose

  For more information about SQL joins in the Data Access window, see "Joining tables in the Data Access window" on page 368.

> **Note**
>
> Analytics uses the term "many-to-many join" in a manner unique to Analytics. It is not the same as a SQL many-to-many join.

Sorting of joined tables

The combined table resulting from a join is sorted in ascending order on the primary key field, assuming you **Presort** the primary key field while performing the join, or the primary table already uses this sort order.

If you do not sort or presort by the primary key field, the resulting joined table uses the existing sort order of the primary table.

Additional information about sorting and joining

- It is not mandatory that the primary table be sorted, but processing time increases significantly if the primary table is completely unsorted, or sorted in descending order.
- If you perform a join using a partially sorted primary table key field – for example, joining on account code when the table is sorted by month and then by account code – the increase in processing time is not as significant.
- When joining, the **Presort** option exists for both the primary and the secondary tables.
- The primary and secondary key fields can be indexed instead of sorted. The secondary key field must be indexed in ascending order. Applying an index to the secondary table can only be performed from the command line or in a script.

Additional information about joining

The table below provides additional information about joining.

Functional area	Details
Unmatched records and missing field values	If you include unmatched primary or unmatched secondary records in a join, for the missing field values, Analytics displays a blank in character and datetime fields, a zero in numeric fields, and “F” in logical fields.
Duplicates or blanks in secondary table key field	If duplicates or missing values in a secondary table key field render subsequent analysis invalid, pre-processing the secondary table to remove the duplicates and/or blanks may be a solution in some circumstances.
Partial matching	Partial matching of key field values is not supported. To be matched, values must be 100% identical. For example: ◦ **matched** - AB-123, AB-123 ◦ **not matched** - AB-123, 123 **Note** Partial matching is supported by the Analytics "Fuzzy join" on page 913.
Identical key field length not enforced	With the exception of character key fields, Analytics does not enforce identical lengths for the primary and secondary key fields when joining tables. It is recommended that you always use identical lengths for numeric key fields, manually harmonizing the lengths prior to performing the join, if necessary. Results derived from joining using numeric key fields of different lengths are not reliable. Datetime key fields can be different lengths because Analytics, when performing operations involving dates, datetimes, or times, uses an internal Analytics datetime format.

Functional area	Details
Harmonizing justification and case	When you join tables using character key fields, justification and case must be the same: ◦ Both key fields must have the same justification. Use the LTRIM( ) function to remove leading blanks from the key fields. ◦ Both key fields must be in the same case – UPPER, lower, or Proper. To harmonize the case, use the UPPER( ), LOWER( ), or PROPER( ) function.
Count of records not included in a join	Depending on the type of join you perform, records in the primary and/or secondary tables may not be included in the joined table. The command log displays the number of primary records not included (***<n>* records bypassed**), but not the number of bypassed secondary records.
Conditional expressions and scope options used in join operation	In many-to-one joins, the If, While, First, and Next parameters that limit which records are processed apply only to the primary table. In many-to-many joins, If and While expressions can also reference the secondary table.
Identical field names in tables being joined	If the primary and secondary key fields, or any other included fields, have identical names, Analytics adds '2' to the end of the secondary field name in the layout for the output table. For example, 'vendor_ID' becomes 'vendor_ID2' (or 'vendor_ID3', and so on, until Analytics finds a name that does not conflict with any other field names in the output table). The alternate column titles in the view for the output table continue to display the identical names unaltered.
Table unavailable as secondary table	A table is unavailable to select as the secondary table in a join if it is currently related to the primary/parent table as a child table. To avoid this restriction, you can create a copy of the primary/parent table layout, or the child table layout, and join using the copied layout, or you can delete the relation.
Restrictions on location of tables being joined	To be joined, tables must be in the same Analytics project. Server tables must be on the same server, and must be accessed using the same server profile. You cannot join a local table with a server table.
Size of joined table	Depending on the type of join performed, the number of records in the resulting combined table can be greater than, equal to, or less than the sum of the records in the two tables being joined.
Joining UTC-based and non-UTC data	A UTC-based and a non-UTC datetime key field can be used to join two tables. (UTC is Coordinated Universal Time, the time at zero degrees longitude.) When performing operations involving datetimes or times, Analytics uses an internal Analytics datetime format, so the following two datetimes are interpreted as identical, and constitute a match: ◦ **UTC-based** - 31/12/2014 10:30:15-05:00 ◦ **non-UTC** - 31/12/2014 15:30:15 You should exercise caution if you mix UTC-based and non-UTC time data in an Analytics operation. Although Analytics will match the two time values above, it may not make logical sense to do so, because one value references a time zone, and the other value does not. For more information about UTC, see "Date and Time options" on page 133.

Join tables

Using a common key field from each table, you can join two Analytics tables with different record structures into a new third table. The fields contained in the third table can be any combination of fields from the two original tables.

> **Note**
> Carefully identify the primary and secondary tables in a join because results can differ if you reverse the order. For more information, see "Common uses of joining or relating" on page 888.

Steps

1. In the Navigator, open the primary table, and right-click the secondary table and select **Open as Secondary**.

   The primary and secondary table icons update with the numbers 1 and 2 to indicate their relation to each other  .
2. Select **Data > Join**.
3. On the **Main** tab:
   a. Select the join type.

      Join types are explained below.
   b. Select the primary key field from the **Primary Keys** list.
   c. Select the secondary key field from the **Secondary Keys** list.
   d. Select the fields to include in the joined table from the **Primary Fields** and **Secondary Fields** lists.

      > **Note**
      > You must explicitly select the primary and secondary key fields if you want to include them in the joined table.

      > **Tip**
      > You can **Ctrl+click** to select multiple non-adjacent fields, and **Shift+click** to select multiple adjacent fields.
4. In the **To** text box, specify the name of the new, joined table.
5. (Optional) On the **More** tab:
   a. If you want to process only a subset of records, select one of the options in the **Scope** panel.
   b. If you want to append (add) the output results to the end of an existing Analytics table, select **Append To Existing File**.

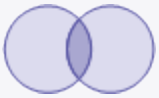
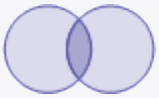
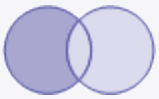
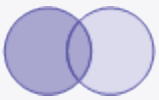
6. Click **OK**.

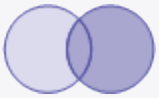
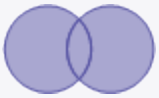
   The new, joined table is output.

Join dialog box options

The tables below provide detailed information about the options in the **Join** dialog box.

Main tab

Options – Join dialog box	Description
Join Types	Specifies which Analytics join type to use. For detailed information, see "Which records are included in the joined table?" on page 892
Matched primary and secondary (1st secondary match) 	The joined output table contains: ◦ all matched primary records and the first matched secondary record
Matched primary and secondary (all secondary matches) 	The joined output table contains: ◦ all matched primary records and all matched secondary records ◦ one record for each match between the primary and secondary tables
Unmatched primary 	The joined output table contains: ◦ unmatched primary records
All primary and matched secondary 	The joined output table contains: ◦ all primary records (matched and unmatched) and the first matched secondary record
All secondary and matched primary	The joined output table contains: ◦ all secondary records (matched and unmatched) and all matched primary records Only the first instance of any duplicate secondary matches is joined to a primary

Options – Join dialog box	Description
	record.
All primary and secondary 	The joined output table contains: ◦ all primary and all secondary records, matched and unmatched Only the first instance of any duplicate secondary matches is joined to a primary record.
Secondary Table	An alternate method for selecting the secondary table.
Primary Keys Secondary Keys	Specifies the common key field to use to join the two tables. ◦ You can select the common key field directly in the **Primary Keys** and **Secondary Keys** lists. ◦ You can also click **Primary Keys** or **Secondary Keys** to open the **Selected Fields** dialog box where you can select the common key field, or create an expression on the primary key. **Key field guidelines:** ◦ **Data type** – The key fields can be any data type, but they must be the same data type as each other. The one exception is that character and numeric key fields can be joined to each other. ◦ **Datetime subtypes** – Datetime subtypes (date, datetime, and time) can only be joined to the same subtype. ◦ **Length** • It is recommended that numeric key fields be the same length. • If character key fields are not the same length, they are automatically harmonized. • Datetime key fields do not need to be the same length. ◦ **Names and start positions** – Key field names and start positions can be different, but they must describe the same data element. ◦ **Multiple key fields** – If required, the common key can include more than one key field per table. For more information, see "Using multiple key fields" on page 941.
Primary Fields Secondary Fields	Specifies the fields to include in the joined table. ◦ You can select fields directly in the **Primary Fields** and **Secondary Fields** lists. ◦ You can also click **Primary Fields** or **Secondary Fields** to open the **Selected Fields** dialog box where you can select the fields, or create an expression on one or more primary fields. ◦ The order in which you select primary and secondary fields dictates the field order in the resulting joined table. As a group, the primary fields appear before the secondary fields in the joined table.
Presort Primary Table Presort Secondary Table	Sorts the primary or secondary tables by their key field, or fields. ◦ If one or both key fields are already appropriately sorted or indexed, you can deselect **Presort**.

Options - Join dialog box	Description
	◦ Presorting increases the length of time it takes to join tables, so you should use this feature only if you need to. ◦ The secondary key field must be sorted or indexed in **ascending** order.
Local	If you are connected to a server table, specifies where to save the joined table. ◦ **Local selected** - saves the output table to the same location as the Analytics project, or to a specified path, or location you navigate to. ◦ **Local deselected** - saves the output table to the Prefix folder on AX Server.
Use Output Table	Specifies whether the Analytics table containing the output results opens automatically upon completion of the operation.
If	(Optional) Allows you to create a condition to exclude records from processing. ◦ You can enter a condition in the **If** text box, or click **If** to create an IF statement using the **Expression Builder**. ◦ For most Analytics join types, the condition can reference the primary table only. ◦ For the join type **Matched primary and secondary (all secondary matches)**, the condition can reference either the primary or the secondary table, or both. **Note** To access the secondary table fields in the **Expression Builder**, select the secondary table in the **From Table** drop-down list. The **If** condition is evaluated against only the records remaining in a table after any scope options have been applied (**First**, **Next**, **While**).
To	Specifies the name and location of the output table. ◦ **To save the output table to the Analytics project folder** - enter only the table name. ◦ **To save the output table in a location other than the project folder** - specify an absolute or relative file path, or click **To** and navigate to a different folder. For example: `C:\Results\Output.fil` or `Results\Output.fil`. Regardless of where you save the output table, it is added to the open project if it is not already in the project. If Analytics prefills a table name, you can accept the prefilled name, or change it.

More tab

Options - Join dialog box	Description
Scope panel	Specifies which records in the primary table are processed: ◦ **All** - (default) all records in the primary table are processed. ◦ **First** - select this option and enter a number in the text box to start processing at the first record in the primary table and include only the specified number of records. ◦ **Next** - select this option and enter a number in the text box to start processing at

Options - Join dialog box	Description
	the currently selected record in the primary table view and include only the specified number of records. The actual record number in the leftmost column must be selected, not data in the row. ◦ **While** - select this option to use a WHILE statement to limit the processing of records in the primary table based on criteria. • You can enter a condition in the **While** text box, or click **While** to create a WHILE statement using the **Expression Builder**. • A WHILE statement allows records to be processed only while the specified condition evaluates to true. • You can use the **While** option in conjunction with the **All**, **First**, or **Next** options. Record processing stops as soon as one limit is reached. **Note** The number of records specified in the **First** or **Next** options references either the physical or the indexed order of records in a table, and disregards any filtering or quick sorting applied to the view. However, results of analytical operations respect any filtering. If a view is quick sorted, **Next** behaves like **First**.
Append To Existing File	Specifies that the output results are appended (added) to the end of an existing Analytics table. **Note** Leaving **Append To Existing File** deselected is recommended if you are uncertain whether the output results and the existing table have an identical data structure. For more information about appending and data structure, see "Appending output results to an existing table" on page 200.
OK	Executes the operation. ◦ If you are joining character and numeric key fields to each other, or joining character key fields of different lengths, a message appears stating that Analytics will attempt to harmonize the fields. ◦ If the overwrite prompt appears, select the appropriate option. If you are expecting the **Append** option to appear and it does not, click **No** to cancel the operation and see "Appending output results to an existing table" on page 200.

Examples of join types

Examples follow that illustrate the six types of joins in Analytics. The examples show how you can use different join types to get exactly the information you want in the joined output table.

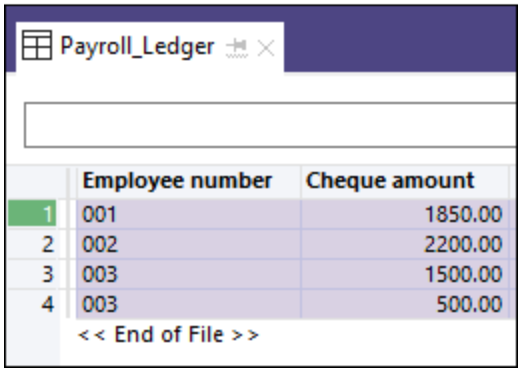
For a summary view of the six join types, see "Which records are included in the joined table?" on page 892

To use fuzzy matching of key field values to join two Analytics tables, see "Fuzzy join" on page 913.

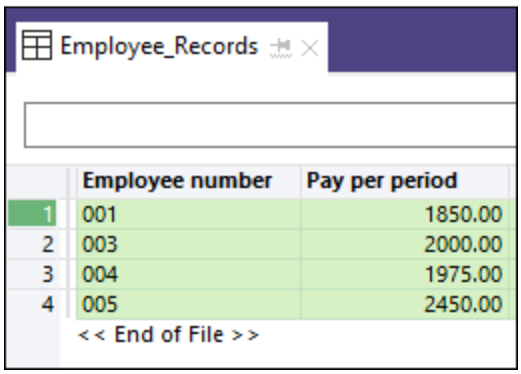
Sample data

The first five examples use the sample data shown below.

Primary table



Secondary table



Sample data and example details

Payroll Ledger table	Contains a single pay period and includes all payroll disbursements. One employee, 003, received two checks.

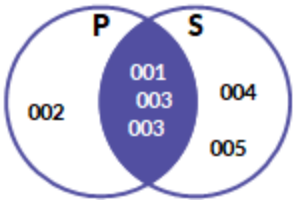
(primary)	
Employee Records table (secondary)	Maintained by the human resources department. Employee records consist of a complete list of valid employees and the amount that they are paid each period. One employee, 002, does not appear in the table.
Join	In the examples that follow, the Payroll Ledger table is joined with the Employee Records table using the common key field of Employee number. All five examples are many-to-one joins.
Objective	In each example, the objective is to test for payroll irregularities.

Matched primary and secondary records (1st secondary match)

Example

Test - You want to verify that employees were paid correctly.

Approach - You use a join type that creates one output record for every record in the Payroll Ledger table (**P**) that has a match in the Employee Records table (**S**).



Output table - Contains all employees who have been paid and who are also listed in the Employee Records table.

	Cheque amount	Employee number	Employee number	Pay per period
1	1850.00	001	001	1850.00
2	1500.00	003	003	2000.00
3	500.00	003	003	2000.00
	<< End of File >>			

Note that the two employee 003 records in the primary table are joined to the same employee 003 record in the secondary table.

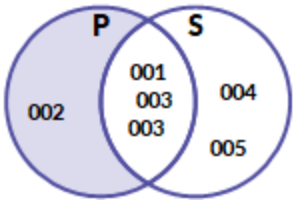
Analysis - In the output table, you can compare **Cheque amount** to **Pay per period** to verify that an employee was paid correctly. Even though employee 003 received two cheques, the total amount of $2000 is correct.

Unmatched primary records

Example

Test - You want to find out if someone who is not listed as an employee was paid.

Approach - You use a join type that creates one output record for every record in the Payroll Ledger table (**P**) that does not have a match in the Employee Records table (**S**).



Output table - Contains people who have been paid but who are not listed in the Employee Records table.

	Cheque amount	Employee number
1	2200.00	002
	<< End of File >>	

Analysis - Any record in the output table requires follow-up.

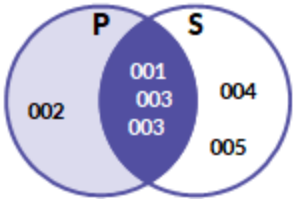
Perhaps employee 002 is a valid employee who has been omitted from the Employee Records table in error, or is listed with the wrong employee number. Or employee 002 may be a phantom employee created as part of a fraud.

All primary records and matched secondary records

Example

Test - You want to verify amounts for all checks that were issued.

Approach - You use a join type that creates one output record for every record in the Payroll Ledger table (**P**) whether or not it has a match in the Employee Records table (**S**).



Output table - Contains a complete list of people who have been paid.

	Cheque amount	Employee number	Employee number	Pay per period
1	1850.00	001	001	1850.00
2	2200.00	002		0.00
3	1500.00	003	003	2000.00
4	500.00	003	003	2000.00
	<< End of File >>			

Analysis - In the output table, you can compare **Cheque amount** to **Pay per period** to verify that an employee was paid correctly. You can see that employee 002 was paid $2200 but according to the **Pay per period** field was not supposed to be paid anything.

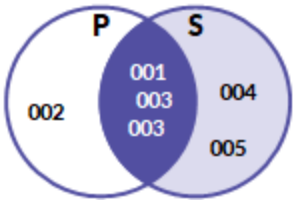
> **Note**
> Analytics fills missing secondary fields for unmatched primary records with blanks or zeros.

All secondary records and matched primary records

Example

Test - You want to verify that all employees listed in the Employee Records table were paid.

Approach - You use a join type that creates one output record for every record in the Employee Records table (**S**) whether or not it has a match in the Payroll Ledger table (**P**).



Output table - Contains a complete list of all employees and what they were paid.

	Cheque amount	Employee number	Employee number	Pay per period
1	1850.00	001	001	1850.00
2	1500.00	003	003	2000.00
3	500.00	003	003	2000.00
4	0.00		004	1975.00
5	0.00		005	2450.00
	<< End of File >>			

Analysis – In the output table, you can compare **Cheque amount** to **Pay per period** to verify that an employee was paid, and paid correctly. You can see that employees 004 and 005 were not paid at all.

> **Note**
> Analytics fills missing primary fields for unmatched secondary records with blanks or zeros.

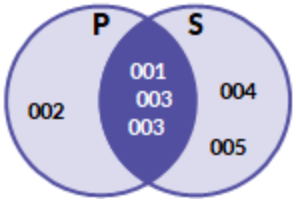
All primary and secondary records, matched and unmatched

Example

Test – You want to examine all payroll and employee data.

Approach – You use a join type that creates:

- one output record for every record in the Payroll Ledger table (**P**) that has a match in the Employee Records table (**S**)
- one output record for every unmatched record in either table



Output table – Contains all payroll and employee data, whether it is matched or unmatched.

	Cheque amount	Employee number	Employee number	Pay per period
1	1850.00	001	001	1850.00
2	2200.00	002		0.00
3	1500.00	003	003	2000.00
4	500.00	003	003	2000.00
5	0.00		004	1975.00
6	0.00		005	2450.00

<< End of File >>

Analysis – In the output table, you can compare **Cheque amount** to **Pay per period**:

- to verify that an employee was paid, and paid correctly
- to identify people who were paid but who are not listed in the Employee Records table
- to identify employees who were not paid

> **Note**
> Analytics fills missing fields for unmatched records with blanks or zeros.

Matched primary and secondary records (all secondary matches)

The example uses the sample data shown below.

Primary table

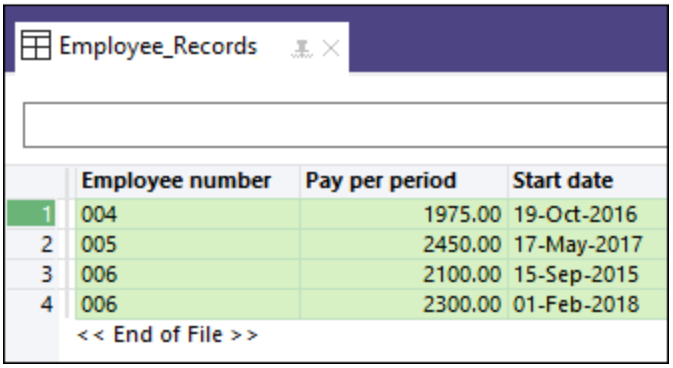
Payroll_Ledger

	Employee number	Cheque amount	Pay date
1	004	1975.00	15-Jan-2018
2	004	1975.00	31-Jan-2018
3	004	1975.00	15-Feb-2018
4	004	1975.00	28-Feb-2018
5	005	2450.00	15-Jan-2018
6	005	2450.00	31-Jan-2018
7	005	2450.00	15-Feb-2018
8	005	2450.00	28-Feb-2018
9	006	2100.00	15-Jan-2018
10	006	2100.00	31-Jan-2018
11	006	2300.00	15-Feb-2018
12	006	2300.00	28-Feb-2018

<< End of File >>

Sample data and example details

Secondary table

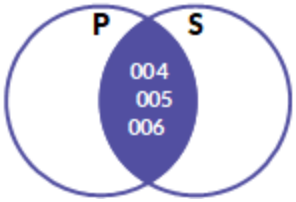


Payroll Ledger table (primary)	The complete Payroll Ledger table has all pay periods and all payroll disbursements for 2018. The example uses January and February disbursements.
Employee Records table (secondary)	Maintained by the human resources department. The Employee Records table contains: ◦ a complete list of valid employees ◦ each employee's pay per period ◦ each employee's start date ◦ any employee's start date in a new position Two records exist for employee 006: ◦ start date ◦ data of a promotion and a pay raise
Join	In the example that follows, the Payroll Ledger table is joined with the Employee Records table using the common key field of Employee number. The example is a many-to-many join.
Objective	In the example, the objective is to test for payroll irregularities.

Example

Test - You want to verify that employees were paid correctly.

Approach - You use a join type that creates one output record for every match between records in the Payroll Ledger table (**P**) and the Employee Records table (**S**).



> **Note**
>
> Because both source tables in the join contain multiple occurrences of matching key values, you need to use the join type that includes all secondary matches to ensure that you capture all relevant data and derive accurate results.

Output table – For each pay date, contains all employees who have been paid and who are also listed in the Employee Records table.

	Pay date	Cheque amount	Employee number	Employee number	Pay per period	Start date
1	15-Jan-2018	1975.00	004	004	1975.00	19-Oct-2016
2	31-Jan-2018	1975.00	004	004	1975.00	19-Oct-2016
3	15-Feb-2018	1975.00	004	004	1975.00	19-Oct-2016
4	28-Feb-2018	1975.00	004	004	1975.00	19-Oct-2016
5	15-Jan-2018	2450.00	005	005	2450.00	17-May-2017
6	31-Jan-2018	2450.00	005	005	2450.00	17-May-2017
7	15-Feb-2018	2450.00	005	005	2450.00	17-May-2017
8	28-Feb-2018	2450.00	005	005	2450.00	17-May-2017
9	15-Jan-2018	2100.00	006	006	2100.00	15-Sep-2015
10	15-Jan-2018	2100.00	006	006	2300.00	01-Feb-2018
11	31-Jan-2018	2100.00	006	006	2100.00	15-Sep-2015
12	31-Jan-2018	2100.00	006	006	2300.00	01-Feb-2018
13	15-Feb-2018	2300.00	006	006	2100.00	15-Sep-2015
14	15-Feb-2018	2300.00	006	006	2300.00	01-Feb-2018
15	28-Feb-2018	2300.00	006	006	2100.00	15-Sep-2015
16	28-Feb-2018	2300.00	006	006	2300.00	01-Feb-2018
	<< End of File >>					

Analysis – In the output table, you can compare **Cheque amount** to **Pay per period** to verify that an employee was paid correctly for each **Pay date**.

Because you used the join type that includes all secondary matches (the Analytics many-to-many join), the $200 increase in **Cheque amount** received by employee 006 starting on 15 February is explained by the matching employee record that shows a $200 raise beginning 01 February.

Remove redundant joined records – Depending on the nature of the data being joined, a many-to-many join can create redundant joined records. In the example above, some of the employee 006 joined records contained invalid **Pay date-Start date** combinations. You can use a filter to remove the invalid combinations and make the output table easier to read:

```
Emp_Num="004" OR Emp_Num="005" OR (Emp_Num="006" AND Pay_date <=
`20180131` AND Start_date = `20150915`) OR (Emp_Num="006" AND Pay_date >
`20180131` AND Start_date = `20180201`)
```

	Pay date	Cheque amount	Employee number	Employee number	Pay per period	Start date
1	15-Jan-2018	1975.00	004	004	1975.00	19-Oct-2016
2	31-Jan-2018	1975.00	004	004	1975.00	19-Oct-2016
3	15-Feb-2018	1975.00	004	004	1975.00	19-Oct-2016
4	28-Feb-2018	1975.00	004	004	1975.00	19-Oct-2016
5	15-Jan-2018	2450.00	005	005	2450.00	17-May-2017
6	31-Jan-2018	2450.00	005	005	2450.00	17-May-2017
7	15-Feb-2018	2450.00	005	005	2450.00	17-May-2017
8	28-Feb-2018	2450.00	005	005	2450.00	17-May-2017
9	15-Jan-2018	2100.00	006	006	2100.00	15-Sep-2015
11	31-Jan-2018	2100.00	006	006	2100.00	15-Sep-2015
14	15-Feb-2018	2300.00	006	006	2300.00	01-Feb-2018
16	28-Feb-2018	2300.00	006	006	2300.00	01-Feb-2018

<< End of File >>

Fuzzy join

An Analytics fuzzy join use fuzzy matching of key field values to combine two Analytics tables into a new third table. In most respects, a fuzzy join is like a regular Analytics join (see "Joining tables" on page 889). The main difference is that in addition to joining records based on exact matching of key field values, a fuzzy join can join records based on approximate matching.

Fuzzy joining is useful when primary and secondary keys contain the same kind of data, but in slightly different form. Or the data in the keys has slight irregularities, such as typos, that might prevent an exact match.

Example

Scenario

You want to identify any vendors who are also employees as one way of analyzing data for possible improper payments.

Approach

You join the Vendor master table with the Employee table, using the address field in each table as a common key (**Vendor_Street**, and **Emp_Address**). However, the form of the address data in the key fields varies slightly, so you use a fuzzy join instead of a regular join.

A look at some of the data

Without significant data cleansing and harmonization work, the primary and secondary key values shown below would not be joined by a regular Analytics join, even though they are very likely matching addresses.

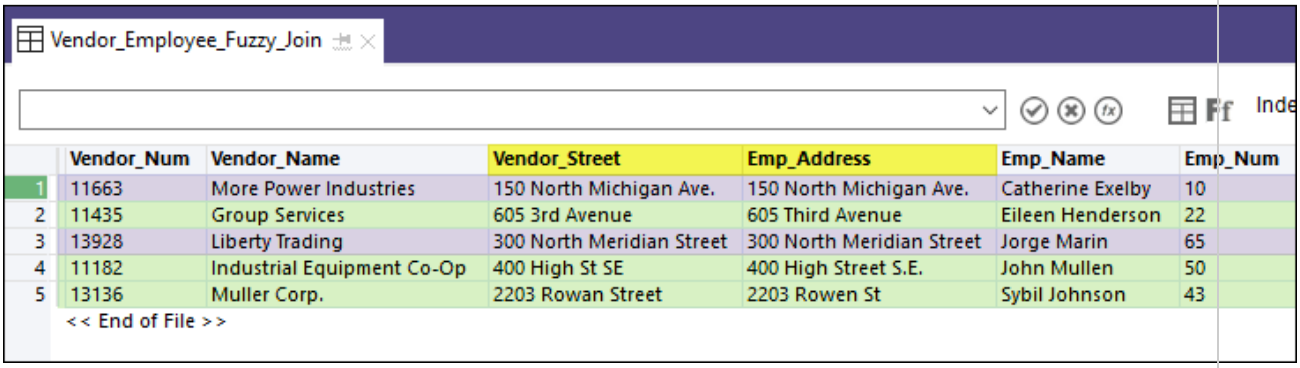
Primary key values	Secondary key values
605 3rd Avenue	605 Third Avenue
400 High St SE	400 High Street S.E.
2203 Rowan Street	2203 Rowen St

Even with data cleansing and harmonization, key values with minor differences in spelling, such as "Rowan" and "Rowen", would probably not be matched.

The key values could be joined by a fuzzy join, depending on the fuzzy join settings.

Output results

In the example of the joined table below, exact key field matches are highlighted purple, and fuzzy key field matches are highlighted green.


Vendor_Employee_Fuzzy_Join

	Vendor_Num	Vendor_Name	Vendor_Street	Emp_Address	Emp_Name	Emp_Num
1	11663	More Power Industries	150 North Michigan Ave.	150 North Michigan Ave.	Catherine Exelby	10
2	11435	Group Services	605 3rd Avenue	605 Third Avenue	Eileen Henderson	22
3	13928	Liberty Trading	300 North Meridian Street	300 North Meridian Street	Jorge Marin	65
4	11182	Industrial Equipment Co-Op	400 High St SE	400 High Street S.E.	John Mullen	50
5	13136	Muller Corp.	2203 Rowan Street	2203 Rowen St	Sybil Johnson	43

<< End of File >>

Fuzzy join versus fuzzy duplicates

A fuzzy join analyzes values in key fields in two tables. To test a single field in a single Analytics table for nearly identical values, see "Fuzzy duplicates analysis" on page 1215.

Improving the effectiveness of fuzzy joining

You can significantly improve the effectiveness of fuzzy joining by incorporating one or more of the following techniques:

- sorting individual elements in primary and secondary key field values
- removing generic elements from primary and secondary key field values
- harmonizing primary and secondary key field values

These techniques allow you to use tighter fuzzy settings and still get the same fuzzy matches, while reducing the number of false positive matches. You can use the techniques separately, or in combination.

Create an expression or a computed field

To use any of the techniques, you need to create an expression or a computed field using the appropriate Analytics function and one or both key fields.

For more information about expressions, see "Using expressions" on page 795.

For more information about computed fields, see "Computed fields" on page 722.

> **Note**
> The **Fuzzy Join** dialog box does not allow the creation of an expression on a secondary key field. However, you can manually create a secondary key field expression in the Analytics command line or in a script. Another option is to create a computed field for use as the secondary key field.

Sorting individual elements in key field values

The SORTWORDS( ) function can improve the effectiveness of fuzzy joining by sorting individual elements in primary and secondary key field values into a sequential order.

Sorting elements, such as the components of an address, can make key field values with the same information, but a different format, more closely resemble each other. A closer resemblance improves the chances that key field values are selected as fuzzy matches for each other.

For more information, see "SORTWORDS( ) function" on page 2378.

For a video providing an overview of SORTWORDS( ), see Fuzzy Matching Using SORTWORDS() (English only).

> **Note**
> Sorting elements in key field values is best suited for fuzzy joining using the Levenshtein distance algorithm.
>
> Sorting elements when fuzzy joining using the Dice coefficient algorithm may or may not be beneficial. Test a set of sample data before deciding whether to use SORTWORDS( ) in conjunction with the Dice coefficient algorithm in a production setting.

> **Caution**
> If you use SORTWORDS( ) in conjunction with fuzzy joining you must apply SORTWORDS( ) to both strings or both fields being compared.

Removing generic elements from key field values

The OMIT( ) function can improve the effectiveness of fuzzy joining by removing generic elements such as "Corporation" or "Inc.", or characters such as commas, periods, and ampersands (&), from primary and secondary key field values.

Removal of generic elements and punctuation focuses fuzzy matching on just the portion of the key field values where a meaningful difference may occur.

For more information, see "OMIT( ) function" on page 2268.

Harmonizing key field values

The REPLACE( ) or REGEXREPLACE( ) functions can improve the effectiveness of fuzzy joining by harmonizing variant forms of the same element in primary and secondary key field values. For example, you could harmonize "Street", "St.", and "St" to use the single value "St".

Harmonizing elements can make key field values with the same information, but a different format, more closely resemble each other. A closer resemblance improves the chances that key field values are selected as fuzzy matches for each other.

For more information, see "REPLACE( ) function" on page 2349 for straightforward replacements, and "REGEXREPLACE( ) function" on page 2335 for more complex replacements.

Output table size and command performance

Output table size

The fuzzy join is similar to the Analytics many-to-many join. All primary key values can potentially be matched to all secondary key values. The size of the output table can be many times greater than the size of either the primary or the secondary input tables.

Command performance

The fuzzy matching algorithms ensure that only key values within a specified degree of fuzziness, or exactly matching values, are actually joined. However, every possible primary-secondary match must be tested, which means that the fuzzy joining process can be time-consuming. The number of individual tests that must be performed is equal to the number of records in the primary table times the number of records in the secondary table.

Limit matching to the first secondary match

You can significantly reduce processing time, and reduce the size of the output results, by selecting **Join only the first occurrence of secondary key matches**. Enabling this option specifies that each primary key value is joined to only the first occurrence of any matching secondary key values.

Enabling the option is appropriate in either of these situations:

- **Any matches?** – you only want to know if any matches, exact or fuzzy, exist between two tables, and you want to avoid the processing time required to identify all matches
- **At most one match** – you are certain that at most only one match exists in the secondary table for each primary key value

Enabling the option is not appropriate if you need output results that contain all possible joins between primary and secondary key values.

> **Note**
> If you select **Join only the first occurrence of secondary key matches** and the first occurrence of a match happens to be an exact match, any subsequent fuzzy matches for the primary key value are not included in the joined output table.

Best practices

Keep output table size and command performance in mind when you prepare primary and secondary input tables, and specify the degree of fuzziness.

- **Tailor the data** – Ensure that only relevant records are included in the primary and secondary tables. If some records have no chance of being matched, filter them out before performing fuzzy matching.
- **Test runs** – For large data sets, do test runs on a small portion of the data as a more efficient way of arriving at suitable settings for the fuzzy matching algorithms. Start with more conservative fuzzy settings, and if required, progressively loosen them.

Fuzzy matching algorithms

When you perform a fuzzy join, you choose between two different fuzzy matching algorithms:

- Dice coefficient
- Levenshtein distance

The algorithms operate completely independently of each other, and can produce somewhat different results. One approach is to perform a fuzzy join twice, once with each algorithm, and compare results. Typically, a number of the fuzzy matches in each result set overlap, but some matches can be unique to each result set.

Degree of fuzziness

You specify the degree of fuzziness for each algorithm, which can dramatically change the size and makeup of the result set. "Degree of fuzziness" refers to how closely two values match.

Depending on the algorithm you select, you use the following settings to control the degree of fuzziness:

Algorithm	Setting
Dice coefficient	◦ N-gram ◦ Percent
Levenshtein distance	◦ Distance

Try experimenting with different degrees of fuzziness. Begin conservatively and produce smaller result sets, and then progressively loosen the settings until you start getting too many joined values that are obviously not matches (false positives).

Dice coefficient

The Dice coefficient algorithm works by measuring the degree of similarity between a primary and a secondary key value, on a scale from 0.0000 to 1.0000. The greater the Dice's coefficient of the two values, the more similar they are.

Show me more

Dice's coefficient	Meaning
1.0000	Each value is composed of an identical set of characters, although the characters may be in a different order, and may use different case. The *n*-grams in the two values are 100% identical. *N*-grams are explained below.
0.7500	The *n*-grams in the two values are 75% identical.
0.0000	The two values have no identical *n*-grams, or the length specified in the **N-gram** setting is longer than the shorter of the two values being compared.

N-grams

Dice's coefficient is calculated by first dividing the values being compared into *n*-grams. *N*-grams are overlapping blocks of characters, with a length of *n*, which is whatever length you specify in the **N-gram** setting.

Here are two of the values from the example above, divided into *n*-grams with a length of 2 characters (*n* = 2).

2203 Rowan Street	22 | 20 | 03 | 3_ | _R | Ro | ow | wa | an | n_ | _S | St | tr | re | ee | et
2203 Rowen St	22 | 20 | 03 | 3_ | _R | Ro | ow | we | en | n_ | _S | St

The Dice's coefficient is the percentage of *n*-grams in the two values that are identical. In this case, 20 of 28 *n*-grams are identical, which is 71.43%, or 0.7143 expressed as a decimal fraction.

> **Note**
> Increasing the length in the **N-gram** setting makes the criterion for similarity between two values stricter.

Percent

When you specify a **Percent** setting, you are specifying the minimum allowable Dice's coefficient of two values for them to qualify as a fuzzy match. For example, if you specify `0.7500`, at least 75% of the *n*-grams in two values must be identical to create a match.

Percent setting	Meaning	2203 Rowan Street / 2203 Rowen St
0.7500	To qualify as a fuzzy match, at least 75% of the *n*-grams in two values must be identical.	Not matched, not included in the joined table (Dice's coefficient = 0.7143)
0.7000	To qualify as a fuzzy match, at least 70% of the *n*-grams in two values must be identical.	Matched, included in the joined table (Dice's coefficient = 0.7143)

For detailed information about how Dice's coefficient works, see "DICECOEFFICIENT( ) function" on page 2116.

Levenshtein distance

The Levenshtein distance algorithm works by measuring the degree of difference between a primary and a secondary key value, on a whole number scale starting at 0. The scale represents the number of single-character edits required to make one value identical to the other value. The greater the Levenshtein distance between the two values, the more different they are.

Show me more

Levenshtein distance	Meaning
0	Each value is composed of an identical set of characters, in an identical order. Case may differ.
2	Two single-character edits are required to make the two values identical. For example: "Smith" and "Smythe" ◦ **edit 1** - substitute 'y' for 'i' ◦ **edit 2** - insert 'e'
3	Three single-character edits are required to make the two values identical. For example: "Hanssen" and "Jansn" ◦ **edit 1** - substitute 'J' for 'H' ◦ **edit 2** - delete 's' ◦ **edit 3** - delete 'e'

Distance

When you specify a **Distance** setting, you are specifying the maximum allowable Levenshtein distance between two values for them to qualify as a fuzzy match. For example, if you specify `2`, no more than two edits can be required to make two values identical.

Distance setting	Meaning	Hanssen / Jansn
2	To qualify as a fuzzy match, no more than 2 character edits can be required to make two values identical.	Not matched, not included in the joined table (Levenshtein distance = 3)
3	To qualify as a fuzzy match, no more than 3 character edits can be required to make two values identical.	Matched, included in the joined table (Levenshtein distance = 3)

For detailed information about how Levenshtein distance works, see "LEVDIST( ) function" on page 2214. Unlike the function, the Levenshtein distance algorithm used in the fuzzy join automatically trims leading and trailing blanks, and is not case-sensitive.

Steps

You can use fuzzy matching of key field values to combine two Analytics tables into a new third table.

Show me how

1. In the Navigator, open the primary table, and right-click the secondary table and select **Open as Secondary**.

   The primary and secondary table icons update with the numbers 1 and 2 to indicate their relation to each other  .
2. Select **Data > Fuzzy Join**.
3. On the **Main** tab select the fuzzy matching algorithm you want to use:
   - **Dice coefficient**
   - **Levenshtein**
4. Depending on the algorithm you selected, provide settings to control the degree of fuzziness.

   Dice coefficient

   - **N-gram**
   - **Percent**

   Levenshtein

   - **Distance**

   The settings are explained below.
5. (Optional) Select **Join only the first occurrence of secondary key matches** to specify that each primary key value is joined to only the first occurrence of any matching secondary key values.

6. Select the primary key field from the **Primary Keys** list.

   You can select only one primary key field, and it must be a character field.

7. Select the secondary key field from the **Secondary Keys** list.

   You can select only one secondary key field, and it must be a character field.

8. Select the fields to include in the joined table from the **Primary Fields** and **Secondary Fields** lists.

   > **Note**
   > You must explicitly select the primary and secondary key fields if you want to include them in the joined table.

   > **Tip**
   > You can **Ctrl+click** to select multiple non-adjacent fields, and **Shift+click** to select multiple adjacent fields.

9. In the **To** text box, specify the name of the new, joined table.
10. (Optional) On the **More** tab:
    a. If you want to process only a subset of records, select one of the options in the **Scope** panel.
    b. If you want to append (add) the output results to the end of an existing Analytics table, select **Append To Existing File**.
11. Click **OK**.

    The new, joined table is output.

Fuzzy Join dialog box options

The tables below provide detailed information about the options in the **Fuzzy Join** dialog box.

Main tab

Options – Fuzzy Join dialog box	Description
Dice coefficient	Use Dice's coefficient for fuzzy matching between primary and secondary key values. ◦ **N-gram** – the *n*-gram length to use Specify a whole number, 1 or greater. Increasing the *n*-gram length makes the criterion for similarity between two values stricter. ◦ **Percent** – the minimum allowable Dice's coefficient of two values for them to qualify as a fuzzy match Specify a decimal fraction, from 0.0000 to 1.0000 (for example, 0.7500). Use a maximum of four decimal places. Decreasing the value increases the number of matches by including matches with a greater degree of fuzziness – that is, values that are more different from

Options – Fuzzy Join dialog box	Description
	each other.
Levenshtein	Use Levenshtein distance for fuzzy matching between primary and secondary key values. ◦ **Distance** – the maximum allowable Levenshtein distance between two values for them to qualify as a fuzzy match Specify a whole number, 1 or greater. Increasing the value increases the number of matches by including matches with a greater degree of fuzziness – that is, values that are more different from each other.
Join only the first occurrence of secondary key matches	Specifies that each primary key value is joined to only the first occurrence of any secondary key matches. If you leave the option unchecked, the default behavior is to join each primary key value to all occurrences of any secondary key matches.
Secondary Table	An alternate method for selecting the secondary table.
Primary Keys Secondary Keys	Specifies the common key field to use to join the two tables. ◦ You can select the common key field directly in the **Primary Keys** and **Secondary Keys** lists. ◦ You can also click **Primary Keys** or **Secondary Keys** to open the **Selected Fields** dialog box where you can select the common key field, or create an expression on the primary key. **Key field guidelines:** ◦ **Data type** – The key fields must be the character data type. ◦ **Length** – If key fields are not the same length, they are automatically harmonized. ◦ **Names and start positions** – Key field names and start positions can be different, but they must describe the same data element. ◦ **Multiple key fields** – Only one key field per table is supported.
Primary Fields Secondary Fields	Specifies the fields to include in the joined table. ◦ You can select fields directly in the **Primary Fields** and **Secondary Fields** lists. ◦ You can also click **Primary Fields** or **Secondary Fields** to open the **Selected Fields** dialog box where you can select the fields, or create an expression on one or more primary fields. ◦ The order in which you select primary and secondary fields dictates the field order in the resulting joined table. As a group, the primary fields appear before the secondary fields in the joined table.
Use Output Table	Specifies whether the Analytics table containing the output results opens automatically upon completion of the operation.
If	(Optional) Allows you to create a condition to exclude records from processing. ◦ You can enter a condition in the **If** text box, or click **If** to create an IF statement using the **Expression Builder**.

Options - Fuzzy Join dialog box	Description
	◦ The condition can reference either the primary or the secondary table, or both. **Note** To access the secondary table fields in the **Expression Builder**, select the secondary table in the **From Table** drop-down list. The **If** condition is evaluated against only the records remaining in a table after any scope options have been applied (**First**, **Next**, **While**).
To	Specifies the name and location of the output table. ◦ **To save the output table to the Analytics project folder** - enter only the table name. ◦ **To save the output table in a location other than the project folder** - specify an absolute or relative file path, or click **To** and navigate to a different folder. For example: `C:\Results\Output.fil` or `Results\Output.fil`. Regardless of where you save the output table, it is added to the open project if it is not already in the project. If Analytics prefills a table name, you can accept the prefilled name, or change it.

More tab

Options - Fuzzy Join dialog box	Description
Scope panel	Specifies which records in the primary table are processed: ◦ **All** - (default) all records in the primary table are processed. ◦ **First** - select this option and enter a number in the text box to start processing at the first record in the primary table and include only the specified number of records. ◦ **Next** - select this option and enter a number in the text box to start processing at the currently selected record in the primary table view and include only the specified number of records. The actual record number in the leftmost column must be selected, not data in the row. ◦ **While** - select this option to use a WHILE statement to limit the processing of records in the primary table based on criteria. • You can enter a condition in the **While** text box, or click **While** to create a WHILE statement using the **Expression Builder**. • A WHILE statement allows records to be processed only while the specified condition evaluates to true. • You can use the **While** option in conjunction with the **All**, **First**, or **Next** options. Record processing stops as soon as one limit is reached.

Options - Fuzzy Join dialog box	Description
	Note The number of records specified in the **First** or **Next** options references either the physical or the indexed order of records in a table, and disregards any filtering or quick sorting applied to the view. However, results of analytical operations respect any filtering. If a view is quick sorted, **Next** behaves like **First**.
Append To Existing File	Specifies that the output results are appended (added) to the end of an existing Analytics table. **Note** Leaving **Append To Existing File** deselected is recommended if you are uncertain whether the output results and the existing table have an identical data structure. For more information about appending and data structure, see "Appending output results to an existing table" on page 200.
OK	Executes the operation. ◦ If you are joining character key fields of different lengths, a message appears stating that Analytics will attempt to harmonize the fields. ◦ If the overwrite prompt appears, select the appropriate option. If you are expecting the **Append** option to appear and it does not, click **No** to cancel the operation and see "Appending output results to an existing table" on page 200.

Automatic harmonization when joining tables

When you join two tables, Analytics automatically harmonizes the key fields in two situations:

- You use one character key field and one numeric key field
- You use character key fields of different lengths

Automatic harmonization simplifies certain kinds of joins and reduces the associated labor.

Whenever Analytics automatically harmonizes key fields, the action and the associated syntax are recorded in the command log.

Automatic harmonization of character-numeric key field joins

If you use a character and a numeric key field to join tables, Analytics automatically harmonizes the data type by using the VALUE( ) function to convert the character field to numeric while performing the join.

Example

You want to join two tables using social security number as the common key field.

- One key field contains numbers and punctuation formatted as character data: 555-44-3322
- The other key field contains only numbers formatted as numeric data: 555443322

Because Analytics automatically harmonizes character-numeric joins, you can perform a standard join without needing to first manually harmonize the fields using functions.

Additional details

- Any alpha characters or punctuation marks such as hyphens and parentheses in the character field are ignored, and only the numbers are considered when matching values in the numeric field.
- The position of alpha characters has no effect on the numeric matching.
- The character field retains its original data type and all its characters, including alpha and punctuation, in the resulting joined table.
- Either the character or the numeric field can be the primary key field.

- Neither the character field, nor the numeric characters in the character field, need to be the same length as the numeric field. Regardless of field length, only numeric values that are identical are matched.

Automatic harmonization involving negative numbers

Automatic harmonization of character and numeric key fields does not directly support the matching of negative numbers. To account for some intended behavior of the VALUE( ) function, Analytics also uses the ABS( ) function on the character key field, which temporarily converts all numeric values to positive while the join is being performing.

If you want to use automatic harmonization with key fields that include negative numbers, perform the join in the usual manner, and then re-run the join using the command log entry manually edited to apply only the VALUE( ) function to the character key field. This method produces correct results when joining character and numeric key fields that include negative values, but it does not work reliably if any non-numeric data exists in either field.

Automatic harmonization of character key field length

If you select character key fields of different lengths when joining tables, Analytics automatically harmonizes their length by adding blanks to the shorter field. The shorter field retains its original length in the resulting joined table.

Automatic length harmonization also works for character-based computed key fields, and common keys composed of multiple character key fields.

Only character key fields are automatically harmonized for length. Numeric and datetime key fields are not.

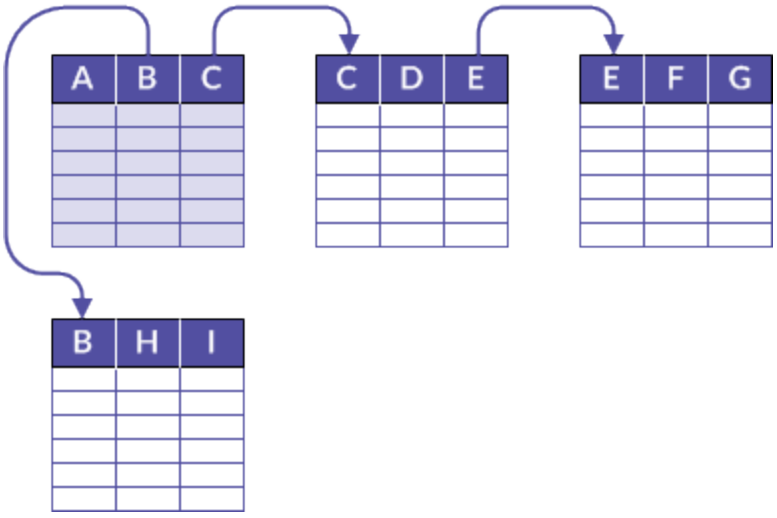
Matching of values not affected by harmonization

Matching of values in character key fields harmonized for length still depends on an exact match between the values themselves. Shorter and longer versions of a value – for example, 'ABC' and 'ABC Corporation' – still do not produce a match even though during the processing of the join they are contained in fields of harmonized length.

Relating tables

Relating tables allows you to combine up to 18 Analytics tables with different record structures and access and analyze data from any combination of fields in the related tables as if they existed in a single table.

Record structures are different if they have one or more fields (data elements) that differ. Relating is a good choice for informational work that requires a quick picture of data associations across several physical tables, or for linking codes with corresponding full names prior to generating reports.



Example

Scenario

You want to create a sales report that contains details about the customers, and the products sold, for the month of March, but the data is spread across three tables.

Approach

You relate the Customer master table with the Orders table, and the Orders table with the Product master table, to create a temporary association of tables that contains all the information you need for the report:

- **customer name and location** – from the Customer master table
- **order details** – from the Orders table
- **product details** – from the Product master table

Relating creates a "virtual" table

The result of relating tables is virtual – the related tables remain independent, and can be unrelated at any time.

Unlike joining or merging, relating does not create a new table. Instead, the fields of the related or child table(s) become available in the parent table from which the relation was created.

If required, you can perform a separate operation and extract any combination of fields from the parent and child tables to a new table.

Relate tables using a common key field

You relate tables using a common key field – that is, a data element such as employee number, vendor ID, or address, that appears in both tables. When identical values exist in the key fields, the result is a match that relates individual records from the separate tables. Partial matching is not supported by the basic relation operation.

Several requirements apply to the key fields in the tables you are relating:

Key field characteristic	Requirement
Data element	Must be the same. For example, both key fields are employee number fields.
Data type	Can be any data type, but key fields must be the same data type as each other. For example, two character fields. Datetime subtypes (date, datetime, and time) can only be related to the same subtype.
Field type	Can be physical fields or computed fields.
Field name	Can be different.
Start position	Can be different.
Field length	Must be the same.
Justification and case in character fields	Must be the same.

Parent and child tables and key fields

The tables and key fields in the relation operation are identified as **parent** and **child** based on the order in which you add the tables to the relation:

- **parent table** – the first table you add (automatically added when you open a table and begin the relation operation)
- **parent key field** – the key field you choose from the parent table
- **child table** – the second table you add, and any subsequent tables you add
- **child key field** – the key field you choose from the child table, or tables

You are free to choose whatever parent and child tables and key fields you want. However, the relation will succeed only if the key fields meet the requirements for relating.

For more information, see "About key fields" on page 216.

Accessing child table fields

Once a relation is established, you can add fields from any of the child tables to the parent view, although there is no requirement that you do so. You can access and analyze child table fields through the parent table – using the **From Table** drop-down list in Analytics dialog boxes and the **Expression Builder** – whether or not you have added them to the parent view.

Child table fields accessed through the parent table are displayed in *table name.field name* format to indicate which table the fields are from. You can access related tables, and modify relations, only through the parent table, not through a child table.

Sorting and indexing of related tables

The virtual table resulting from a relation uses the existing sort order of the parent table. You do not have to sort or index the parent table key field prior to relating tables. However, you might choose to do so, because there is no **Presort** option available for the parent table during a relation.

As part of the internal functioning of the relation operation, the child table key field(s) are automatically indexed in ascending order. These child table indexes persist even after the child tables are unrelated, and can be manually deleted, if necessary.

Additional information about relating

The following table provides additional information about relating.

Functional area	Details
Record matching	Relating a table pair is the logical equivalent of joining them using the **All Primary Records** option – that is, using the type of many-to-one join that includes matching primary and secondary records (parent and child records), and unmatched primary records. As with the corresponding many-to-one join, the relating operation matches parent key values to the first occurrence only of a matching child key value. If additional occurrences of a matching child key value exist, they are ignored. You need to take this behavior into account when planning your table relations, especially if a child table contains legitimate multiple occurrences of a matching key value. One approach is to try reversing the

Functional area	Details
	relation of the two tables, making the child the parent, and vice versa.
Unmatched records and missing field values	If a parent key value has no match in a related child table, for the missing field values, Analytics displays a blank in character and datetime fields, a zero in numeric fields, and “F” in logical fields.
Duplicates or blanks in child table key field	If duplicates or missing values in a child table key field render subsequent analysis invalid, pre-processing the child table to remove the duplicates and/or blanks may be a solution in some circumstances.
Extracting data from related tables	You have two options when extracting data from related tables: ◦ Using either the **View** or **Fields** option in the **Extract** dialog box, you can extract some or all of the data from the parent and child tables into a new Analytics table. The new table is no longer related to any other tables. If you use the **View** option, you must first add the pertinent child table data to the parent view. ◦ Using the **Record** option in the **Extract** dialog box, you can extract the data from the parent table into a new Analytics table. The new table retains the relations of the original parent table. The **Record** option does not support extracting child table data.
Identical key field length not enforced	Analytics does not enforce identical lengths for the common key fields in parent and child tables. It is recommended that you always use fields of identical length, manually harmonizing the lengths prior to performing the relation, if necessary. Results derived from relating using key fields of different lengths are not reliable. Datetime key fields can be different lengths because Analytics, when performing operations involving dates, datetimes, or times, uses an internal Analytics datetime format.
Changing key field data type	You cannot change the data type of a parent or child key field while it is being used to relate tables. If you need to change the data type of either field, you must first delete the relation. If the change results in the data types being different from each other, you are no longer able to use the two fields to relate tables.
Avoiding conditional indexing	Do not use a conditional index for child table key fields. Instead, apply conditions when you perform operations against a parent table and its related table(s). Using conditional indexes when building relations can cause unintended data gaps at different points in a relational hierarchy. The safer approach is to build relations that present as full a data set as the relations warrant, and subsequently apply conditions, if required.
Restrictions on location of tables being related	To be related, tables must be in the same Analytics project. Server tables must be on the same server, and must be accessed using the same server profile. You cannot relate a local table and a server table.
Harmonizing justification and case	When you relate table pairs using character key fields, justification and case must be the same:

Functional area	Details
	◦ Both key fields must have the same justification. Use the LTRIM( ) function to remove leading blanks from the key fields. ◦ Both key fields must be in the same case – UPPER, lower, or Proper. To harmonize the case, use the UPPER( ), LOWER( ), or PROPER( ) function.
Relating UTC-based and non-UTC data	A UTC-based and a non-UTC datetime key field can be used to relate two tables. (UTC is Coordinated Universal Time, the time at zero degrees longitude.) When performing operations involving datetimes or times, Analytics uses an internal Analytics datetime format, so the following two datetimes are interpreted as identical, and constitute a match: ◦ **UTC-based** - 31/12/2014 10:30:15-05:00 ◦ **non-UTC** - 31/12/2014 15:30:15 You should exercise caution if you mix UTC-based and non-UTC time data in an Analytics operation. Although Analytics will match the two time values above, it may not make logical sense to do so, because one value references a time zone, and the other value does not. For more information about UTC, see "Date and Time options" on page 133.

Relate tables

Using a common key field from each table pair, you can relate two or more Analytics tables with different record structures. Once tables are related, you can use the parent table to access and analyze data from any combination of fields in the related tables.

> **Note**
> Carefully identify the parent and child tables in a relation because results can differ if you reverse the order. For more information, see "Common uses of joining or relating" on page 888.

Steps

1. In the Navigator, open the parent table.
2. Select **Data > Relate**.
3. In the **Relations** dialog box, click **Add Table** and select one or more child tables.

   You can relate up to 18 tables, including the parent table.

   > **Tip**
   > You can **Ctrl+click** to select multiple non-adjacent tables, and **Shift+click** to select multiple adjacent tables.
   > You can double-click a child table to add it singly.

4. Click **Add** and then **Close**.

   > **Tip**
   > You can resize the **Relations** dialog box, or tables in the dialog box, and move tables, to create more room in which to work, or to make field information more visible.

5. Drag the key field from the parent table to the corresponding key field in the child table.

   An arrow appears between the two key fields, indicating the relation between the tables.

   Parent and child tables are related using an index on the child table key field. For more information, see "Child table index" on page 934.

6. Relate any additional tables in the same manner as the first table pair, by dragging key field to key field.

   Each additional relation must create a direct or indirect link to the parent table.

> **Note**
> Individual instances of two tables can have only one relation. If you try to relate the same table pair a second time, the operation is prohibited and an error message appears. Add another instance of the required table by clicking the **Add Table** button and selecting the appropriate table.
> For more information, see "Using multiple key fields in isolation" on page 943.

7. (Optional) To remove an individual relation, or a table, from the **Relations** dialog box, do the following:
   - **To remove a relation** – right-click the key field arrow and select **Delete**
   - **To remove a table** – right-click the body of the table and select **Remove Table**

   > **Note**
   > If the table has an existing relation, you must delete the relation first.

8. Click **Finish** to exit the **Relations** dialog box.

   You can now access and analyze data from any combination of fields in the tables you have just related, as if all the fields existed in a single table.

Relations dialog box options

The table below provides detailed information about the options in the **Relations** dialog box.

Options – Relations dialog box	Description
Add Table	Opens the **Add Table** dialog box.
Add Table dialog box	Specifies the tables to include in the relation.
Key field arrow	Specifies the common key field to use to relate each table pair. ◦ You select the common key field by dragging key field to key field. ◦ Once the key field arrow is in place, you can right click it and select **Edit Relation** to change the common key field. **Key field guidelines:** ◦ **Data type** – The key fields can be any data type. For each table pair, key fields must be the same data type. ◦ **Datetime subtypes** – Datetime subtypes (date, datetime, and time) can only be related to the same subtype. ◦ **Length** – It is recommended that key field lengths be identical for each table pair. ◦ **Names and start positions** – Key field names and start positions can be different, but they must describe the same data element. ◦ **Multiple key fields** – If required, the common key can include more than one key field per table. For more information, see "Using multiple key fields" on page 941.
Arrange Tables	(Optional) You can right-click the working area of the **Relations** dialog box and select

Options - Relations dialog box	Description
	Arrange Tables to neaten the arrangement of tables and key field arrows.
Finish	Executes the operation. You can now access and analyze data from any combination of fields in the tables you have just related, as if all the fields existed in a single table. When accessed from the parent table, the **From Table** drop-down list in Analytics dialog boxes and the **Expression Builder** allows you to select related tables from which you can then select individual fields for analysis or processing.

Child table index

Parent and child tables are related using an index on the child table key field.

If no index exists	If no index exists on the child table key field, Analytics automatically creates one when you relate the parent and child tables.
If you want to specifically name the index	If you want to specifically name the child table index auto-created by Analytics: a. Right-click when you drag the key field from the parent table to the child table. b. Select **Relate using Named Index**. **Relate using Named Index** is disabled if an index already exists. c. Specify a name for the index, and if desired a location other than the default (the folder containing the Analytics project) d. Click **OK**.
If multiple indexes exist	If the child table has two or more existing indexes on its key field, you are presented with a list of these eligible indexes. Select the appropriate index and click **OK**

(Optional) Add child table fields to the parent view

There is no requirement that you add child table fields to the parent view, although you may find that doing so helps you better visualize the related data.

1. Right-click the parent view and select **Add Columns**.
2. Select a child table from the **From Table** drop-down list.

   In the **Available Fields** list, child table fields appear in the format *child table name.field name*.
3. Select one or more child table fields to add to the parent view.

   Child table fields appear in the parent view in the order in which you select them.

4. If applicable, select additional child tables from the **From Table** drop-down list, and select additional child table fields to add to the parent view.
5. Click **OK**.

   The child table fields are added to the parent view. Analytics fills missing values in child table fields for unmatched parent table records with blanks or zeros.

Modify relations

You can modify table relations, including adding or removing related tables, changing key fields, and creating expressions. You can also delete a relation entirely.

1. Open the parent table and select **Data > Relate**.
2. In the **Relations** dialog box, do one or more of the following:

Delete a relation entirely	Right-click the arrow connecting the two tables and select **Delete**. You cannot delete the relation if any child table fields are referenced in either the current parent view, or in an active computed field. You must first remove the fields from the view, or delete the computed field, or make it inactive. You can make a computed field inactive by switching to a different view, or by closing and reopening the parent table. You also cannot delete the relation if it indirectly links another child table to the parent table. You must first delete the relation to the other child table.
Remove a table from the relation	Right-click the body of the table and select **Remove Table**. You cannot remove a table without first deleting the table relation (the arrow connecting the table). You cannot remove the parent table. Tables without relations are automatically removed when you exit the **Relations** dialog box.
Add a table to the relation	Click **Add Table**, select one or more tables, click **Add**, and click **Close**. Create the relation to the new table(s) in the usual manner.
Edit a relation between two tables	Right-click the arrow connecting the two tables and select **Edit Relation**, or double-click the arrow. You can change the parent and/or child key fields, and select a different index for the child table, if one or more additional indexes already exist. Click **OK** to save your changes. **Note** You may have to close and reopen the parent table to refresh the relation so that it reflects the changes you made.
Create an expression	Right-click the parent table and select **New Expression** to open the **Expression Builder**. You can create an expression using fields from the parent and any related child tables. You can only create an expression through the parent table.
Auto-arrange tables and relation arrows	Right-click the working area and select **Arrange Tables**.

3. Click **Finish** to exit the **Relations** dialog box.

How table relations are structured

Relations between tables are hierarchical. A single parent table is at the top of the hierarchy, and it can be related to multiple child tables.

Once the tables are related, you can access and analyze data from any combination of fields in the related tables as if they existed in a single table.

General guidelines for relating tables

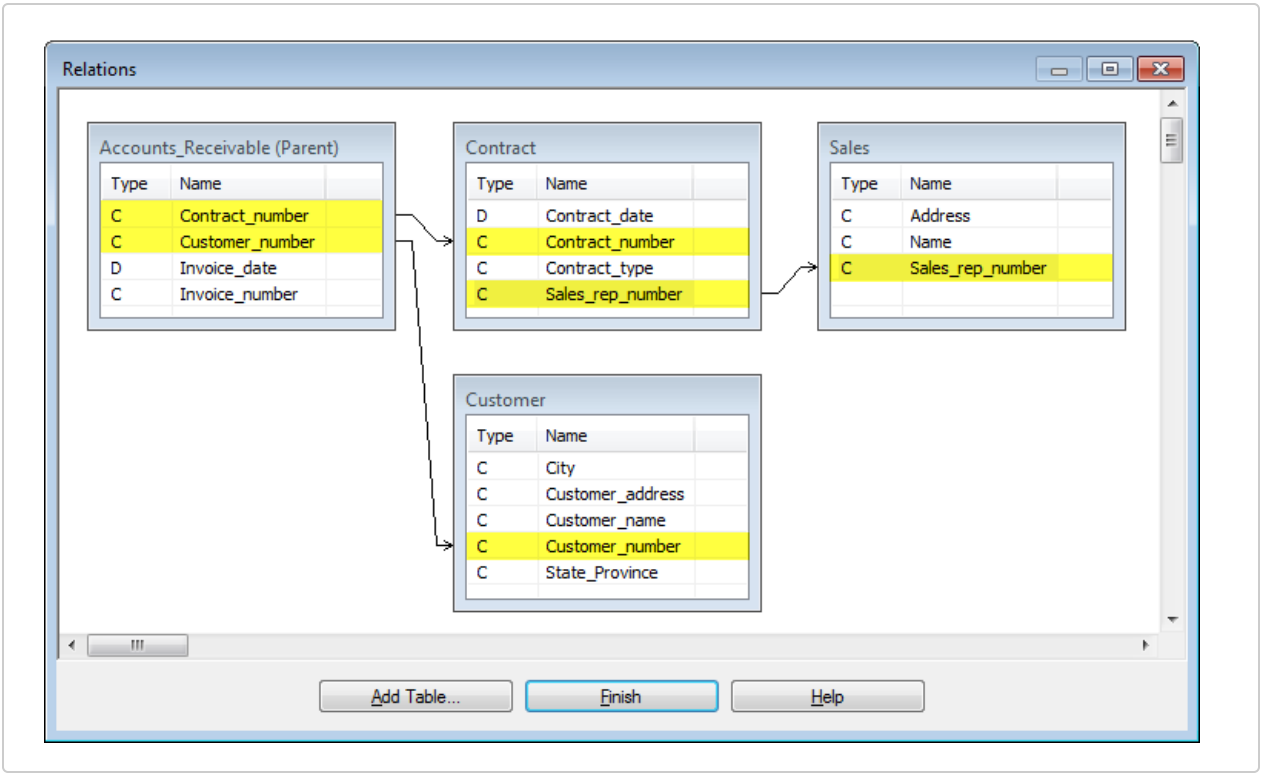
- Only one parent table is allowed.
- Child tables can be related to child tables of their own, which are grandchild tables to the parent table, and so on.
- A maximum of 18 tables can be related to one another, including the parent table.

The **Relations** dialog box provides a graphical work area, so you can easily manage multiple relations.

Example

In the figure below, the tables and key fields are related in the following manner:

Parent table	Common key field	Child tables	Common key field	Grandchild tables
Accounts Receivable	Contract_number	Contract	Sales_rep_number	Sales
	Customer_number	Customer		



Detailed guidelines for relating tables

Building relations that successfully display the data you want requires a certain amount of planning, especially if the relation involves a number of tables. The guidelines that follow may help you as you build a relation.

Matching between key fields

Carefully consider the common key fields in each table pair, the type of values they contain, and the matches that are likely to result. The completeness and accuracy of matching between key fields directly impacts the quality of any subsequent informational review or analysis.

Reversing the parent-child relation

Consider the implications of making one table the parent and the other the child, and how the results might differ if you reverse the position of the two tables.

Keep in mind that relations in Analytics are of the many-to-one type. Single or identical parent key values are related to the first occurrence only of a matching child key value. Additional occurrences of matching child key values, and the records containing them, are ignored.

If legitimate duplicate child key values exist, making the child table the parent may yield more complete results, assuming the current parent table does not also contain legitimate duplicates.

If both tables contain legitimate duplicates, joining the tables using a many-to-many join may be a better approach.

The effect of a table's position in the relational hierarchy

At each level of a relational hierarchy, a child table can be a parent to a table at the next lower level of the hierarchy.

Because of the many-to-one matching of key values between parent and child, as a table is positioned progressively lower in the hierarchy, the possibility increases that more of its data will be omitted from the final relation. This potential cumulative effect of many-to-one matching is not an issue if a one-to-one correspondence exists between the values in all common key fields.

If you want to ensure that all the records in a table are included in a relation, make that table the parent table.

Using intermediary tables

If you want to relate two tables that lack a common key field, you may be able to build the relation using one or more intermediary tables.

In the figure above, the **Contract** table serves as an intermediary table that indirectly relates the **Accounts Receivable** and the **Sales** tables. You may or may not be interested in the data in the intermediary table.

Variations on the basic relational association

Beyond the basic parent-to-child relational association, the following types of relational associations are allowed or disallowed:

Individual instances of two tables can have only one relational association	If you try to relate the same table pair a second time, the operation is prohibited and the following message appears: "One of these files is already part of a relation. To create another relation, add another instance of the file." You can add another instance of the required table by clicking the **Add Table** button in the **Relations** dialog box and selecting the appropriate table. An additional table instance is added with an incrementing numeric suffix, or a name of your choosing. Alternatively, you can make a copy of the appropriate table layout in the **Navigator**, and add the copy to the **Relations** dialog box.
Relating tables using multiple key fields	◦ Tables can be related using two or more key fields in combination if the key fields are concatenated. For more information, see "Using multiple key fields in combination" on page 941.

	◦ A parent table can be related to two (or more) separate instances of the same child table. The relation can be between: • the same parent key field and two different child key fields • two different parent key fields and the same child key field For more information, see "Using multiple key fields in isolation" on page 943.
Relating a table to itself	A table can be related to a separate instance of itself.

Using multiple key fields

Two different situations can require that you use multiple common key fields to accurately join or relate tables:

Use…	When…	Example
Multiple key fields in combination	The values in a single common key field are insufficiently unique to accurately join or relate two tables.	You need to use both the vendor ID field and the Location field to accurate join or relate two tables.
Multiple key fields in isolation	The values required to join or relate two tables are split between two (or more) key fields in one of the tables being joined or related.	You are joining or relating tables on Name. The primary or parent table contains a single Name field. However, names can occur in either of two fields in the secondary or child table.

Using multiple key fields in combination

If the values in a single common key field are insufficiently unique to accurately join or relate two tables, you need to use multiple common key fields in combination.

Example

You want to join or relate two tables using `Vendor_ID` as a common key field. However, some vendors have multiple locations for the same vendor ID.

In this example, Vendor A4538 has locations in Vancouver, Richmond, and Coquitlam.

Single key field

If you join or relate tables using only `Vendor_ID`, secondary or child table records with anything other than the first listed location for the vendor are not included in the joined table (assuming a many-to-one join), or the related table, and locations are incorrectly matched between tables.

Primary/Parent table

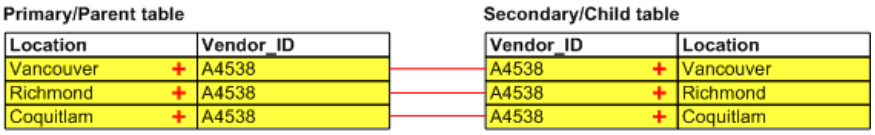
Location	Vendor_ID
Vancouver	A4538
Richmond	A4538
Coquitlam	A4538

Secondary/Child table

Vendor_ID	Location
A4538	Vancouver
A4538	Richmond
A4538	Coquitlam

Multiple key fields in combination

To capture all vendor locations, and ensure proper location matching, you need to use both the `Vendor_ID` and the `Location` fields as key fields in both tables. When combined, the values in each field form a single unique value that can be used to reliably match records between the two tables.



Specifying multiple key fields in combination when joining tables

When joining tables, you can use either of these methods to specify multiple key fields in combination:

- Select more than one key field in the **Join** dialog box.
- In each table, create a computed field that concatenates (adds together) two or more key fields, and join the tables using the computed fields. For more information, see "Concatenate key fields" on page 948.

Select more than one key field in the **Join** dialog box

When you select more than one key field for each table in the **Join** dialog box, the following conditions apply:

Data structure	The data structure and data format requirements that apply when using one key field still apply to the corresponding key fields in each table when using multiple key fields. For more information, see "Data structure and data format requirements" on page 848.
Data type	Within a table, the multiple key fields can be of different data types - for example, first name, last name, and date of birth.
Sort order	Selecting more than one key field creates a nested sort order in the output table, assuming you **Presort** the primary table while performing the join. The order in which you select the key fields dictates the priority of the nested sort order.

Specifying multiple key fields in combination when relating tables

When relating tables, you can use either of these methods to specify multiple key fields in combination:

- In each table, create a computed field that concatenates (adds together) two or more key fields, and relate the tables using the computed fields. For more information, see "Concatenate key fields" on page 948.
- In each table, define a new field long enough to encompass the data in the multiple key fields, and relate the tables using the new field. For more information, see "Define a physical field" on page 717.

> **Note**
> Unlike joining, when you relate tables you can select only one key field per table pair, so you need to employ one of the methods above to use multiple key fields in combination.

Define a new field to encompass data in multiple key fields

When you define a new field to encompass data in multiple key fields, the following conditions apply:

Data structure	The data structure and data format requirements that apply when using one key field still apply to newly created fields that encompass multiple key fields. For more information, see "Data structure and data format requirements" on page 848.
Field adjacency	This method works only if the multiple key fields are adjacent in each table. Fields can be made adjacent by extracting by field to a new table and selecting the fields for extraction in the required order.
Data type	New fields that encompass multiple key fields can be any data type supported by the source data. If the multiple key fields are of different data types, you can create the new field encompassing them as a character field because you are using it only for the purposes of relating tables.

Using multiple key fields in isolation

If the values required to accurately join or relate two tables are contained in two (or more) key fields in one of the tables being joined or related, you need to use these multiple key fields in isolation.

For each of the multiple key fields you perform a separate join operation, or form a separate relational association, in a process that yields a unified final result with a complete set of data.

If you join or relate the tables using only one of the key fields in the tables with multiple key fields, the resulting data is incomplete.

Example 1: Two key fields in the secondary or child table

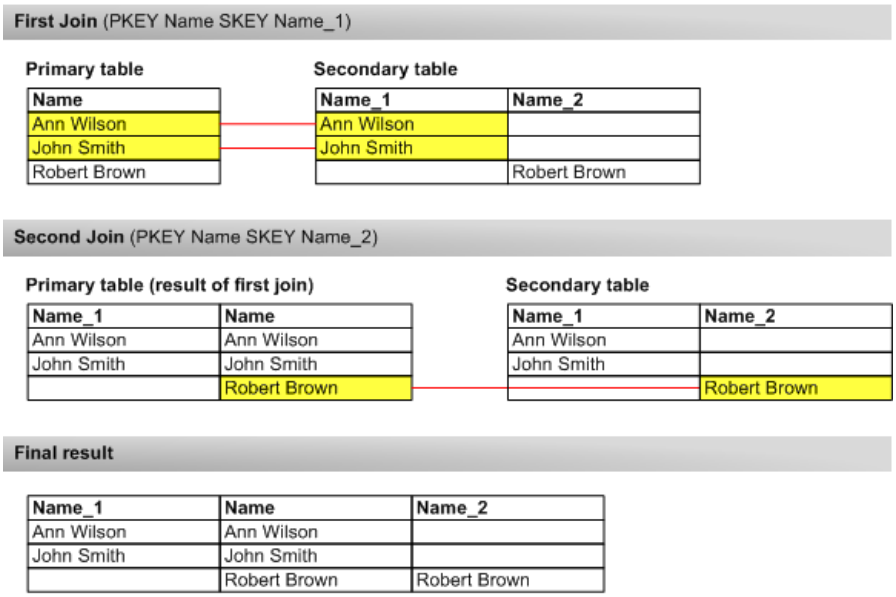
You want to use names to join or relate two tables. The primary or parent table contains the **Name** field. However, the secondary or child table contains two different name fields - **Name_1** and **Name_2**. Matching names in the secondary or child table could appear in either of the two name fields.

Joining tables

To capture all possible matches between names you need to perform two successive joins, with each join using only one of the key fields in the secondary table. You use the output table from the first join as the primary table in the second join.

With each join you must select the join type that includes matched and unmatched primary records (that is, all primary records) so you do not lose the unmatched primary records at any point in the process.

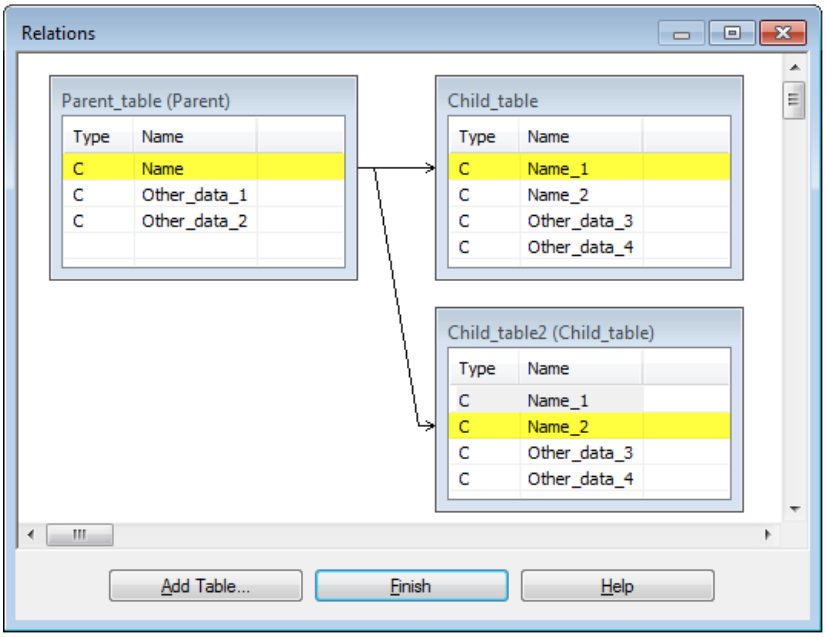
> **Note**
>
> The figures below illustrate only the key fields in the tables being joined. Typically, tables also include other data in non-key fields.



Relating tables

To capture all possible matches between names you need to add an additional instance of the child table for the additional relation between the parent key field and the second child key field.

You add additional instances of the child table by clicking the **Add Table** button in the **Relations** dialog box and selecting the appropriate table.



Final result

Name_1	Name	Name_2
Ann Wilson	Ann Wilson	
John Smith	John Smith	
	Robert Brown	Robert Brown

Example 2: Two key fields in the primary or parent table

You want to use a tax filer ID number to join or relate two tables. The primary or parent table contains tax return information, and the secondary or child table maps the tax filer ID number to social security numbers.

The primary or parent table contains the **ID** field with ID numbers of the main filers, and the **Secondary_ID** field with ID numbers of the secondary filers (spouses), when applicable. The secondary or child table contains one **ID** field with ID numbers for everyone. You want to produce joined or related data that associates social security numbers to both main and secondary tax filers.

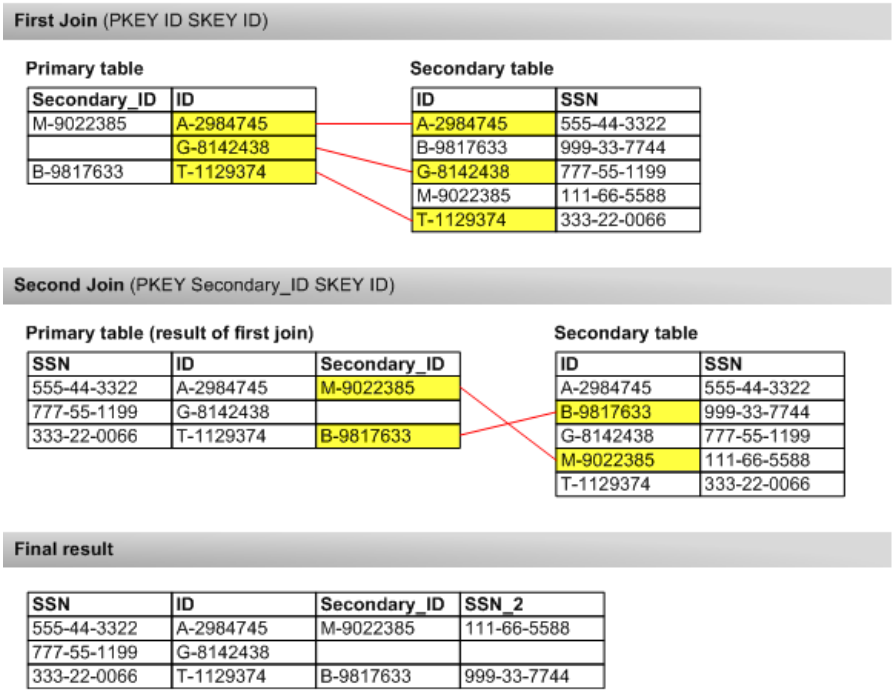
Joining tables

To associate social security numbers to both main and secondary tax filers you need to perform two successive joins, with each join using only one of the key fields in the primary

table. You use the output table from the first join as the primary table in the second join.

With each join you must select the join type that includes matched and unmatched primary records (that is, all primary records) so you do not lose the unmatched primary records at any point in the process.

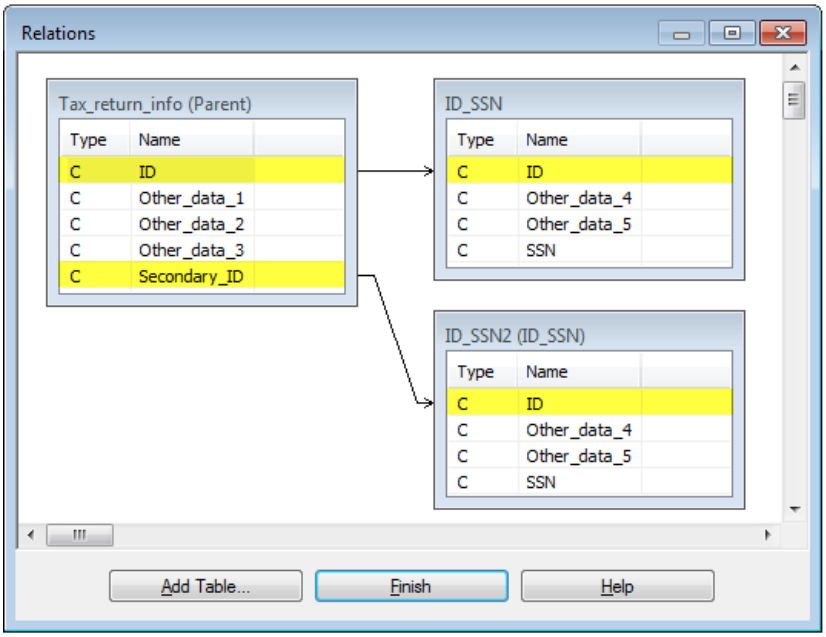
> **Note**
>
> The figures below illustrate only the key fields in the tables being joined. Typically, tables also include other data in non-key fields.



Relating tables

To associate social security numbers to both main and secondary tax filers you need to add an additional instance of the child table for the relation between the second parent key field and the child key field.

You add additional instances of the child table by clicking the **Add Table** button in the **Relations** dialog box and selecting the appropriate table.



Final result

SSN	ID	Secondary_ID	SSN_2
555-44-3322	A-2984745	M-9022385	111-66-5588
777-55-1199	G-8142438		
333-22-0066	T-1129374	B-9817633	999-33-7744

Concatenate key fields

If a single key field is insufficiently unique to accurately relate two tables, you can create a computed field in each table that concatenates two or more key fields, and relate the tables using the computed fields.

You can also use this method when joining tables, but there is no requirement that you do so because joining allows you to select more than one key field per table.

As with single key fields, concatenated key fields must have an identical data structure and data format in the two tables being related.

> **Note**
>
> You can concatenate only character key fields, so you may have to use Analytics functions to convert non-character data prior to concatenating. For more information, see "Harmonizing fields" on page 851.

1. Open the parent table and select **Edit > Table Layout**.
2. Click **Add a New Expression**  .
3. Enter a **Name** for the concatenated key field.
4. Click **f(x)**  to open the **Expression Builder**.
5. Build an expression using two or more key fields and the Add operator (**+**).

   For example: `vendor_ID + location_code`
6. Click **OK**.

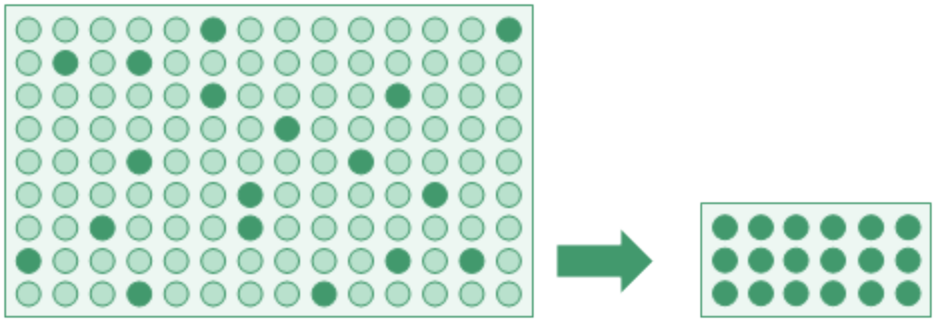
   If you get an “Expression type mismatch” error, one or more of the key fields are probably not character key fields.
7. Click **Accept Entry**  and click **Close**  to exit the **Table Layout** dialog box.
8. Open the child table and repeat the same steps to add an identical concatenated key field to the child table.
9. Relate the two tables using the concatenated key field.

Sampling data

You want to discover the rate of deviation from a prescribed control, or the total amount of monetary misstatement, in an account or class of transactions. However, you may not have the time or the budget to examine every record in the data set.

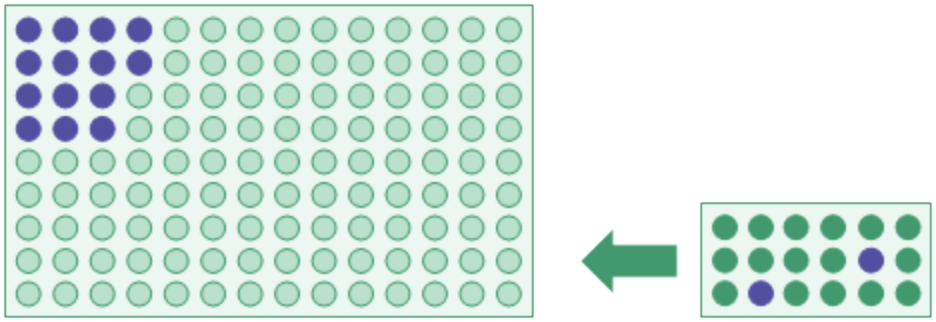
You can use Analytics to draw a statistically valid subset of the data, called **a sample**, and analyze this much smaller set of data instead.

Drawing a sample



You can then **project** the results you get from analyzing the smaller set of data to the entire population of data. The projection creates an estimate of the overall deviation rate, or the overall amount of misstatement.

Projecting results



The sample selection and the projection use statistical formulas, which ensure a reasonable and measurable degree of confidence that the estimate is close to what you would have got if you had actually examined every record.

> **Note**
>
> The information about sampling in this guide is intended to help users already familiar with audit sampling perform sampling tasks in Analytics. The information is not intended to explain audit sampling theory, which is a complex subject.
>
> For in-depth coverage of audit sampling, consult a resource such as AICPA's *Audit Guide: Audit Sampling*.

Sampling types

Analytics has three types of sampling:

- record sampling (attributes sampling)
- monetary unit sampling
- classical variables sampling

The type of sampling you choose depends on the nature of the analysis you are doing, and the nature of the data.

Sampling in Analytics is statistical sampling

Sampling in Analytics is statistical sampling. A sample drawn by Analytics is statistically valid, or representative, because it is planned, drawn, and evaluated using accepted statistical formulas.

The formulas are based on probability distributions. Record sampling and monetary unit sampling are based on the Poisson distribution, and classical variables sampling is based on the normal distribution.

Which type of sampling should I use?

The table below provides guidelines about choosing the type of sampling to use.

> **Note**
> If all you require is a non-representative random selection of records, see "Generate a random selection of records" on page 221. Projecting results based on a non-representative selection has no statistical validity.

Sampling type	Use if:
"Record sampling (attributes sampling)" on page 964	You are testing controls. Record sampling is appropriate if you are auditing the rate of deviation from a prescribed control. If your analysis will yield a yes/no or pass/fail result for each record being analyzed then you should use record sampling.
"Monetary unit sampling" on page 993	You are analyzing an account or class of transactions for monetary misstatement, and you expect the financial data to have the following characteristics: ◦ no misstatements, or only a small number of misstatements For example, less than 5% of the items are misstated. ◦ more likelihood of overstatements than understatements ◦ no zero dollar items

Sampling type	Use if:
"Classical variables sampling" on page 1028	You are analyzing an account or class of transactions for monetary misstatement, and you expect the financial data to have the following characteristics: ◦ a moderate to larger number of misstatements For example, 5% or more of the items are misstated. ◦ either overstatements or understatements may exist ◦ zero dollar items may exist

Sampling involves professional judgment

The sampling features in Analytics automate a number of the calculations and processes involved in audit sampling. However, reliable and effective sampling also requires that you apply professional judgment in several areas.

> **Note**
> If you are not familiar with the professional judgments required to perform audit sampling in a reliable manner, we recommend that you consult audit sampling resources, or an audit sampling specialist, before using Analytics for sampling in a production environment.

Area	Required judgment
Confidence	assessing the required degree of confidence that a sample is representative
Materiality Monetary precision	deciding upon the acceptable level of misstatement in an account or class of transactions
Tolerable deviation rate	deciding upon the acceptable rate of deviation from a prescribed control
Selection method	choosing an appropriate sample selection method
Evaluation method	for classical variables sampling, choosing an appropriate evaluation method

A word about terminology

To help those less familiar with audit sampling, several terms that appear in this guide are defined very simply below.

Show me more

> **Note**
> These are layperson definitions that intentionally simplify the more precise definitions used in the audit and assurance profession.

control	A mandated process that provides assurance. For example: all vouchers must be approved and signed by a manager.
deviation, control deviation	A failure to comply with a control. For example: a voucher was processed without a manager's signature.
misstatement	An inaccurate number – typically, an inaccurate monetary amount.
materiality	The point at which something becomes significant.
material misstatement	An inaccuracy that is large enough to matter. Can refer to a single number, or to an entire account, as in: "The account is materially misstated".
population	The entire set of records in a file, or the entire monetary amount in an account or class of transactions, from which a sample is drawn.
project	To estimate; to extrapolate an unknown value based on an observed value or values.
representative	Having the same characteristics as the larger group.
sampling, sample	A statistically valid process that selects less than 100% of a population. This subset is known as "the sample".
seed	A number that you specify, or that Analytics randomly selects, to initialize the Analytics random number generator.
tainting	In a misstated amount, the percentage of the book value (recorded value) that the misstatement represents. For example: a $200 book value that should actually be $180 is misstated by $20 and therefore has a 10% tainting.
tolerable	Acceptable; within the bounds of acceptability.
universe	Another term for "population".

Analytics terms versus industry terms

A number of the labels on the sampling dialog boxes in Analytics use Analytics terms. You may find the terms disorienting if you already understand audit sampling and the associated terminology. For a mapping of Analytics terms to industry terms, see "Audit sampling terminology" on page 959.

Sample selection methods

Sample selection methods are the specific methods used to select the records contained in a sample.

For record sampling and monetary unit sampling, Analytics supports three sample selection methods:

- fixed interval
- cell
- random

For classical variables sampling, the random selection method is the only possibility.

Sample selection method versus sampling type

It is important to understand the distinction between sample selection method and sampling type.

Sampling type refers to the overall statistical method used to arrive at an estimate about a population.

Sample selection method refers to the way in which records are drawn from a population for inclusion in a sample.

Sampling type	Sample selection methods available	Details
Record sampling	◦ fixed interval ◦ cell ◦ random	The records contained in the sample are directly selected
Monetary unit sampling	◦ fixed interval ◦ cell ◦ random	The records contained in the sample are those that correspond to the selected monetary units
Classical variables sampling	◦ random	The records contained in the sample are directly selected

Fixed interval selection method

With the fixed interval selection method, an initial monetary unit or record is selected, and all subsequent selections are a fixed interval or distance apart – for example, every 5000th monetary unit, or every 20th record, after the initial selection.

Show me more

To use the fixed interval selection method, you specify:

- The interval value that Analytics generates when you calculate the sample size
- A start number greater than zero and less than or equal to the interval value

The start number and the interval value are used to select which records are contained in the sample.

> **Note**
> If you want Analytics to randomly select a start number, you can enter a start number of '0', or leave the start number blank.

Example

If 62 is the interval generated by Analytics, and you choose 17 as the start number, the following monetary units, or record numbers, are selected:

- 17
- 79 (17+62)
- 141 (79+62)
- 203 (141+62)
- and so on

Each selection is the same distance, or fixed interval, apart.

For monetary unit sampling, the actual record numbers selected are the ones that correspond to the selected monetary units. For more information, see "How monetary unit sampling selects records" on page 994.

Considerations

When you use the fixed interval selection method, you need to be alert to any patterns in the data. Because a fixed interval is used for sample selection, a non-representative sample can result if the data has a pattern that coincides with the interval you specify.

For example, you sample expenses using an interval of $10,000, and the same expense category appears at ten-thousand-dollar intervals in the file, which results in all the selected records coming from a single expense category. This type of scenario is uncommon, but you should be aware that it could occur.

Cell selection method

With the cell selection method, the data set is divided into multiple equal-sized cells or groups, and one monetary unit, or one record, is randomly selected from each cell.

Show me more

To use the cell selection method, you specify:

- The interval value that Analytics generates when you calculate the sample size
- A seed value used to initialize the random number generator in Analytics

The interval value dictates the size of each cell. The random number generator specifies which monetary unit or which record number is selected from each cell.

> **Note**
>
> If you want Analytics to randomly select a seed value, you can enter a seed value of '0', or leave the seed value blank.

Example

If 62 is the interval generated by Analytics, one monetary unit, or one record number, is randomly selected from each of the following cells:

- cell 1 (1 to 62)
- cell 2 (63 to 124)
- cell 3 (125 to 186)
- and so on

Each selection is a random distance apart, but constrained within its cell.

For monetary unit sampling, the actual record numbers selected are the ones that correspond to the selected monetary units. For more information, see "How monetary unit sampling selects records" on page 994.

The seed value

If you specify a seed value, it can be any number. Each unique seed value results in a different set of random numbers. If you respecify the same seed value, the same set of random numbers is generated. Explicitly specify a seed value, and save it, if you want to replicate a particular sample selection.

Considerations

The main advantage of the cell selection method over the fixed interval selection method is that it avoids problems related to patterns in the data.

For monetary unit sampling, two disadvantages exist:

- Amounts can span the dividing point between two cells, which means they could be selected twice, yielding a less consistent sample than the sample generated by the fixed interval method.
- Larger amounts that are less than the top stratum cutoff have a slightly reduced chance of being selected.

Random selection method

With the random selection method, all monetary units or records are randomly selected from the entire data set, or from each stratum if you are using classical variables sampling.

Show me more

> **Note**
> Do not use the random selection method with monetary unit sampling if you intend to use Analytics to evaluate any misstatement detected in the resulting sample. Evaluating monetary unit samples requires that you use the fixed interval or the cell selection methods.

To use the random selection method, you specify:

- The sample size, as calculated by Analytics – that is, the number of samples to select
- A seed value used to initialize the random number generator in Analytics
- The population size – that is, the absolute value of the sample field, or the total number of records in the data set

For classical variables sampling, sample size and population size can be automatically prefilled by Analytics.

The random number generator specifies which monetary units or which record numbers are selected from the data set. Each selection is a random distance apart.

> **Note**
> If you want Analytics to randomly select a seed value, you can enter a seed value of '0', or leave the seed value blank.

The seed value

If you specify a seed value, it can be any number. For classical variables sampling, the seed value must be a positive number not greater than 2,147,483,647.

Each unique seed value results in a different set of random numbers. If you respecify the same seed value, the same set of random numbers is generated. Explicitly specify a seed value, and save it, if you want to replicate a particular sample selection. You can also retrieve a seed value from the command log.

Considerations

Large amounts may be excluded from a monetary unit sample

With the random selection method, each monetary unit has an equal chance of selection, and there is no guarantee that the resulting sample will be evenly distributed. As a result, the distance or gap between selected units may be large in some instances. If all the monetary units associated with a large amount happen to fall into a gap, the amount is not included in the sample. There is also no top stratum cutoff available when using the random selection method.

With the fixed interval and cell selection methods, there is an assurance that the selected units are evenly distributed, or relatively evenly distributed. And top stratum cutoff is available.

Amounts may be included more than once in a monetary unit sample

Analytics does not generate the same random number twice, however random numbers that are close, or sequential, can occur.

With monetary unit sampling, close or sequential random numbers equate to close or sequential monetary units being selected, which in turn can lead to an associated amount being selected more than once.

With record sampling and classical variables sampling, the same problem does not exist because each random number equates to a different record.

Random number algorithms

For record sampling and monetary unit sampling, the random number generator in Analytics has two algorithm options:

- Mersenne-Twister
- The default Analytics algorithm

Mersenne-Twister is a widely used random number algorithm and it has better statistical properties than the default Analytics algorithm. Use the default algorithm if you require backward compatibility with Analytics scripts or sampling results created prior to Analytics version 12.

For classical variables sampling, Mersenne-Twister is not an option and the default Analytics algorithm is used.

Add a record number field

You may find it useful to add a record number field to the Analytics table from which you are drawing a sample. After you draw the sample, the specific record numbers that were selected from the source table are displayed in the output table containing the sample.

> **Note**
> A record number field is automatically included in the output table when you use classical variables sampling.

Show me how

1. In the source table, create a computed field that uses the following expression:

   ```
   RECNO( )
   ```

   For more information, see "Define a conditional computed field" on page 734.

2. When you sample the data, output by **Fields**, not by **Record**.

   You must output by **Fields** in order to convert the computed record number field to a physical field that preserves the record numbers from the source table.

3. Include the computed record number field in the output fields you specify.

Audit sampling terminology

In a number of instances, Analytics sampling terminology differs from the terms commonly used in the audit and assurance profession. The sections below outline the Analytics terms and the equivalent industry terms.

For simple definitions of some common audit sampling terms, see "A word about terminology" on page 951.

Names of the types of sampling in Analytics

Analytics has three types of sampling:

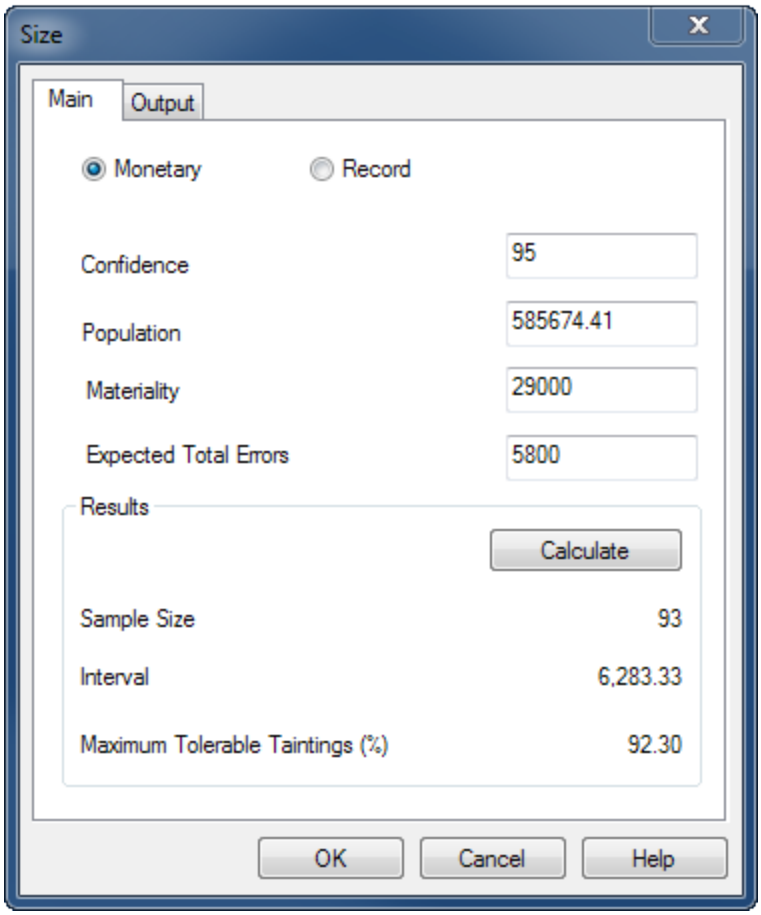
- record sampling
- monetary unit sampling
- classical variables sampling

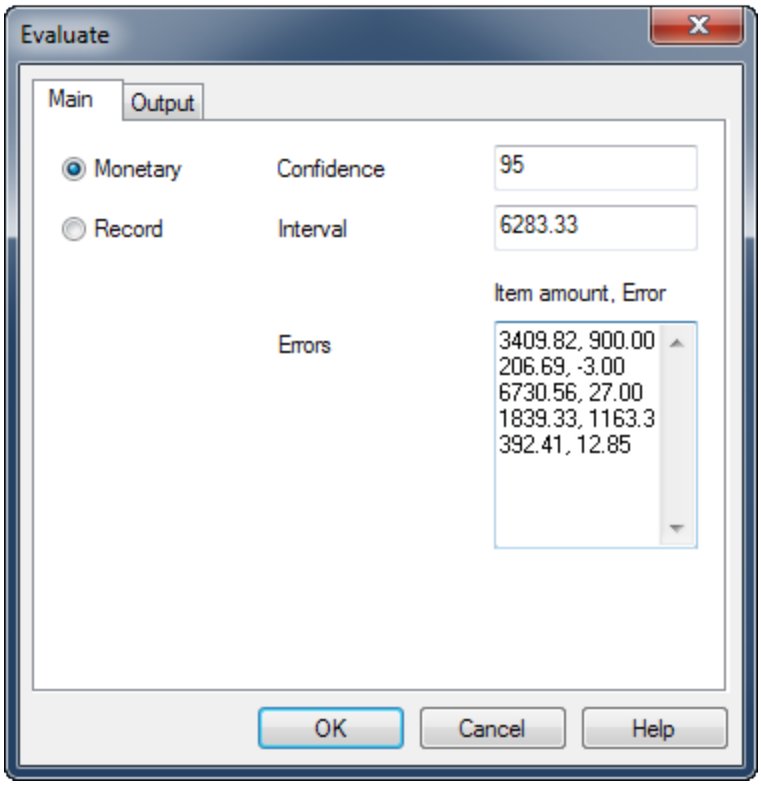
The table below lists other terms in use for these types of sampling.

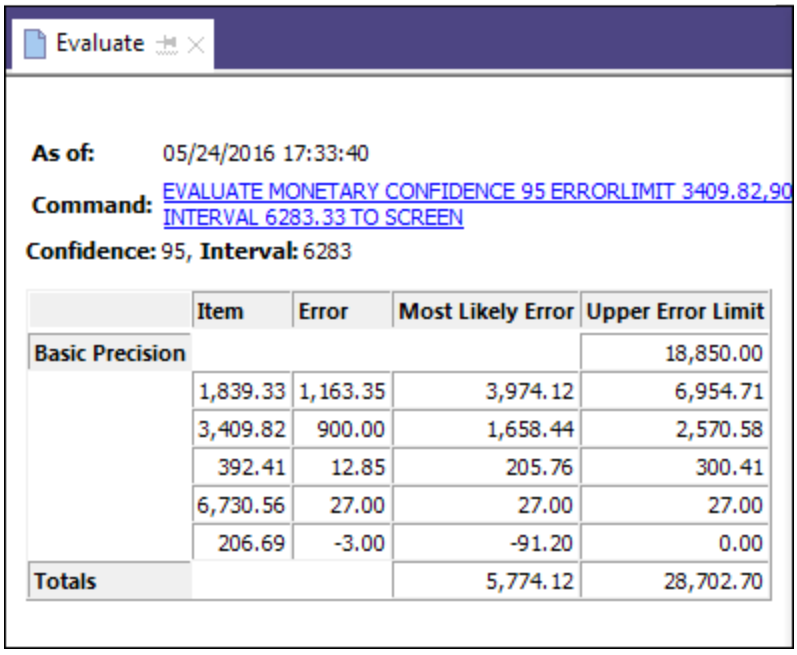
Analytics term	Industry term
Monetary unit sampling	monetary unit sampling dollar-unit sampling probability-proportional-to-size sampling cumulative monetary amount sampling
Record sampling	attribute sampling attributes sampling
Classical variables sampling	classical variables sampling

Monetary unit sampling terminology

The table below lists the monetary unit sampling terminology used in Analytics and provides the equivalent industry term or terms.

Size dialog box	Analytics term	Industry term
	**Confidence**	confidence confidence level reliability
	Population	population population size
	Materiality	tolerable misstatement
	Expected Total Errors	expected misstatement estimated misstatement expected population misstatement
	Sample Size	sample size
	Interval	sampling interval
	Maximum Tolerable Taintings (%) (an Analytics-specific term)	
Evaluate dialog box		

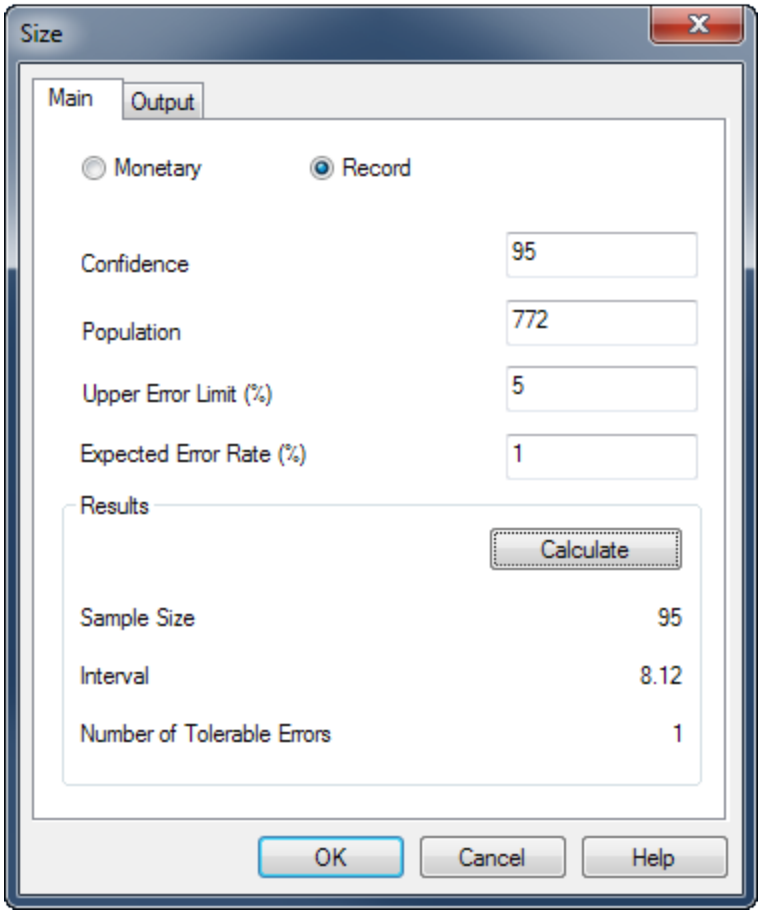
Size dialog box	Analytics term	Industry term
	**Errors**	misstatements

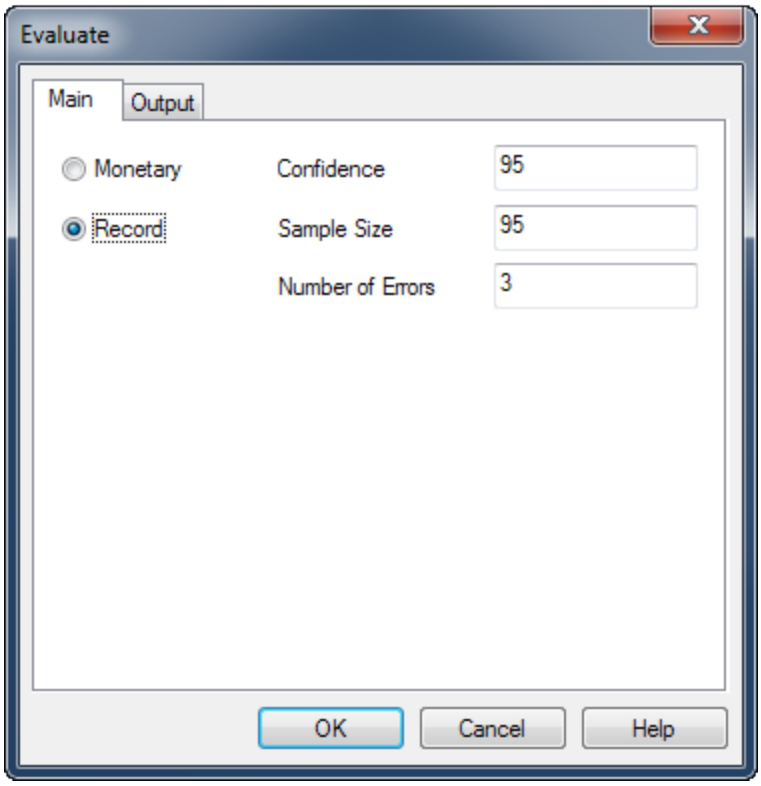
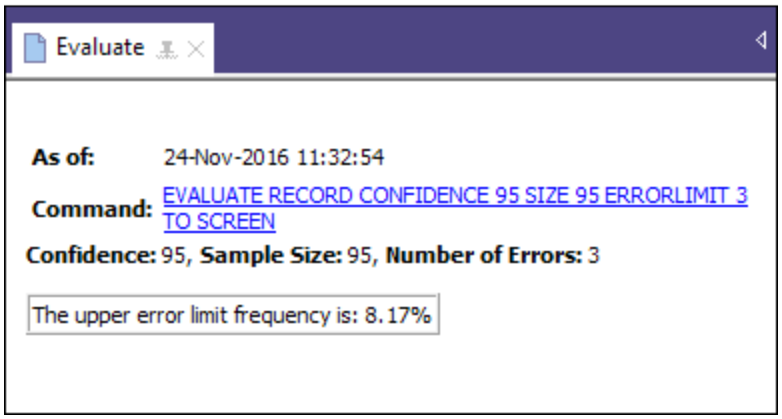
Evaluate results		
	**Basic Precision**	basic precision basic allowance for sampling risk
	Most Likely Error	projected misstatement
	Upper Error Limit	upper misstatement limit

	Item	Error	Most Likely Error	Upper Error Limit
Basic Precision				18,850.00
	1,839.33	1,163.35	3,974.12	6,954.71
	3,409.82	900.00	1,658.44	2,570.58
	392.41	12.85	205.76	300.41
	6,730.56	27.00	27.00	27.00
	206.69	-3.00	-91.20	0.00
Totals			5,774.12	28,702.70

Record sampling terminology

The table below lists the record sampling terminology used in Analytics and provides the equivalent industry term or terms.

Size dialog box	Analytics term	Industry term
	**Confidence**	confidence confidence level reliability
	Population	population population size
	Upper Error Limit (%)	tolerable deviation rate
	Expected Error Rate (%)	expected population deviation rate estimated population deviation rate
	Sample Size	sample size
	Interval	sampling interval
	Number of Tolerable Errors (an Analytics-specific term)	
Evaluate dialog box		

Size dialog box	Analytics term	Industry term
	**Number of Errors**	number of deviations
Evaluate result		
	**Upper error limit frequency**	computed upper deviation rate

Record sampling (attributes sampling)

Record sampling is a statistical sampling method for estimating the rate of deviation from a prescribed control in an account or class of transactions.

If your analysis of the sampled data will yield a yes/no or pass/fail result for each record, then you should use record sampling.

> **Tip**
> For a hands-on introduction to the end-to-end process of record sampling in Analytics, see "Record sampling tutorial" on page 967.

How it works

Record sampling allows you to select and analyze a small subset of a population, and based on the result estimate the rate of error or deviation from an internal control for the entire population.

You can then compare the estimated rate to the rate that you judge is acceptable, and make a determination regarding the control.

Record sampling supports making this sort of statement:

- *There is a 95% probability that the deviation rate from the prescribed control does not exceed 4.35%, which is below the tolerable deviation rate of 5.0%. Therefore the prescribed control is operating effectively.*

Overview of the record sampling process

> **Caution**
> Do not skip calculating a valid sample size.
>
> If you go straight to drawing a sample of records, and guess at a sample size, there is a high likelihood that the projection of your analysis results will be invalid, and your final conclusion flawed.

The record sampling process involves the following general steps:

1. Calculate the required sample size
2. Choose a sample selection method:
   - Fixed interval
   - Cell
   - Random

3. Draw the sample of records
4. Perform your intended audit procedures on the sampled data.
5. Evaluate whether the observed rate of control deviation in the sampled data represents an acceptable or unacceptable deviation rate in the entire population.

How record sampling selects records

Record sampling uses the following process for selecting sample records from an Analytics table:

- When you specify record sampling, the sampling unit is an individual record with a unique record number. You do not specify a particular field.
- Using one of the sample selection methods, Analytics selects samples from among the record numbers. The selected records are included in the sampling output table.

Example

In a table with 100 records, Analytics could select the following record numbers:

- 9
- 13
- 40
- 52
- 78
- 91
- 99

Unbiased sample selection

Record sampling is unbiased and it is not based on the amounts contained in a record. Each record has an equal chance of being selected for inclusion in the sample. A record containing a $1000 amount, a record containing a $250 amount, and a record containing a $1 amount all have the same chance of being selected.

In other words, the probability that any given record will be selected has no relation to the size of the amount it contains.

Considerations

Record sampling is appropriate for use with controls testing that results in a yes/no, or pass/fail, result. In controls testing, you are more concerned with the rate of errors in the total population than with the cumulative monetary amount of the errors.

Because record sampling does not consider the amounts contained by records, there is a significant chance that large monetary transactions may be excluded from the sample.

Record sampling tutorial

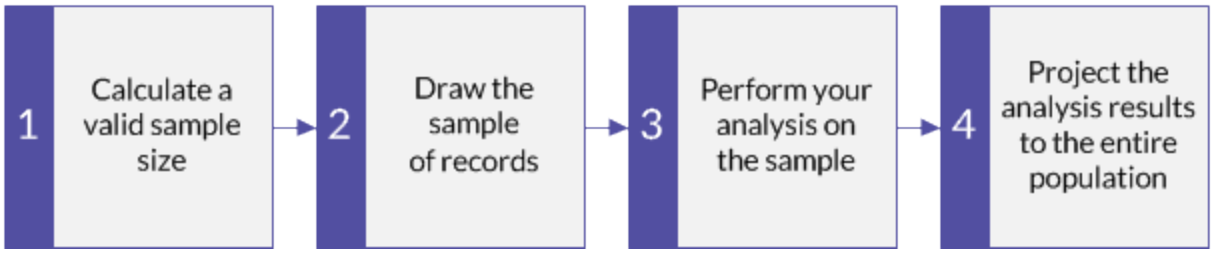
This tutorial introduces you to the end-to-end process of record sampling in Analytics.

Estimated time – 20 minutes

Summary – You will draw a sample of records from a vouchers table, and identify control deviations in the sample. Based on the sample results, you will make a statistical estimate of the deviation rate for the entire table.

You then use the statistical estimate to judge whether the voucher control process is operating effectively.

Main tasks – To perform record sampling correctly, you need to do four main tasks:



The tutorial leaves out optional aspects of record sampling, and focuses on a single path, so you can quickly get a basic understanding of how record sampling in Analytics works.

> **Tip**
> For simple definitions of sampling terms, see "A word about terminology" on page 951.

Record sampling scenario

Testing the voucher control process

The scenario

You are examining a vouchers table with over 5000 records. You want to pull hard copies of a sample of the vouchers to confirm that they match the system entries, and that the voucher control process is working effectively.

You will check the hard copies to confirm:

- vouchers have been approved
- the same person did not create and approve a voucher

How do you proceed?

How many hard copies should you pull? How do you decide which ones to pull? How do any control deviations you find in the sample relate to the entire population of vouchers?

You can use Analytics record sampling to get answers to these questions.

Analytics table used in the scenario

This scenario uses the `Vouchers` table in the `ACL_Rockwood.acl` sample data file included with Analytics.

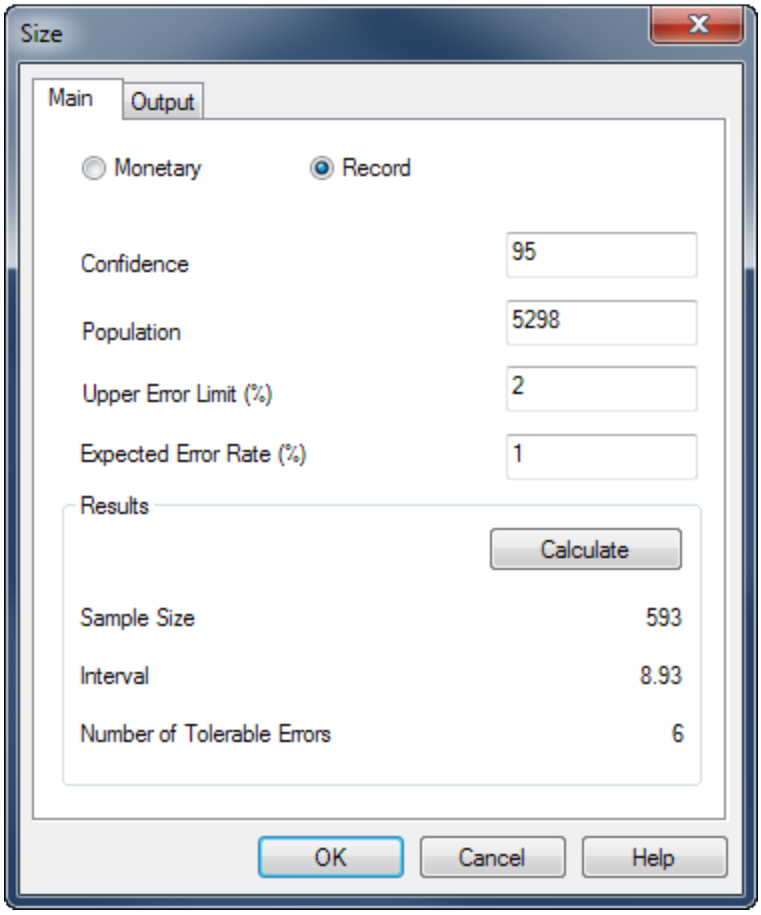
> **Note**
> Do not include the thousands separator, or the percentage sign, when you specify values. These characters prevent the command from running, or cause errors.

1 Calculate a valid sample size

> **Note**
> In a production environment, the values you specify to calculate a valid sample size are dependent on your professional judgment.

1. In `ACL_Rockwood.acl`, open the `Vouchers` table, located in the `Acquisitions_Payment` folder.
2. Select **Sampling > Record/Monetary Unit Sampling > Calculate Size**.
3. Select **Record**.
4. Specify the input values exactly as they appear in the screen below and click **Calculate** to calculate the sample size.

   After reviewing the results, you can click **OK** to finalize the sample size calculation, or you can experiment with specifying different values (see below).



What the input values mean

Confidence	You want a 95% degree of confidence that the sample you are going to draw is representative of the entire population. Put another way: if you drew the sample 100 times, it would be representative 95 times, and unrepresentative only 5 times.
Population	The total number of records in the `Vouchers` table.
Upper Error Limit (%)	A maximum of 2% of the vouchers can lack proper approval and you still consider the control effective.
Expected Error Rate (%)	You expect that 1% of the vouchers lack proper approval.

What the results mean

Sample Size	You should pull 593 hard copy vouchers.

Interval	If you use one of the interval methods of sample selection, the records selected are either: ◦ every 9th record ◦ a randomly selected record from each block of 9 records **Note** 8.93 is rounded up to 9.
Number of Tolerable Errors	**Note** The record sampling tutorial does not use this number, which provides an alternative method for evaluating control deviations. What the number means: ◦ As you examine hard copy vouchers in the sample, if more than 6 vouchers violate the control you can consider the control ineffective.

Learn more: experiment by specifying different values

Change a value in any of the following fields in the **Size** dialog box, click **Calculate**, and notice how the results change. Change only one value at a time so it is easier to see how the change affects the results.

- **Confidence**
- **Upper Error Limit (%)**
- **Expected Error Rate (%)**

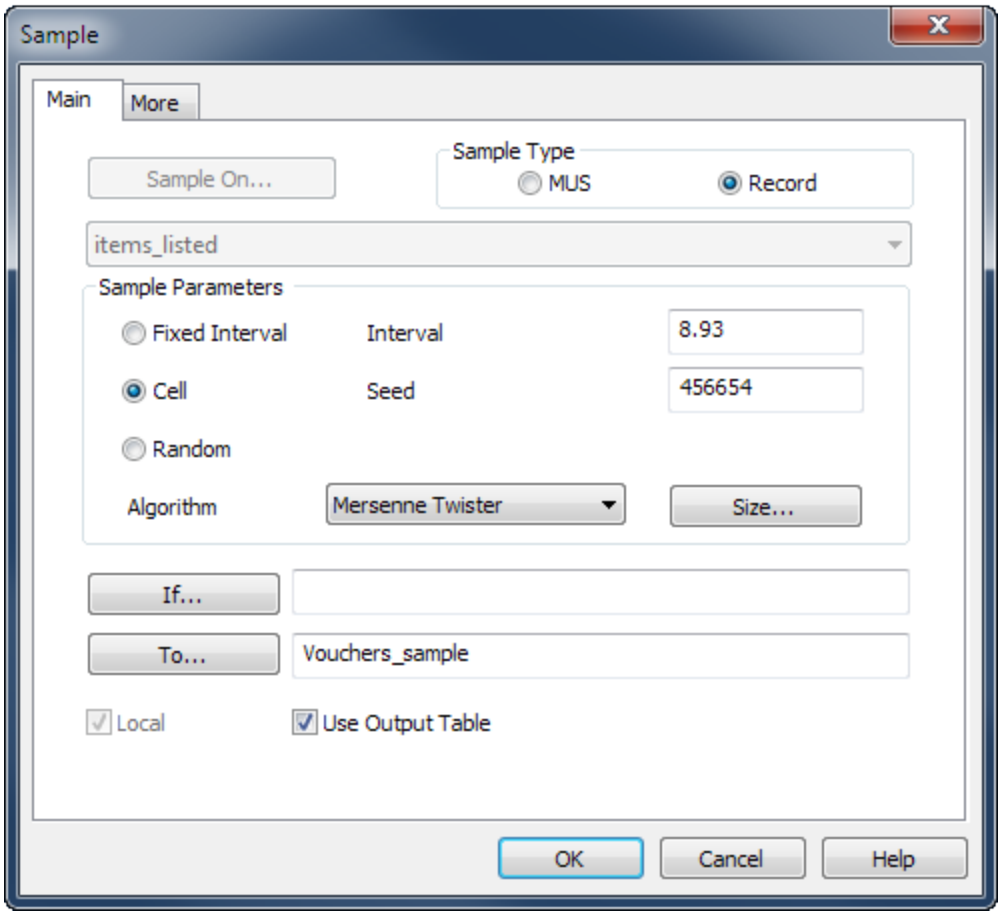
More stringent requirements increase the sample size. More lenient requirements decrease the sample size.

Reset the values to match the screen above and click **OK**. Pin the **Size** tab with the results of the sample size calculation.

2 Draw the sample of records

1. Return to the Vouchers table.
2. Select **Sampling > Record/Monetary Unit Sampling > Sample**.
3. Select **Record**.
4. Specify the input values exactly as they appear in the screen below and click **OK** to draw the

sample of records.



What the input values mean

Cell	You are using the cell selection method for drawing the sample of records. With the cell selection method, each selected record is randomly chosen from within identically sized cells or blocks of records.
Interval	The size of each cell is 9 records. 8.93 is rounded up to 9.
Seed	A seed value of 456654 is used to initialize the random number generator in Analytics. You can specify any seed value you want. The random number generator specifies which record number is selected from each cell.
Algorithm	The random number generator uses the Mersenne-Twister algorithm to generate random numbers.
To	The sample of records drawn from the `Vouchers` table is output to a new table called `Vouchers_sample`.

> **Note**
> Because Analytics rounds up the **Interval** to 9, the actual number of records drawn is slightly less than the calculated sample size of 593.

3 Perform your analysis on the sample

For the purposes of the tutorial, assume that you do the following:

1. Pull the hard copies for the voucher numbers that appear in the `Vouchers_sample` table.
2. Examine each voucher and record any vouchers that lack proper approval.

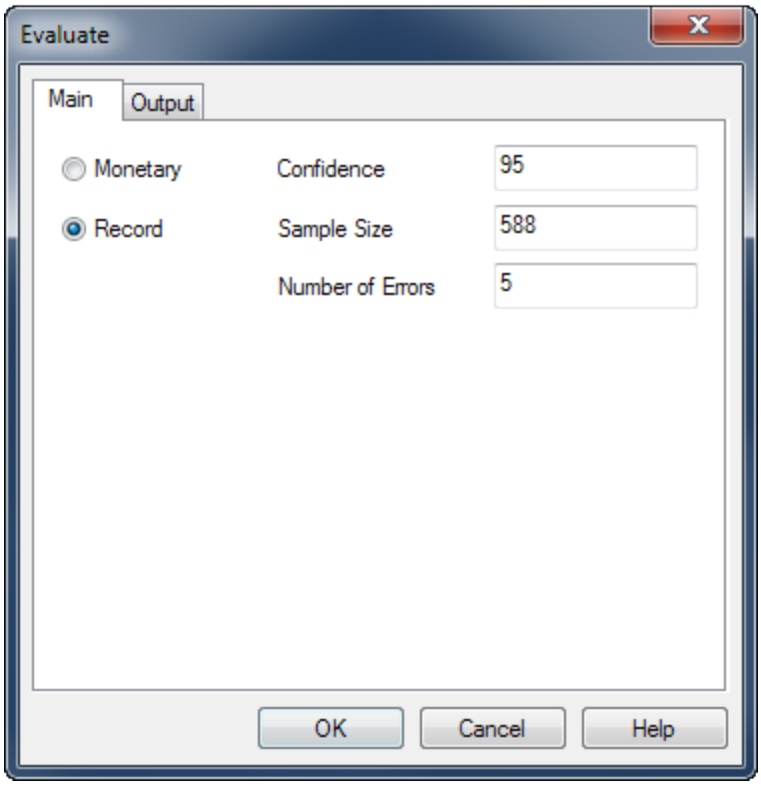
4 Project the analysis results to the entire population

1. Select **Sampling > Record/Monetary Unit Sampling > Evaluate**.

   > **Note**
   > The menu option is disabled if a table is not open.

2. Select **Record**.
3. Specify the input values exactly as they appear in the screen below and click **OK** to project the

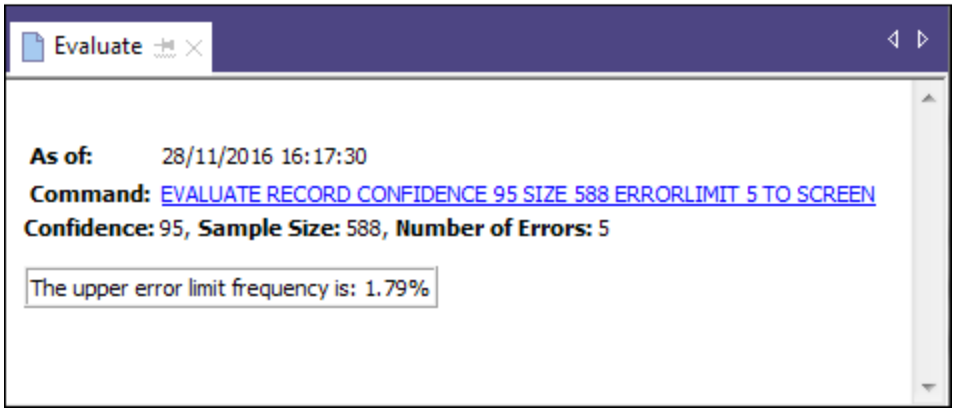
results.



What the input values mean

Confidence	The same degree of confidence you specified when you calculated sample size.
Sample Size	The actual number of records in the sample you drew - that is, the number of records in the `Vouchers_sample` table.
Number of Errors	When you examined the hard copies, the number of vouchers that lacked proper approval.

What the projected result means



Upper error limit frequency	The maximum deviation rate for the entire population of vouchers, projected with a 95% degree of confidence. Put another way: There is a 95% probability that the number of vouchers that lack proper approval in the `Vouchers` table does not exceed 1.79%, or 95 vouchers. *Because 1.79% is less than the 2.00% you specified for the **Upper Error Limit (%)** when you calculated the sample size, you can conclude that the voucher control is operating effectively.* For a detailed explanation, see "What “Upper error limit frequency” tells you" on page 992.

Learn more: experiment by specifying different values

Rerun the evaluate command with a different **Number of Errors** to see how the result changes.

The table below summarizes some different results.

Number of Errors (in the sample)	Upper error limit frequency (projected maximum)	Vouchers lacking approval in `Vouchers` table (projected maximum)	Conclusion
5	1.79%	95 (0.0179 x 5298)	The voucher control is operating effectively. 1.79% **<** the upper error limit of 2.00%
6	2.02%	107 (0.0202 x 5298)	In strict terms, the voucher control is **not** operating effectively. However, 2.02% is very close to the upper

Number of Errors (in the sample)	Upper error limit frequency (projected maximum)	Vouchers lacking approval in `Vouchers` table (projected maximum)	Conclusion
			error limit of 2.00%. **Note** This example demonstrates the difference between using **Upper error limit frequency** and **Number of Tolerable Errors** to evaluate control deviations. If you use the more lenient **Number of Tolerable Errors** method, the voucher control is operating effectively. The number of observed errors in the "Number of Errors" column to the left is 6, which does not exceed the **Number of Tolerable Errors** of 6 reported when you calculated sample size.
7	2.24%	119 (0.0224 x 5298)	The voucher control is **not** operating effectively. 2.24% **>** the upper error limit of 2.00%
10	2.89%	153 (0.0289 x 5298)	The voucher control is **not** operating effectively. 2.89% **>** the upper error limit of 2.00%

Calculating sample size for a record sample

Before sampling a set of data, you must calculate the statistically appropriate sample size, and other values required by the subsequent sample and evaluate operations.

The **Calculate Sample Size** feature in Analytics calculates the required values for you based on input values you provide.

The importance of calculating a sample size

Calculating an appropriate sample size is critical to the validity of the subsequent sample. If the sample is not valid, or representative, you cannot reliably project the results of audit procedures you perform on the sample to the entire population.

Do not skip calculating a sample size, or guess at a sample size.

Most of the input values you use to calculate sample size are based on your professional judgment. Ensure that you fully understand the implications of the values before relying on the results of sampling in a production environment. Consult audit sampling resources, or an audit sampling specialist, if you are in doubt.

How input values affect sample size

Input values affect the sample size calculated by Analytics. You can use the **Calculate** button in the **Size** dialog box to experiment with how the different input values affect the sample size.

The table below summarizes the effect of input values on sample size.

> **Caution**
> In a production environment, do not manipulate input values solely to achieve a smaller sample size. Input values should be based on your professional judgment about what is most appropriate for the data being sampled and the audit objective.

Increasing this input value:	Decreases sample size	Increases sample size
Confidence		✔
Population	Has no effect on sample size	

Increasing this input value:	Decreases sample size	Increases sample size
Upper Error Limit (%)	✔	
Expected Error Rate (%)		✔

Steps

> **Note**
> Do not include the thousands separator, or the percentage sign, when you specify values. These characters prevent the command from running, or cause errors.

1. Select **Sampling > Record/Monetary Unit Sampling > Calculate Size**

   > **Note**
   > The menu option is disabled if a table is not open.

2. On the **Main** tab, select **Record**.
3. Enter the input values to use for calculating the sample size:
   - **Confidence**
   - **Population**
   - **Upper Error Limit (%)**
   - **Expected Error Rate (%)**

   > **Note**
   > The input values are explained in detail below.

4. (Optional) Click **Calculate** to see a preview of the output results.

   > **Tip**
   > Clicking **Calculate** instead of **OK** allows you to experiment with different input values before outputting the results.

   > **Note**
   > The output results are explained in detail below.

5. On the **Output** tab:
   a. In the **To** panel, select one of the following:
      - **Screen** - displays the results in the Analytics display area

        > **Tip**
        > You can click any linked result value in the display area to drill down to the associated record or records in the source table.

      - **File** - saves or appends the results to a text file

        The file is saved outside Analytics.

   b. If you selected **File** as the output type, do one of the following:
      - Enter a file name in the **Name** text box.
      - Click **Name** and enter the file name, or select an existing file in the **Save** or **Save File As** dialog box to overwrite or append to the file.

        If Analytics prefills a file name, you can accept the prefilled name, or change it.

        You can also specify an absolute or relative file path, or navigate to a different folder, to save or append the file in a location other than the project location. For example: `C:\Results\Output.txt` or `Results\Output.txt`.

        > **Note**
        > **ASCII Text File** or **Unicode Text file** (depending on which edition of Analytics you are using) is the only option for **File Type**.

6. Click **OK**.
7. If the overwrite prompt appears, select the appropriate option.

Size dialog box inputs and results

The tables below provide detailed information about the input values and output results in the **Size** dialog box.

Main tab – input values

Input values – Size dialog box	Description
Confidence	Your desired confidence level that the resulting sample is representative of the entire population. For example, entering 95 means that you want to be confident that 95% of the time the sample will in fact be representative. Confidence is the complement of "sampling risk". A 95% confidence level is the same as a 5% sampling risk.
Population	The number of records in the data set you are sampling.

Input values - Size dialog box	Description
	Note In record sampling, the population size does not affect the resulting sample size. For example, if the other input values remain the same, the same statistically valid sample size is calculated for populations of 150,000 or 1,000,000 records. The resulting interval value does increase in direct relation to the population size.
Upper Error Limit (%)	The maximum rate of deviation from a prescribed control that can occur and you still consider the control effective. For example, entering 5 means that the deviation rate must be greater than 5% for you to consider the control not effective.
Expected Error Rate (%)	The rate of deviation from a prescribed control that you expect to find. For example, entering 1 means that you expect the deviation rate to be 1%. **Note** The **Expected Error Rate (%)** you specify must be less than the **Upper Error Limit (%)**. If the difference between them is too small, the error message **Error rate too high for calculation** appears. In audit sampling terms, the degree of sampling precision represented by the difference is too small to be calculated for the confidence level you specified.

Main tab - output results

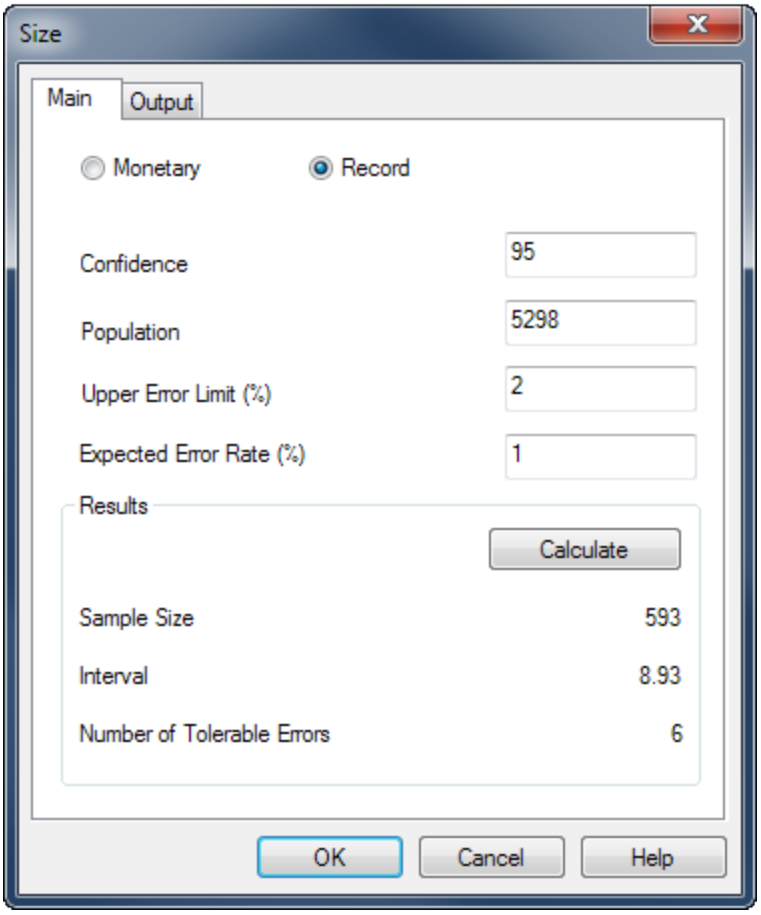
Output results - Size dialog box	Description
Sample Size	The required sample size.
Interval	The interval value - required for the fixed interval and the cell selection methods.
Number of Tolerable Errors	The maximum number of errors or deviations that can occur in the resulting sample without exceeding the **Upper Error Limit (%)**. For more information, see "Number of Tolerable Errors" on page 981.

An example of inputs and results

Calculating the size of a record sample for the Vouchers table

The figure below provides an example of input values and output results when calculating sample size for record sampling.

The table contains 5298 records. Based on the other input values, the required sample size is 593 records.



The calculation is based on the `Vouchers` table in `ACL_Rockwood.acl` (`ACL DATA\Sample Data Files\ACL_Rockwood\ACL_Rockwood.acl`).

Number of Tolerable Errors

> **Note**
>
> If you intend to use the evaluation feature in Analytics, you do not need to use the value reported by **Number of Tolerable Errors**. Instead, you use the **Upper error limit frequency** calculated by the evaluation feature. For more information, see "Evaluating errors in a record sample" on page 988.

Number of Tolerable Errors provides one way of evaluating deviation in a population.

If you use this method, you know in advance the threshold value reported by Analytics, before you begin audit procedures on the sampled data. If cumulative errors you observe in the course of performing the procedures exceed the Analytics reported value, you know at that point that the deviation rate from a prescribed control is unacceptably high.

After performing your control tests on the sampled data you can compare the number of errors or deviations you found to the **Number of Tolerable Errors**. If the number of observed errors is less than or equal to the **Number of Tolerable Errors** you can consider the control is effective, for your specified confidence level.

Statistical validity of sample sizes generated by Analytics

Analytics generates statistically valid sample sizes for most analyses. Exceptions may apply in the following situations:

- You are sampling data sets of less than 1000 records.
- Your organization has in-house sampling experts who can define sample sizes precisely tailored to your needs.
- Your organization has mandated the use of another sampling tool or methodology.

Poisson distribution versus binomial distribution

Two commonly used methods of generating sample sizes are the Poisson and the binomial distributions. Analytics generates sample sizes using the Poisson distribution.

For typical data sets of a thousand or more records, the Poisson and the binomial distributions generate nearly identical sample sizes. For populations of under a thousand records, sample sizes determined with the Poisson distribution tend to be slightly larger and therefore more conservative than sizes determined with the binomial distribution. The binomial distribution adjusts the sample size

downward for small populations but the Poisson distribution does not. With very small populations, the sample size generated by the Poisson distribution can actually exceed the population size.

When calculating sample sizes in Analytics, recognize that for record sampling of small data sets, the sample size may be larger than you need. This larger sample size does not present an obstacle to analysis because it is common practice to manually oversample small populations.

Performing record sampling

You can create a new table that contains a representative sample of the data in the active table.

Record sampling is appropriate if you are interested in the rate of deviation from a prescribed control.

> **Note**
> This procedure does not include filtering (IF statements) or scope parameters because applying these options compromises the validity of a sample.

Steps

> **Note**
> Do not include the thousands separator, or the percentage sign, when you specify values. These characters prevent the command from running, or cause errors.

1. In the Navigator, open the table you want to draw a sample from.
2. Select **Sampling > Record/Monetary Unit Sampling > Sample**.
3. On the **Main** tab, select **Record**.
4. In the **Sample Parameters** panel, specify a sample selection method:
   - **Fixed Interval**
   - **Cell**
   - **Random**
5. Enter the sample parameters for the selection method you chose:

Selection method	Sample parameters
Fixed Interval	◦ **Interval** ◦ **Start** (optional)
Cell	◦ **Interval** ◦ **Seed** (optional) ◦ **Algorithm** - leave **Mersenne Twister** selected
Random	◦ **Size** ◦ **Seed** (optional) ◦ **Population** (optional) - prefilled with the number of records in the table ◦ **Algorithm** - leave **Mersenne Twister** selected

> **Note**
> Sample parameters are explained in detail below.

6. In the **To** text box, specify the name of the Analytics table that will contain the output results.

7. On the **More** tab, select one of the following:
   - **Record** -The entire record is included in the output table.
   - **Fields** - Only the selected fields are included in the output table.
8. If you chose **Fields**, select the field(s) to include in the output table from the **Extract Fields** list.
9. Optional. Select **Report Selection Order** if you want to add the ORDER field to the output results.

> **Note**
> **Report Selection Order** is available only if both the **Random** selection method and **Fields** output are selected.

10. Click **OK**.

Sample dialog box options

The tables below provide detailed information about the options in the **Sample** dialog box.

Main tab

Options - Sample dialog box	Description
MUS Record	The sample type: ◦ **MUS** - Monetary unit sampling Appropriate if you are interested in the total amount of monetary misstatement in a file. ◦ **Record** - Record sampling Appropriate if you are interested in the rate of deviation from a prescribed control.
Sample On	Not used for record sampling.
Fixed Interval	Specifies that the fixed interval method is used for sample selection. Samples are selected based on an interval value and a start number that you specify. For more information, see "Fixed interval selection method" on page 953. If you selected **Fixed Interval** enter the following values: ◦ **Interval** (required) - the interval value that was generated by calculating the sample size > **Note** > If you have not already calculated the sample size, you can click **Size** to open the **Size** dialog box. For more information, see "Calculating sample size for a record sample" on page 976. ◦ **Start** (optional) - a start number that is greater than zero and less than the interval

Options - Sample dialog box	Description
	value **Tip** Enter a start number of '0', or leave the start number blank, if you want Analytics to randomly select a start number.
Cell	Specifies that the cell method is used for sample selection. The data set is divided into multiple equal-sized cells or groups, and one sample is randomly selected from each cell. The interval value dictates the size of each cell. For more information, see "Cell selection method" on page 954. If you selected **Cell** enter the following values: ◦ **Interval** (required) - the interval value that was generated by calculating the sample size **Note** If you have not already calculated the sample size, you can click **Size** to open the **Size** dialog box. For more information, see "Calculating sample size for a record sample" on page 976. ◦ **Seed** (optional) - can be any number This number is used to initialize the random number generator in Analytics. **Tip** Enter a seed value of '0', or leave the seed blank, if you want Analytics to randomly select a seed value. ◦ **Algorithm** (required) - leave **Mersenne Twister** selected Only select **Default** if you require backward compatibility with Analytics scripts or sampling results created prior to Analytics version 12.
Random	Specifies that the random method is used for sample selection. Samples are randomly selected from the entire data set. For more information, see "Random selection method" on page 956. If you selected **Random** enter the following values: ◦ **Size** (required) - the sample size that was calculated by Analytics **Note** If you have not already calculated the sample size, you can click **Size** to open the **Size** dialog box. For more information, see "Calculating sample size for a record sample" on page 976. ◦ **Seed** (optional) - can be any number This number is used to initialize the random number generator in Analytics.

Options - Sample dialog box	Description
	Tip Enter a seed value of '0', or leave the seed blank, if you want Analytics to randomly select a seed value. ◦ **Population** (optional) -prefilled with the number of records in the table, which is the population from which the sample will be selected ◦ **Algorithm** (required) - leave **Mersenne Twister** selected Only select **Default** if you require backward compatibility with Analytics scripts or sampling results created prior to Analytics version 12.
If	**Caution** Do not create an IF statement or filter records in the course of sampling. Doing so compromises the validity of the sample. For more information, see "Conditional sampling" on page 1086.
To	The name and location of the output table. ◦ **To save the output table to the Analytics project folder** - enter only the table name. ◦ **To save the output table in a location other than the project folder** - specify an absolute or relative file path, or click **To** and navigate to a different folder. For example: `C:\Results\Output.fil` or `Results\Output.fil`. Regardless of where you save the output table, it is added to the open project if it is not already in the project. If Analytics prefills a table name, you can accept the prefilled name, or change it.
Local	If you are connected to a server table, specifies where to save the output table. ◦ **Local selected** - saves the output table to the same location as the Analytics project, or to a specified path, or location you navigate to. ◦ **Local deselected** - saves the output table to the Prefix folder on AX Server.
Use output table	Specifies whether the Analytics table containing the output results opens automatically upon completion of the operation.

More tab

Options - Sample dialog box	Description
Scope panel	**Caution** Do not limit which records are processed in the course of sampling. Doing so compromises the validity of the sample. For more information, see "Conditional sampling" on page 1086.
Record Fields	Specifies whether the output table includes the entire record, or selected fields. If you choose **Fields**, do one of the following:

Options - Sample dialog box	Description
	◦ Select the field(s) to extract from the **Extract Fields** list. ◦ Click **Extract Fields** to select the field(s), or to create an expression. The order in which you select the fields is the order in which the columns appear in the results. If you are appending results to an existing Analytics table, the column selection and order must be identical to the column selection and order in the existing table.
Report Selection Order	(Optional) Adds the ORDER field to the output results. This field displays the order in which each record is randomly selected. **Note** **Report Selection Order** is available only if both the **Random** selection method and **Fields** output are selected.
Append To Existing File	Specifies that the output results are appended (added) to the end of an existing Analytics table. **Note** Leaving **Append To Existing File** deselected is recommended if you are uncertain whether the output results and the existing table have an identical data structure. For more information about appending and data structure, see "Appending output results to an existing table" on page 200.
OK	Executes the operation. If the overwrite prompt appears, select the appropriate option. If you are expecting the **Append** option to appear and it does not, click **No** to cancel the operation and see "Appending output results to an existing table" on page 200.

Evaluating errors in a record sample

After you have performed your audit procedures on the set of sampled data you can use Analytics to:

- project any errors you found to the entire population
- calculate an upper limit on the deviation rate

Even if you found no errors, you still use the evaluation feature to calculate the basic allowance for sampling risk.

> **Note**
> Evaluating errors requires input of some of the values previously generated by calculating sample size.

How evaluation and comparison work

When you evaluate, Analytics uses a statistical formula to project the errors you found in the sample to the entire population, and calculates the **Upper error limit frequency** (computed upper deviation rate).

You compare the calculated value to the **Upper Error Limit (%)** that you decided upon earlier when you calculated sample size. Based on the comparison, you decide if a prescribed control is operating effectively.

Comparison	Conclusion
Upper error limit frequency is less than or equal to **Upper Error Limit (%)**	The prescribed control is operating effectively
Upper error limit frequency is greater than **Upper Error Limit (%)**	The prescribed control is not operating effectively

Steps

> **Note**
> Do not include the thousands separator, or the percentage sign, when you specify values. These characters prevent the command from running, or cause errors.

1. Select **Sampling > Record/Monetary Unit Sampling > Evaluate**.

   > **Note**
   > The menu option is disabled if a table is not open.

2. On the **Main** tab, select **Record**.
3. Enter the input values to use for evaluating errors:
   - **Confidence**
   - **Sample Size**

     > **Note**
     > Enter the actual sample size as drawn, which might differ from the sample size initially calculated by Analytics.

   - **Number of Errors**

     > **Note**
     > The input values are explained in detail below.

4. On the **Output** tab:
   a. In the **To** panel, select one of the following:
      - **Screen** - displays the results in the Analytics display area

        > **Tip**
        > You can click any linked result value in the display area to drill down to the associated record or records in the source table.

      - **File** - saves or appends the results to a text file

        The file is saved outside Analytics.

   b. If you selected **File** as the output type, do one of the following:
      - Enter a file name in the **Name** text box.
      - Click **Name** and enter the file name, or select an existing file in the **Save** or **Save File As** dialog box to overwrite or append to the file.

        If Analytics prefills a file name, you can accept the prefilled name, or change it.

        You can also specify an absolute or relative file path, or navigate to a different folder, to save or append the file in a location other than the project location. For example: `C:\Results\Output.txt` or `Results\Output.txt`.

        > **Note**
        > **ASCII Text File** or **Unicode Text file** (depending on which edition of Analytics you are using) is the only option for **File Type**.

5. Click **OK**.
6. If the overwrite prompt appears, select the appropriate option.

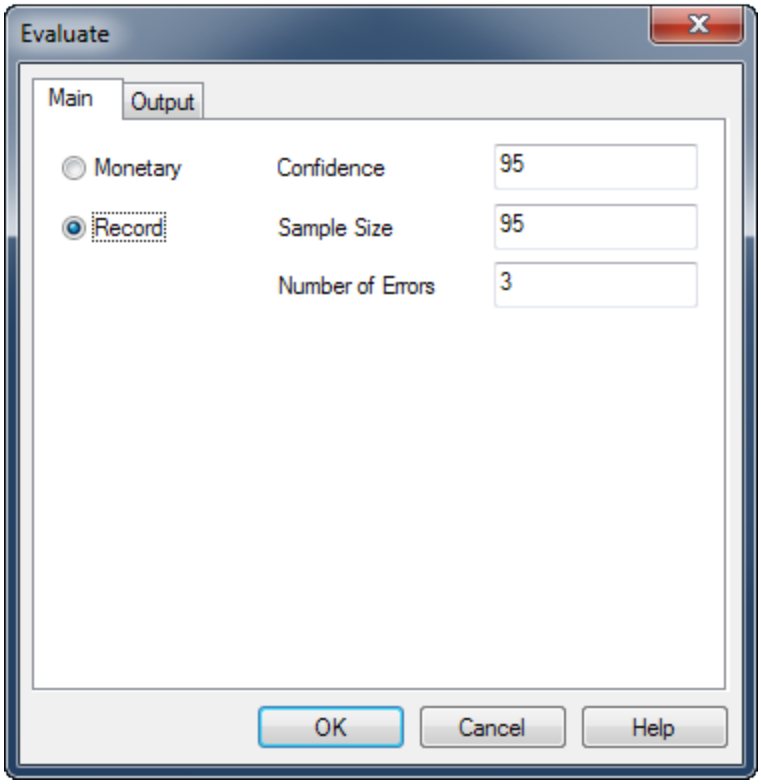
Evaluate dialog box inputs

The table below provides detailed information about the input values in the **Evaluate** dialog box.

Main tab – input values

Input values – Evaluate dialog box	Description
Confidence	The same confidence level that you entered when you calculated the sample size. For more information, see "Calculating sample size for a record sample" on page 976.
Sample Size	The number of records in the sample. **Note** **Sample Size** is the actual sample size as drawn, which might differ from the sample size initially calculated by Analytics.
Number of Errors	The total number of errors, or deviations, that you found in the sample.

The figure below shows an example of input values for evaluating errors in a record sample.

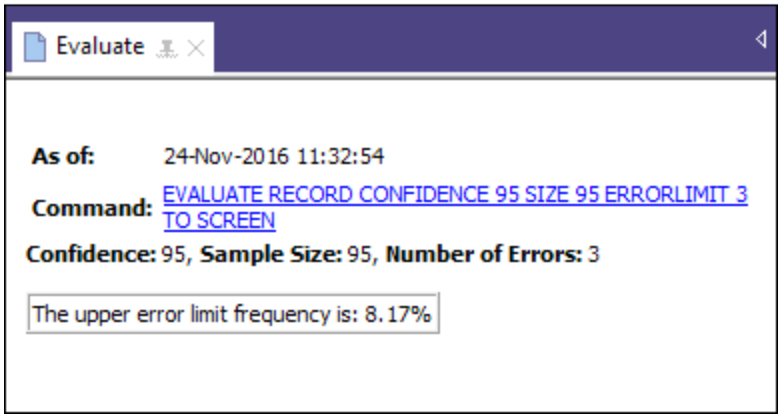


Result

Evaluating the errors you found in a record sample produces the following result:

Input value	Description
Upper error limit frequency (computed upper deviation rate)	An adjusted deviation rate that Analytics calculates is not exceeded in the entire data set, for the confidence level you specified.

The figure below shows the result of evaluating errors found in a record sample.



What “Upper error limit frequency” tells you

The **Upper error limit frequency**, when compared to the **Upper Error Limit (%)** that you decided upon when you calculated the sample size, tells you:

- Whether the control you are examining is operating effectively
- If the control is operating effectively, how effectively it is operating

Example

You evaluate the errors you found in a record sample and Analytics returns an **Upper error limit frequency** of 4.35%. This percentage is less than the **Upper Error Limit (%)** (tolerable deviation rate) of 5% that you specified earlier when you calculated the sample size, and specified a confidence level of 95%.

Based on this information, you can make the following statement:

There is a 95% probability that the actual deviation rate from the prescribed control in the entire population does not exceed 4.35%.

If the **Upper error limit frequency** is greater than 5%, as it is in the figure above, the prescribed control is probably not operating effectively. You need to decide upon further appropriate steps to meet your audit objective.

Monetary unit sampling

Monetary unit sampling is a statistical sampling method for estimating the total amount of monetary misstatement in an account or class of transactions.

Monetary unit sampling works best with financial data that has the following characteristics:

no misstatements, or only a small number of misstatements For example, less than 5% of the items are misstated.
more likelihood of overstatements than understatements
no zero dollar items

Monetary unit sampling is also known as:

- dollar-unit sampling
- probability-proportional-to-size sampling

> **Tip**
> For a hands-on introduction to the end-to-end process of monetary unit sampling in Analytics, see "Monetary unit sampling tutorial" on page 996.

How it works

Monetary unit sampling allows you to select and analyze a small subset of the records in an account, and based on the result estimate the total amount of monetary misstatement in the account.

You can then compare the estimated misstatement to the misstatement amount that you judge is material, and make a determination regarding the account.

Monetary unit sampling supports making this sort of statement:

- *There is a 95% probability that the misstatement in the account balance does not exceed $28,702.70, which is less than the tolerable misstatement of $29,000.00. Therefore the amounts in the account are fairly stated.*

Overview of the monetary unit sampling process

> **Caution**
>
> Do not skip calculating a valid sample size.
>
> If you go straight to drawing a sample of records, and guess at a sample size, there is a high likelihood that the projection of your analysis results will be invalid, and your final conclusion flawed.

The monetary unit sampling process involves the following general steps:

1. Calculate the required sample size
2. Choose a sample selection method:
   - Fixed interval
   - Cell
   - Random
3. Optionally specify one or more of the following options:
   - "Top stratum cutoff" on page 1019
   - "Subsampling" on page 1020
   - "Sample selection without repeats" on page 1020
4. Draw the sample of records
5. Perform your intended audit procedures on the sampled data.
6. Evaluate whether the observed levels of monetary misstatement in the sampled data represent an acceptable or unacceptable amount of misstatement in the account as a whole.

How monetary unit sampling selects records

Monetary unit sampling uses the following process for selecting sample records from an Analytics table:

- You specify a numeric field with monetary amounts as the basis for the sampling.
- The absolute value of all the amounts in the field is treated as a stream of monetary units, with each unit representing one cent ($0.01) of the absolute value.
- Using one of the sample selection methods, Analytics selects samples from among the monetary units. The records corresponding to the selected monetary units are included in the sampling output table.

Example

A table contains an "Amount" field with the values shown below. The field has an absolute value of $11.75, and therefore contains 1,175 monetary units.

If the sampling process selects monetary units 399 and 1,007, records 2 and 5 are included in the output table. Records 1, 3, and 4 are not included.

Record number	Amount	Cumulative balance (absolute)	Monetary units	Unit selected by Analytics
1	$3.50	$3.50	1 to 350	
2	($0.75)	$4.25	351 to 425	399
3	$1.25	$5.50	426 to 550	
4	$0.75	$6.25	551 to 625	
5	($5.50)	$11.75	626 to 1,175	1,007

A bias toward larger amounts

Monetary unit sampling intentionally creates a bias toward the selection of records containing larger amounts, whether positive or negative. Each monetary unit has an equal chance of selection, so a $1000 amount, which contains 100,000 monetary units, is four times more likely to be selected than a $250 amount, which contains 25,000 monetary units.

In other words, the probability that any given record will be selected is directly proportional to the size of the amount it contains.

Considerations

Monetary unit sampling is appropriate for use with substantive or misstatement testing. By biasing larger amounts, monetary unit sampling provides a high level of assurance that all significant amounts in a population are subject to testing. When testing for misstatement, it is the larger amounts that present the greatest risk of containing a material error.

If you choose a sampling method that biases large amounts, you may miss a potential problem related to small transactions. Problems with small transactions, once aggregated, may be material.

Monetary unit sampling tutorial

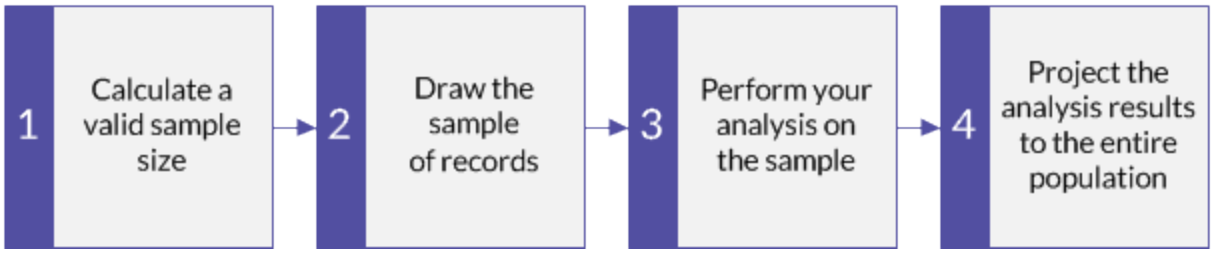
This tutorial introduces you to the end-to-end process of monetary unit sampling in Analytics.

Estimated time – 20 minutes

Summary – You will draw a sample of records from an invoices table, and identify misstatements in the sample. Based on the sample results, you will make a statistical estimate of the total amount of misstatement in the entire table.

You then use the statistical estimate to judge whether the invoice records as a whole are fairly stated.

Main tasks – To perform monetary unit sampling correctly, you need to do four main tasks:



The tutorial leaves out optional aspects of monetary unit sampling, and focuses on a single path, so you can quickly get a basic understanding of how monetary unit sampling in Analytics works.

> **Tip**
> For simple definitions of sampling terms, see "A word about terminology" on page 951.

Monetary unit sampling scenario

Detecting misstatement in accounts receivable

The scenario

You are examining an Invoices table with over 4000 records as part of confirming accounts receivable. You want to contact a sample of invoiced customers to confirm outstanding amounts in the account, and detect any misstatement.

You will use the customer contacts to confirm:

- receivable amounts exist
- receivable amounts are correctly recorded

How do you proceed?

How many customers should you contact? How do you decide which ones to contact? How do any misstatements you find in the sample relate to the entire account?

You can use Analytics monetary unit sampling to get answers to these questions.

Analytics table used in the scenario

This scenario uses the `Invoices` table in the `ACL_Rockwood.acl` sample data file included with Analytics.

> **Note**
> Most of the amounts in the `Invoices` table in `ACL_Rockwood.acl` have a status of "Paid". For this scenario, assume they have a status of "Outstanding" and a payment amount of $0.00.

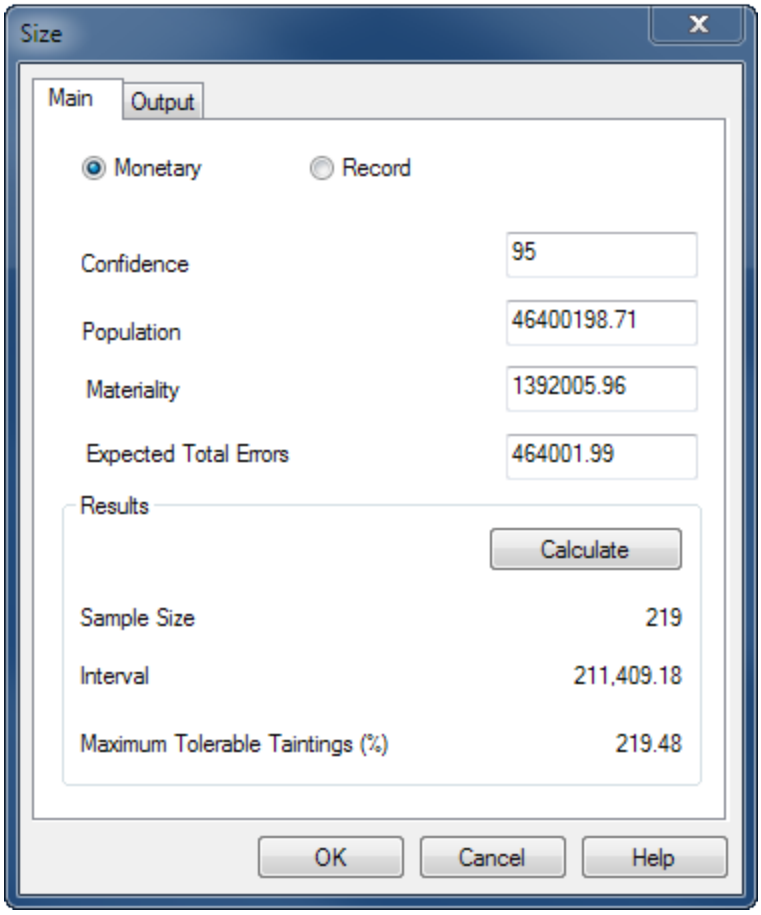
> **Note**
> Do not include the thousands separator, or the percentage sign, when you specify values. These characters prevent the command from running, or cause errors.

1 Calculate a valid sample size

> **Note**
> In a production environment, the values you specify to calculate a valid sample size are dependent on your professional judgment.

1. In `ACL_Rockwood.acl`, open the `Invoices` table, located in the `Sales_and_collection` folder.
2. Click the **Invoice Amount** header to select the column.
3. Select **Analyze > Profile** to calculate the absolute value of the **Invoice Amount** field. Pin the **Profile** tab with the results of the calculation.
4. Select **Sampling > Record/Monetary Unit Sampling > Calculate Size**.
5. Leave **Monetary** selected.
6. Specify the input values exactly as they appear in the screen below and click **Calculate** to calculate the sample size.

   After reviewing the results, you can click **OK** to finalize the sample size calculation, or you can experiment with specifying different values (see below).



What the input values mean

Confidence	You want a 95% degree of confidence that the sample you are going to draw is representative of the entire population. Put another way: if you drew the sample 100 times, it would be representative 95 times, and unrepresentative only 5 times.
Population	The absolute value of the **Invoice Amount** field in the `Invoices` table.
Materiality	The total amount of misstatement in the account must exceed $1,392,005.96 (3%) to be considered a material misstatement.
Expected Total Errors	You expect the total amount of misstatement in the account is $464,001.99 (1%).

What the results mean

Sample Size	You should contact 219 customers.

Interval	If you use one of the interval methods of sample selection, the records selected correspond to either: ◦ the monetary unit that occurs every 21,140,918 units ◦ the monetary unit that is randomly selected from each block of 21,140,918 units **Note** In Analytics, 1 monetary unit = 1 cent For a detailed explanation, see "How monetary unit sampling selects records" on page 994.
Maximum Tolerable Taintings (%)	**Note** The monetary unit sampling tutorial does not use this number, which provides an alternative method for evaluating misstatement. What the number means: ◦ As you confirm invoice amounts in the sample, if the sum of individual tainting percentages exceeds 219.48% you can consider the account to be materially misstated. In a misstated amount, tainting is the percentage of the book value that the misstatement represents. For a detailed explanation, see "Maximum Tolerable Taintings (%)" on page 1010.

Learn more: experiment by specifying different values

Change a value in any of the following fields in the **Size** dialog box, click **Calculate**, and notice how the results change. Change only one value at a time so it is easier to see how the change affects the results.

- **Confidence**
- **Materiality**
- **Expected Total Errors**

More stringent requirements increase the sample size. More lenient requirements decrease the sample size.

Reset the values to match the screen above and click **OK**. Pin the **Size** tab with the results of the sample size calculation.

2 Draw the sample of records

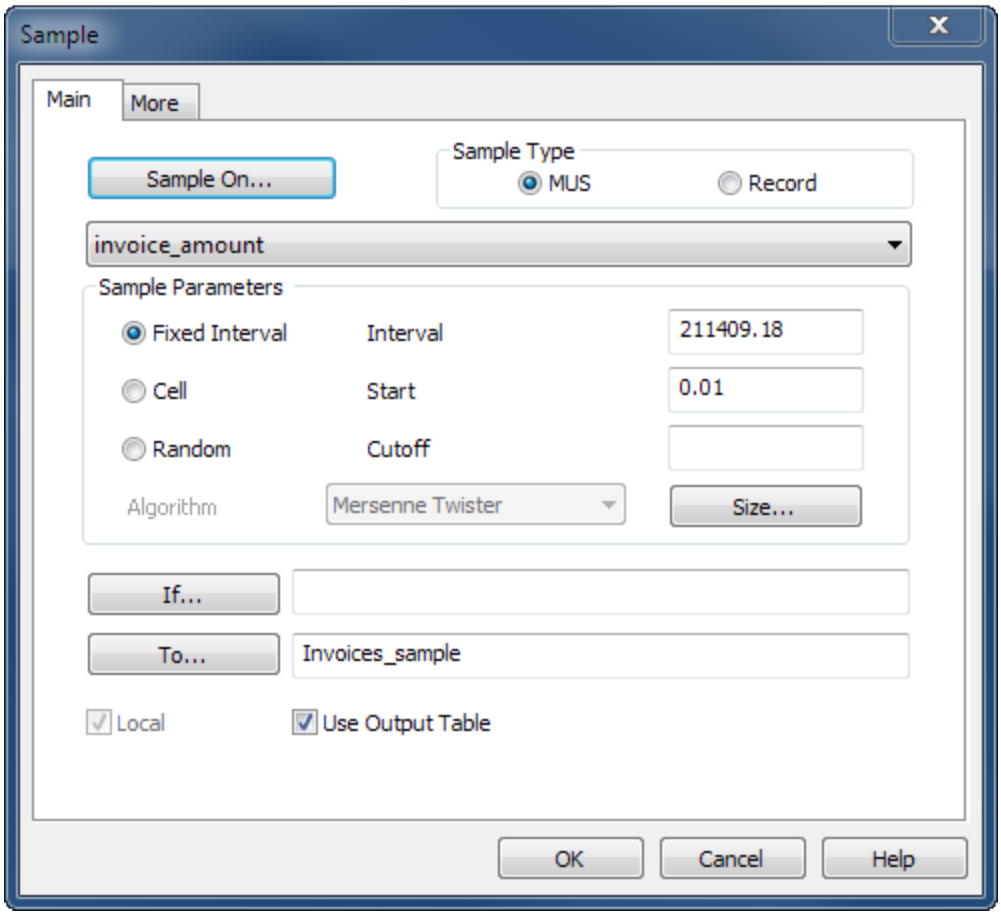
1. Return to the `Invoices` table.

   If the **Invoice Amount** column is still selected, click the top left corner of the table view to deselect it. (The blank area to the left of the first column header.)

2. Select **Sampling > Record/Monetary Unit Sampling > Sample**.

3. Leave **MUS** selected.
4. Specify the input values exactly as they appear in the screen below and click **OK** to draw the sample of records.

   Make sure the **invoice_amount** field is selected in the **Sample On** drop-down list.



What the input values mean

Sample On	The `invoice_amount` field contains the book values you are auditing.
Fixed Interval	You are using the fixed interval selection method for drawing the sample of records. With the fixed interval selection method, you specify the initial monetary unit that is selected, and all subsequent selections are a fixed interval or distance apart. For a detailed explanation, see "Fixed interval selection method" on page 953.
Interval	The interval between selected monetary units is $211,409.18, or 21,140,918 units.
Start	The initial monetary unit selected is $0.01, or unit 1.
To	The sample of records drawn from the `Invoices` table is output to a new table

	called `Invoices_sample`.

3 Perform your analysis on the sample

For the purposes of the tutorial, assume that you do the following:

1. Contact the customers that appear in the `Invoices_sample` table.
2. Confirm receivable amounts and record any misstatements.

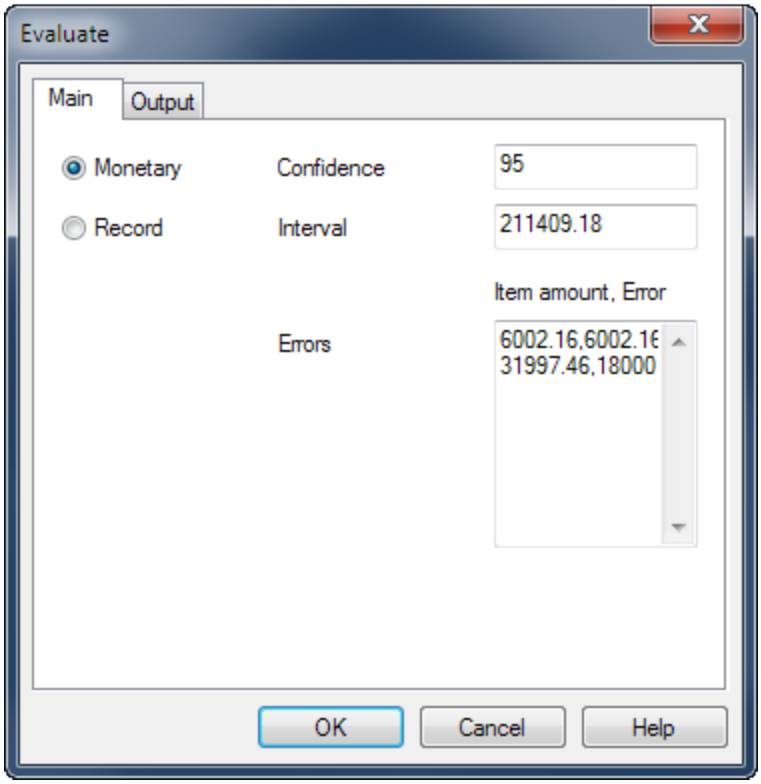
4 Project the analysis results to the entire population

1. Select **Sampling > Record/Monetary Unit Sampling > Evaluate**.

   > **Note**
   > The menu option is disabled if a table is not open.

2. Leave **Monetary** selected.
3. Specify the input values exactly as they appear in the screen below and click **OK** to project the results.

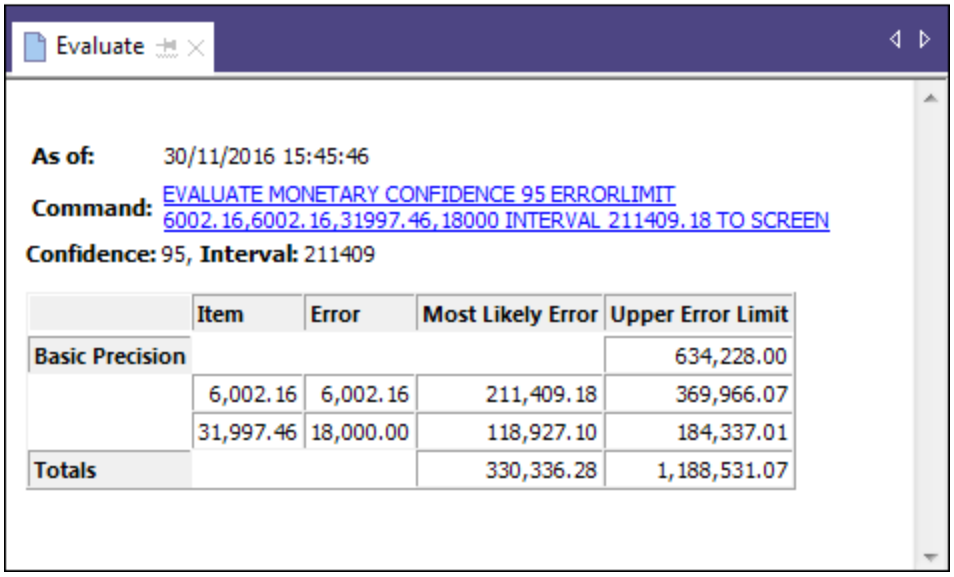
   > **Note**
   > Use a comma between **Item amount** and **Error**, but do not use commas in the amounts. Enter each amount and error on a separate line.



What the input values mean

Confidence	The same degree of confidence you specified when you calculated sample size.
Interval	The interval you used when you drew the sample.
Errors	When you confirmed receivable amounts, misstatements entered in the format: *book amount,misstatement amount* In this example: ◦ One invoiced customer had no record of the amount $6,002.16. ◦ Another invoiced customer had a hard copy invoice amount of 13,997.46, rather than 31,997.46, indicating a probable data entry error in the `Invoices` table.

What the projected results mean



Basic Precision	Basic allowance for sampling risk: $634,228.00. Analytics calculates a basic allowance for sampling risk because even if you found no misstatements in the sample, you cannot be sure that no misstatements exist in the account as a whole.
Most Likely Error	Total projected misstatement for the account: $330,336.28. A projection to the entire account of actual misstated amounts you found in the sample.
Upper error limit (Total)	The maximum amount of misstatement for the entire account, projected with a 95% degree of confidence: $1,188,531.07 Put another way: There is a 95% probability that the total amount of misstatement in the `Invoices` table does not exceed $1,188,531.07. *Because $1,188,531.07 is less than the $1,392,005.96 you specified for **Materiality** when you calculated the sample size, you can conclude that the accounts receivable are not materially misstated.* For a detailed explanation, see "What the “Upper Error Limit” tells you" on page 1026.

Learn more: experiment by specifying different values

Rerun the evaluate command with different values in the **Errors** field to see how the result changes.

The table below summarizes different results.

Misstatements (in the sample)	Upper error limit (projected maximum)	Conclusion
6,002.16, 6,002.16 31,997.46, 18,000.00	1,188,531.07	The account is not materially misstated. $1,188,531.07 **<** the materiality threshold of $1,392,005.96
6,002.16, 6,002.16 31,997.46, 18,000.00 13,225.50, 8,644.34	1,392,005.84	In strict terms, the account is not materially misstated. However, $1,392,005.84 is very close to the materiality threshold of $1,392,005.96. **Note** This example demonstrates the difference between using **Upper error limit** and **Maximum Tolerable Taintings (%)** to evaluate misstatement. If you use the more stringent **Maximum Tolerable Taintings (%)** method, the account is materially misstated. The sum of the tainting percentages in the "Misstatements" column to the left is 221.61% (100% + 56.25% + 65.36%), which is slightly greater than the **Maximum Tolerable Taintings (%)** of 219.48% reported when you calculated sample size.
6,002.16, 6,002.16 31,997.46, 18,000.00 13,225.50,13,225.50	1,505,511.86	The account **is** materially misstated. $1,505,511.86 **>** the materiality threshold of $1,392,005.96

Calculating sample size for a monetary unit sample

Before sampling a set of data, you must calculate the statistically appropriate sample size, and other values required by the subsequent sample and evaluate operations.

The **Calculate Sample Size** feature in Analytics calculates the required values for you based on input values you provide.

The importance of calculating a sample size

Calculating an appropriate sample size is critical to the validity of the subsequent sample. If the sample is not valid, or representative, you cannot reliably project the results of audit procedures you perform on the sample to the entire population.

Do not skip calculating a sample size, or guess at a sample size.

Most of the input values you use to calculate sample size are based on your professional judgment. Ensure that you fully understand the implications of the values before relying on the results of sampling in a production environment. Consult audit sampling resources, or an audit sampling specialist, if you are in doubt.

How input values affect sample size

Input values affect the sample size calculated by Analytics. You can use the **Calculate** button in the **Size** dialog box to experiment with how the different input values affect the sample size.

The table below summarizes the effect of input values on sample size.

> **Caution**
> In a production environment, do not manipulate input values solely to achieve a smaller sample size. Input values should be based on your professional judgment about what is most appropriate for the data being sampled and the audit objective.

Increasing this input value:	Decreases sample size	Increases sample size
Confidence		✔
Population		✔

Increasing this input value:	Decreases sample size	Increases sample size
Materiality	✔	
Expected Total Errors		✔

Steps

> **Note**
> Do not include the thousands separator, or the percentage sign, when you specify values. These characters prevent the command from running, or cause errors.

1. Select **Sampling > Record/Monetary Unit Sampling > Calculate Size**.

   > **Note**
   > The menu option is disabled if a table is not open.

2. On the **Main** tab, select **Monetary**.
3. Enter the input values to use for calculating the sample size:
   - **Confidence**
   - **Population**
   - **Materiality**
   - **Expected Total Errors**

   > **Note**
   > The input values are explained in detail below.

4. (Optional) Click **Calculate** to see a preview of the output results.

   > **Tip**
   > Clicking **Calculate** instead of **OK** allows you to experiment with different input values before outputting the results.

   > **Note**
   > The output results are explained in detail below.

5. On the **Output** tab:
   a. In the **To** panel, select one of the following:
      - **Screen** - displays the results in the Analytics display area

        > **Tip**
        > You can click any linked result value in the display area to drill down to the associated record or records in the source table.

      - **File** - saves or appends the results to a text file

        The file is saved outside Analytics.

   b. If you selected **File** as the output type, do one of the following:
      - Enter a file name in the **Name** text box.
      - Click **Name** and enter the file name, or select an existing file in the **Save** or **Save File As** dialog box to overwrite or append to the file.

        If Analytics prefills a file name, you can accept the prefilled name, or change it.

        You can also specify an absolute or relative file path, or navigate to a different folder, to save or append the file in a location other than the project location. For example: `C:\Results\Output.txt` or `Results\Output.txt`.

        > **Note**
        > **ASCII Text File** or **Unicode Text file** (depending on which edition of Analytics you are using) is the only option for **File Type**.

6. Click **OK**.
7. If the overwrite prompt appears, select the appropriate option.

Size dialog box inputs and results

The tables below provide detailed information about the input values and output results in the **Size** dialog box.

Main tab – input values

Input values – Size dialog box	Description
Confidence	Your desired confidence level that the resulting sample is representative of the entire population. For example, entering 95 means that you want to be confident that 95% of the time the sample will in fact be representative. Confidence is the complement of "sampling risk". A 95% confidence level is the same as a 5% sampling risk.
Population	The absolute value of the numeric sample field.

Input values - Size dialog box	Description
	Note To get the absolute value, profile or generate statistics on the sample field.
Materiality	The maximum total amount of misstatement that can occur in the sample field without being considered a material misstatement. For example, entering 29000 means that the total amount of misstatement must be greater than $29,000 to be considered a material misstatement.
Expected Total Errors	The total amount of misstatement that you expect the sample field to contain. For example, entering 5800 means that you expect the total amount of misstatement to be $5,800. **Note** The **Expected Total Errors** you specify must be less than **Materiality**. If the difference between them is too small, the error message **Error rate too high for calculation** appears. In audit sampling terms, the degree of sampling precision represented by the difference is too small to be calculated for the confidence level you specified.

Main tab - output results

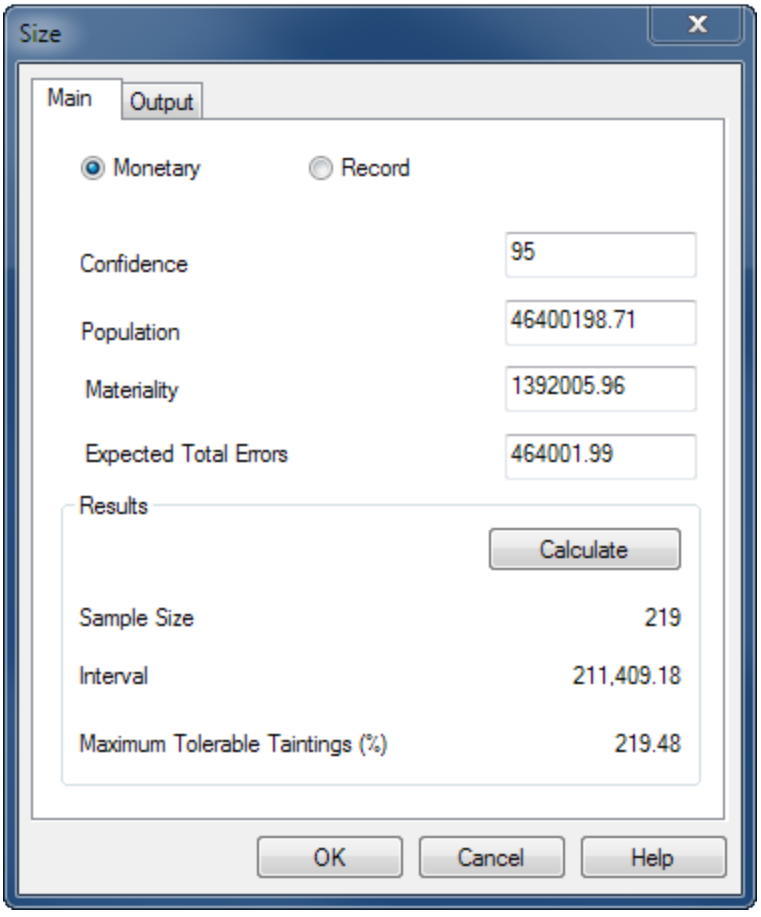
Output results - Size dialog box	Description
Sample Size	The required sample size.
Interval	The interval value - required for the fixed interval and the cell selection methods.
Maximum Tolerable Taintings (%)	The maximum accumulated tainting percentages that can occur in misstated amounts in the resulting sample without exceeding the **Materiality**. **Note** The **Maximum Tolerable Taintings (%)** value reported by Analytics can be greater than 100%. For more information, see "Maximum Tolerable Taintings (%)" on page 1010.

An example of inputs and results

Calculating the size of a monetary unit sample for the Invoices table

The figure below provides an example of input values and output results when calculating sample size for monetary unit sampling.

- the absolute value of the transaction amount field is $46,400,198.71
- **Materiality** is set at 3% of the absolute value
- **Expected Total Errors** is set at 1% of the absolute value
- the required sample size is 219 records



The calculation is based on the `Invoices` table in `ACL_Rockwood.acl` (`ACL DATA\Sample Data Files\ACL_Rockwood\ACL_Rockwood.acl`).

Maximum Tolerable Taintings (%)

> **Note**
> If you intend to use the evaluation feature in Analytics, you do not need to use the value reported by **Maximum Tolerable Taintings (%)**. Instead, you use the **Upper Error Limit** calculated by the evaluation feature. For more information, see "Evaluating errors in a monetary unit sample" on page 1022.

Maximum Tolerable Taintings (%) provides one way of evaluating misstatement in a population.

If you use this method, you know in advance the threshold value reported by Analytics, before you begin audit procedures on the sampled data. If cumulative errors you observe in the course of performing the procedures exceed the threshold value, you know at that point that the sample field is materially misstated.

Example

In an accounts receivable table you discover that a book value of $1000 should actually be $930. In a misstated amount, tainting is the percentage of the book value that the misstatement represents.

Book value	Audit value	Overstatement	Tainting
$1,000	$930	$70	7% (70/1000)

After performing your substantive procedures on sampled data you can sum all the individual tainting percentages from any misstated amounts. If the sum of the tainting percentages is less than or equal to the **Maximum Tolerable Taintings (%)** reported by Analytics, you can consider that the amounts in the sample field as a whole are not materially misstated, for your specified confidence level.

Example

You discover three misstated amounts in an accounts receivable table, which results in the following taintings, and total tainting percentage:

Book value	Audit value	Overstatement	Tainting
$1,000	$930	$70	7% (70/1000)
$2,500	$1,500	$1000	40% (1000/2500)

Book value	Audit value	Overstatement	Tainting
$2,750	$2,695	$55	2% (55/2750)
			49% (total tainting percentage)

Let's assume the **Maximum Tolerable Taintings (%)** reported by Analytics when you calculated the sample size for the table was **92.30%**. Because the total tainting percentage of 49% is less than 92.30%, you can conclude that the amounts in the sample field as a whole are not materially misstated, for your specified confidence level.

> **Note**
>
> Evaluation using **Maximum Tolerable Taintings (%)** is slightly more stringent than the evaluation feature in Analytics.
>
> If the sum of the tainting percentages marginally exceeds the **Maximum Tolerable Taintings (%)** value you should use the evaluation feature to confirm that the sample field is in fact materially misstated.
>
> For more information, see "Evaluating errors in a monetary unit sample" on page 1022.

Statistical validity of sample sizes generated by Analytics

Analytics generates statistically valid sample sizes for most analyses. Exceptions may apply in the following situations:

- You are sampling data sets of less than 1000 records.
- Your organization has in-house sampling experts who can define sample sizes precisely tailored to your needs.
- Your organization has mandated the use of another sampling tool or methodology.

Poisson distribution versus binomial distribution

Two commonly used methods of generating sample sizes are the Poisson and the binomial distributions. Analytics generates sample sizes using the Poisson distribution.

For typical data sets of a thousand or more records, the Poisson and the binomial distributions generate nearly identical sample sizes. For populations of under a thousand records, sample sizes determined with the Poisson distribution tend to be slightly larger and therefore more conservative

than sizes determined with the binomial distribution. The binomial distribution adjusts the sample size downward for small populations but the Poisson distribution does not. With very small populations, the sample size generated by the Poisson distribution can actually exceed the population size.

When calculating sample sizes in Analytics, recognize that for record sampling of small data sets, the sample size may be larger than you need. This larger sample size does not present an obstacle to analysis because it is common practice to manually oversample small populations.

Performing monetary unit sampling

You can create a new table that contains a representative sample of the monetary data in the active table.

Monetary unit sampling is appropriate if you are interested in the total amount of monetary misstatement in a file.

> **Note**
> This procedure does not include filtering (IF statements) or scope parameters because applying these options compromises the validity of a sample.

Steps

> **Note**
> Do not include the thousands separator, or the percentage sign, when you specify values. These characters prevent the command from running, or cause errors.

1. In the Navigator, open the table you want to draw a sample from.
2. Optional. If you intend to use the **Random** selection method, profile or generate statistics on the sample field.
3. Select **Sampling > Record/Monetary Unit Sampling > Sample**.
4. On the **Main** tab, select **MUS**.
5. Select the field to sample from the **Sample On** drop-down list.
6. In the **Sample Parameters** panel, specify a sample selection method:
   - **Fixed Interval**
   - **Cell**
   - **Random**

   > **Note**
   > Do not use the random selection method with monetary unit sampling if you intend to use Analytics to evaluate any misstatement detected in the resulting sample.
   >
   > Evaluating monetary unit samples requires that you use the fixed interval or cell selection methods.

7. Enter the sample parameters for the selection method you chose:

Selection method	Sample parameters
Fixed Interval	◦ **Interval**

Selection method	Sample parameters
	◦ **Start** (optional) ◦ **Cutoff** (optional)
Cell	◦ **Interval** ◦ **Seed** (optional) ◦ **Cutoff** (optional) ◦ **Algorithm** - leave **Mersenne Twister** selected
Random	◦ **Size** ◦ **Seed** (optional) ◦ **Population** ◦ **Algorithm** - leave **Mersenne Twister** selected

> **Note**
> Sample parameters are explained in detail below.

8. In the **To** text box, specify the name of the Analytics table that will contain the output results.
9. On the **More** tab, select one of the following:
   - **Record** -The entire record is included in the output table.
   - **Fields** - Only the selected fields are included in the output table.
10. If you chose **Fields**, select the field(s) to include in the output table from the **Extract Fields** list.
11. Optional. Select one or more of the following options:
    - **Subsample**
    - **Report Selection Order**
    - **No Repeats**

> **Note**
> The options are explained below.
> **Subsample** is available only if **Fields** output is selected.
> **Report Selection Order** is available only if both the **Random** selection method and **Fields** output are selected.

12. Click **OK**.

Sample dialog box options

The tables below provide detailed information about the options in the **Sample** dialog box.

Main tab

Options - Sample dialog box	Description
MUS Record	The sample type: ◦ **MUS** - Monetary unit sampling Appropriate if you are interested in the total amount of monetary misstatement in a file. ◦ **Record** - Record sampling Appropriate if you are interested in the rate of deviation from a prescribed control.
Sample On	The numeric sample field. ◦ You can select the field from the **Sample On** drop-down list. ◦ You can click **Sample On** to select the field, or to create an expression.
Fixed Interval	Specifies that the fixed interval method is used for sample selection. Samples are selected based on an interval value and a start number that you specify. For more information, see "Fixed interval selection method" on page 953. If you selected **Fixed Interval** enter the following values: ◦ **Interval** (required) - the interval value that was generated by calculating the sample size **Note** If you have not already calculated the sample size, you can click **Size** to open the **Size** dialog box. For more information, see "Calculating sample size for a monetary unit sample" on page 1005. ◦ **Start** (optional) - a start number that is greater than zero and less than the interval value **Tip** Enter a start number of '0', or leave the start number blank, if you want Analytics to randomly select a start number. ◦ **Cutoff** (optional) - a top stratum cutoff value Sample field amounts greater than or equal to the cutoff value are automatically selected and included in the sample. If you leave the cutoff value blank, a default cutoff value equal to the interval value is used. For more information, see "Top stratum cutoff" on page 1019.
Cell	Specifies that the cell method is used for sample selection. The data set is divided into multiple equal-sized cells or groups, and one sample is randomly selected from each cell. The interval value dictates the size of each cell. For more information, see "Cell selection method" on page 954. If you selected **Cell** enter the following values: ◦ **Interval** (required) - the interval value that was generated by calculating the sample

Options – Sample dialog box	Description
	size **Note** If you have not already calculated the sample size, you can click **Size** to open the **Size** dialog box. For more information, see "Calculating sample size for a monetary unit sample" on page 1005. ◦ **Seed** (optional) – can be any number This number is used to initialize the random number generator in Analytics. **Tip** Enter a seed value of '0', or leave the seed blank, if you want Analytics to randomly select a seed value. ◦ **Cutoff** (optional) – a top stratum cutoff value Sample field amounts greater than or equal to the cutoff value are automatically selected and included in the sample. If you leave the cutoff value blank, a default cutoff value equal to the interval value is used. For more information, see "Top stratum cutoff" on page 1019. ◦ **Algorithm** (required) – leave **Mersenne Twister** selected Only select **Default** if you require backward compatibility with Analytics scripts or sampling results created prior to Analytics version 12.
Random	Specifies that the random method is used for sample selection. Samples are randomly selected from the entire data set. For more information, see "Random selection method" on page 956. If you selected **Random** enter the following values: ◦ **Size** (required) – the sample size that was calculated by Analytics **Note** If you have not already calculated the sample size, you can click **Size** to open the **Size** dialog box. For more information, see "Calculating sample size for a monetary unit sample" on page 1005. ◦ **Seed** (optional) – can be any number This number is used to initialize the random number generator in Analytics. **Tip** Enter a seed value of '0', or leave the seed blank, if you want Analytics to randomly select a seed value. ◦ **Population** (required) – the absolute value of the sample field, which is the population from which the sample will be selected

Options - Sample dialog box	Description
	Tip This field is prefilled with the correct value if you previously profiled or generated statistics on the sample field. ◦ **Algorithm** (required) - leave **Mersenne Twister** selected Only select **Default** if you require backward compatibility with Analytics scripts or sampling results created prior to Analytics version 12.
If	**Caution** Do not create an IF statement or filter records in the course of sampling. Doing so compromises the validity of the sample. For more information, see "Conditional sampling" on page 1086.
To	The name and location of the output table. ◦ **To save the output table to the Analytics project folder** - enter only the table name. ◦ **To save the output table in a location other than the project folder** - specify an absolute or relative file path, or click **To** and navigate to a different folder. For example: `C:\Results\Output.fil` or `Results\Output.fil`. Regardless of where you save the output table, it is added to the open project if it is not already in the project. If Analytics prefills a table name, you can accept the prefilled name, or change it.
Local	If you are connected to a server table, specifies where to save the output table. ◦ **Local selected** - saves the output table to the same location as the Analytics project, or to a specified path, or location you navigate to. ◦ **Local deselected** - saves the output table to the Prefix folder on AX Server.
Use output table	Specifies whether the Analytics table containing the output results opens automatically upon completion of the operation.

More tab

Options - Sample dialog box	Description
Scope panel	**Caution** Do not limit which records are processed in the course of sampling. Doing so compromises the validity of the sample. For more information, see "Conditional sampling" on page 1086.
Record Fields	Specifies whether the output table includes the entire record, or selected fields. If you choose **Fields**, do one of the following: ◦ Select the field(s) to extract from the **Extract Fields** list.

Options – Sample dialog box	Description
	◦ Click **Extract Fields** to select the field(s), or to create an expression. The order in which you select the fields is the order in which the columns appear in the results. If you are appending results to an existing Analytics table, the column selection and order must be identical to the column selection and order in the existing table.
Subsample Report Selection Order No Repeats	(Optional) Additional sampling options. You can select one or more of the following options: ◦ **Subsample** – Adds the SUBSAMPLE field to the output results. You can use this field to randomly select individual transactions from total amounts. For more information, see "Subsampling" on page 1020. ◦ **Report Selection Order** – Adds the ORDER field to the output results. This field displays the order in which each record is randomly selected. ◦ **No Repeats** – Prevents records being selected more than once. **Note** **Subsample** is available only if **Fields** output is selected. **Report Selection Order** is available only if both the **Random** selection method and **Fields** output are selected. With **No Repeats**, selected records become ineligible for subsequent selection, which can reduce the size of the sample. You should consider oversampling the data set to compensate. For more information, see "Sample selection without repeats" on page 1020.
Append To Existing File	Specifies that the output results are appended (added) to the end of an existing Analytics table. **Note** Leaving **Append To Existing File** deselected is recommended if you are uncertain whether the output results and the existing table have an identical data structure. For more information about appending and data structure, see "Appending output results to an existing table" on page 200.
OK	Executes the operation. If the overwrite prompt appears, select the appropriate option. If you are expecting the **Append** option to appear and it does not, click **No** to cancel the operation and see "Appending output results to an existing table" on page 200.

Monetary unit sampling options

For monetary unit sampling, you can specify one or more of the following options:

- Top stratum cutoff
- Subsampling
- Sample selection without repeats

Top stratum cutoff

> **Note**
> Top stratum cutoff is available only for monetary unit sampling that uses the fixed interval or the cell selection methods.

Top stratum cutoff is an additional method that Analytics uses to bias monetary unit sampling toward larger amounts. By default, sample field amounts greater than or equal to the interval value are considered top stratum amounts and are automatically included in the sample.

Negative as well as positive amounts are eligible for automatic inclusion because it is the absolute value of the amount that is considered.

Note that the greater the amount of automatic selection, the larger the sample size becomes.

You can optionally specify a top stratum cutoff value that is higher or lower than the interval value:

Top stratum cutoff higher than the interval value	Decreases the probability that larger amounts will be automatically included in the sample. If you specify a cutoff value larger than the largest positive or negative amount in the sample field, no amounts are automatically selected.
Top stratum cutoff lower than the interval value	Increases the probability that larger amounts will be automatically included in the sample. If no amounts are automatically selected using the default top stratum cutoff, you can adjust the cutoff value downward in order to automatically select some of the larger amounts in the sample field. **Caution** If you specify a cutoff value that is too small in relation to the sample field amounts, too many amounts are automatically selected, which defeats the purpose of sampling.

Top stratum selections and amount recorded in the log

When you perform a monetary unit sample, the number of top stratum selections, and the total top stratum amount, is displayed in the log.

Example

The log displays that 8 of 93 selected records are top stratum, accounting for $33,153.55 of the absolute value of the numeric sample field.

```
Sample size = 93 (8 top stratum), out of 772 records sampled
Population: 585674.41, Top stratum: 33153.55, Other: 552520.86
```

Subsampling

> **Note**
> Subsampling is available only for monetary unit sampling using field output.

In some cases, each amount in a sample field represents a total of several separate transactions. If you want to perform audit procedures on only one transaction from each sampled total amount, you can use subsampling to randomly select the individual transactions.

When you select **Subsample** in the **Sample** dialog box, the resulting sample includes the SUBSAMPLE field. This field contains amounts that represent the difference between the total amount and the actual monetary unit used to select the total amount.

Example

$12,455	(total amount)
- $4,620	(selected monetary unit)
= $7,835	(amount displayed in the SUBSAMPLE field)

To complete the process, you would select the transaction containing the 7,835th dollar in the cumulative balance of transactions for that particular total amount.

> **Note**
> Any top stratum cutoff amounts in the sample display “0.00” in the SUBSAMPLE field because they are automatically included in the sample and no monetary unit was involved in their selection.

Sample selection without repeats

Monetary unit sampling may select the same record more than once. Each amount in the sample field contains multiple monetary units, and two or more monetary units belonging to the same amount can be selected, which means the record containing the amount is multiply selected.

You can prevent multiple selection of the same record by selecting **No Repeats** in the **Sample** dialog box. The resulting sample will not contain duplicates. However, the number of sampled records may be smaller than the sample size calculated by Analytics. To compensate, you can oversample by using one of the following methods to increase the sample size:

- **Fixed interval or cell selection methods:**
  - decrease the size of the interval
  - adjust the top stratum cutoff value to automatically select a greater number of records
- **Random selection method** – increase the specified sample size

Evaluating errors in a monetary unit sample

After you have performed your audit procedures on the set of sampled data you can use Analytics to:

- project any misstatements you found to the entire account
- calculate an upper limit on misstatement amount

Even if you found no errors, you still use the evaluation feature to calculate the basic allowance for sampling risk.

> **Note**
> Evaluating errors requires input of some of the values previously generated by calculating sample size.
>
> To use the evaluation feature with the results of a monetary unit sample, you must have drawn the sample using either the fixed interval or the cell selection methods.

How evaluation and comparison work

When you evaluate, Analytics uses a statistical formula to project the misstatements you found in the sample to the entire account, and calculates the **Upper Error Limit** (upper misstatement limit).

You compare the calculated value to the **Materiality** that you decided upon earlier when you calculated sample size. Based on the comparison, you decide if monetary data is fairly stated.

Comparison	Conclusion
Upper Error Limit is less than or equal to **Materiality**	The amounts in the sample field as a whole are fairly stated
Upper Error Limit is greater than **Materiality**	The amounts in the sample field as a whole are materially misstated

Steps

> **Note**
> Do not include the thousands separator, or the percentage sign, when you specify values. These characters prevent the command from running, or cause errors.

1. Select **Sampling > Record/Monetary Unit Sampling > Evaluate**

   > **Note**
   > The menu option is disabled if a table is not open.

2. On the **Main** tab, select **Monetary**.
3. Enter the input values to use for evaluating misstatements:
   - **Confidence**
   - **Interval**
   - **Errors**

   > **Note**
   > The input values are explained in detail below.

4. On the **Output** tab:
   a. In the **To** panel, select one of the following:
      - **Screen** - displays the results in the Analytics display area

        > **Tip**
        > You can click any linked result value in the display area to drill down to the associated record or records in the source table.

      - **File** - saves or appends the results to a text file

        The file is saved outside Analytics.

   b. If you selected **File** as the output type, do one of the following:
      - Enter a file name in the **Name** text box.
      - Click **Name** and enter the file name, or select an existing file in the **Save** or **Save File As** dialog box to overwrite or append to the file.

        If Analytics prefills a file name, you can accept the prefilled name, or change it.

        You can also specify an absolute or relative file path, or navigate to a different folder, to save or append the file in a location other than the project location. For example: `C:\Results\Output.txt` or `Results\Output.txt`.

        > **Note**
        > **ASCII Text File** or **Unicode Text file** (depending on which edition of Analytics you are using) is the only option for **File Type**.

5. Click **OK**.
6. If the overwrite prompt appears, select the appropriate option.

Evaluate dialog box inputs

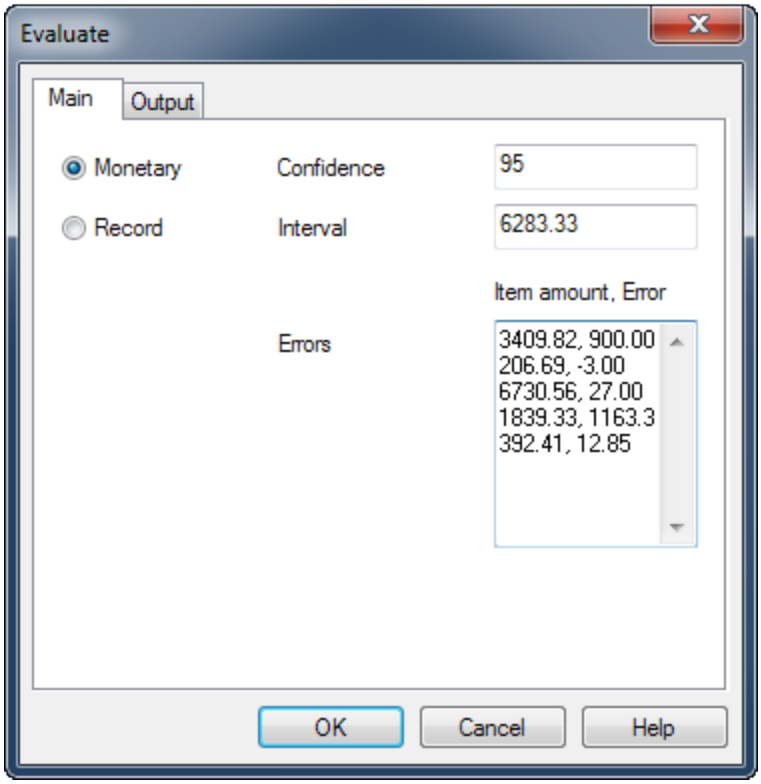
The table below provides detailed information about the input values in the **Evaluate** dialog box.

Main tab – input values

Input values – Evaluate dialog box	Description
Confidence	The same confidence level that you entered when you calculated the sample size. For more information, see "Calculating sample size for a monetary unit sample" on page 1005.
Interval	The interval value that you used when you drew the sample. **Note** The interval value that you used might differ from the interval value initially calculated by Analytics.

Errors
(Item amount, Error)
	A list of all misstatement errors that you found in the sample.
	Enter the book value of the amount and the misstatement amount, separated by a comma. Enter overstatements as positive amounts, and understatements as negative amounts.
	Tip
	If the list of misstatement errors is long, it may be easier to copy and paste the list from another application.
	Example
	If an amount has a book value of $1,000 and an audit value of $930, enter `1000,70`.
	If an amount has a book value of $1,250 and an audit value of $1,450, enter `1250,-200`.
	Enter each error on a separate line:
	`1000,70`
	`1250,-200`

The figure below shows an example of input values for evaluating errors in a monetary unit sample.



Results

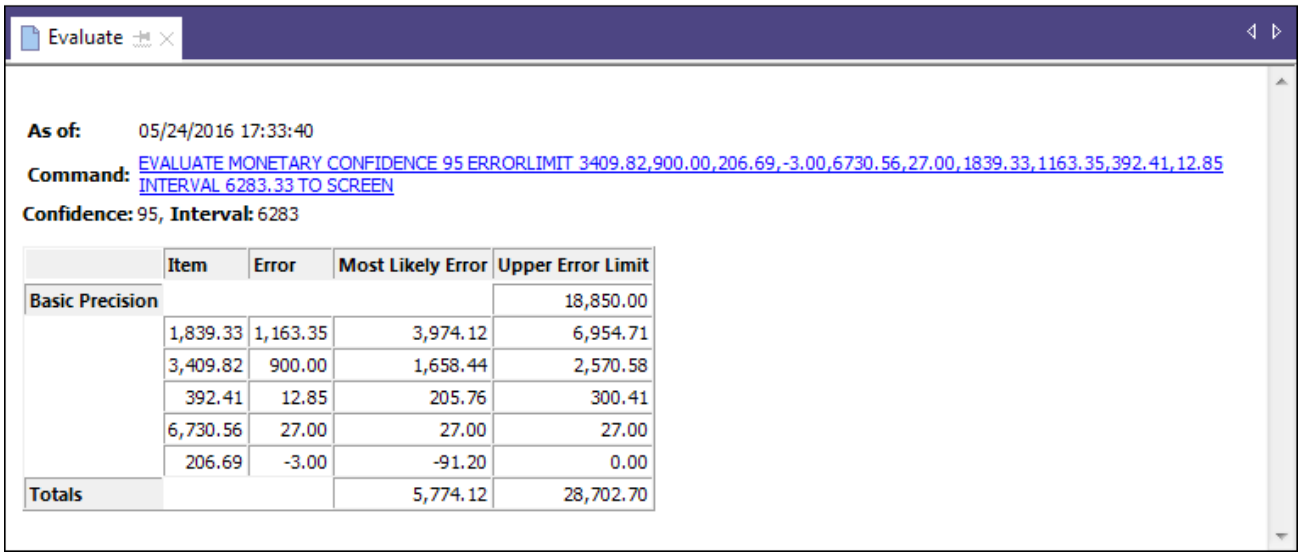
Evaluating the errors you found in a monetary unit sample produces the following results:

Result value	Description
Item	The list that you entered of sample amounts with misstatement error.
Error	The list that you entered of misstatement amounts.
Basic Precision	The basic allowance for sampling risk (18,850.00 in the figure below).
Most Likely Error (projected misstatement)	The misstatement amount for each error projected to the interval containing the sample amount. **Most Likely Error** amounts that are not top stratum are listed in descending order. Top stratum misstatement amounts are listed between projected overstatements and projected understatements. The projection calculation is not performed on top stratum misstatement amounts.
Upper Error Limit (upper misstatement limit)	The **Most Likely Error** amounts adjusted for sampling risk. The adjustment calculation is not performed on top stratum misstatement amounts, or on projected understatements. Projected understatements are listed as “0.00” so they do not affect the **Upper Error Limit**.

Result value	Description
Totals	The **Most Likely Error** amount, and the **Upper Error Limit** amount, for the entire population or account balance.

The figure below shows the results of evaluating errors found in a monetary unit sample.

In the Item column	Projection calculation	Adjustment calculation
First three amounts are not top stratum ◦ 1,839.33 ◦ 3,409.82 ◦ 392.41	✔	✔
Fourth amount is top stratum ◦ 6,730.56	✘	✘
Fifth amount is an understatement ◦ 206.69	✔	✘



	Item	Error	Most Likely Error	Upper Error Limit
Basic Precision				18,850.00
	1,839.33	1,163.35	3,974.12	6,954.71
	3,409.82	900.00	1,658.44	2,570.58
	392.41	12.85	205.76	300.41
	6,730.56	27.00	27.00	27.00
	206.69	-3.00	-91.20	0.00
Totals			5,774.12	28,702.70

What the “Upper Error Limit” tells you

The total amount of **Upper Error Limit**, when compared to the **Materiality** that you decided upon when you calculated the sample size, tells you:

- Whether the amount in the account balance you are examining is fairly stated
- If the amount is fairly stated, what the maximum amount of misstatement is likely to be

Example

You evaluate the errors you found in a monetary unit sample and Analytics returns an **Upper Error Limit** of $28,702.70. This amount is less than the **Materiality** (tolerable misstatement) of $29,000 that you specified earlier when you calculated the sample size, and specified a confidence level of 95%.

Based on this information, you can make the following statement:

There is a 95% probability that the actual misstatement in the account balance does not exceed $28,702.70.

If the **Upper Error Limit** is greater than $29,000, the account balance is probably materially misstated. You need to decide upon further appropriate steps to meet your audit objective.

How the Upper Error Limit is calculated for monetary unit sampling

The **Upper Error Limit** calculated by Analytics is a compound figure that adjusts for sampling risk – that is, the risk that misstatements in the sampled amounts underrepresent the true total amount of misstatement in the account balance you are examining.

Show me more

Upper Error Limit is the sum of the following amounts:

Amount	Explanation
Basic precision	In the absence of any misstatement errors in the sampled amounts, a basic allowance for sampling risk calculated by Analytics using a statistical formula. A basic allowance for sampling risk is required because even if you found no errors in the sampled amounts, you cannot be sure no errors exist in the population as a whole.
An adjusted amount for each misstatement error	The result of the following calculation: `tainting percentage (misstatement amount/book value of sample amount) * interval amount * incremental allowance for sampling risk` ◦ `Tainting percentage * interval amount` projects the observed amount of misstatement for one sample amount to the interval containing the sample amount. Analytics calls this projected misstatement **Most Likely Error**. The sum of all projected misstatements is the **Most Likely Error** for the account balance. ◦ `Most Likely Error * an incremental allowance for sampling risk` is calculated by Analytics using a statistical formula. This additional adjustment is required because the **Most Likely Error** may still underestimate the true amount of misstatement in the account balance.

Classical variables sampling

Classical variables sampling is a statistical sampling method for estimating:

- the total audited value of an account or class of transactions
- the total amount of monetary misstatement in an account or class of transactions

Classical variables sampling works best with financial data that has the following characteristics:

a moderate to larger number of misstatements For example, 5% or more of the items are misstated.
either overstatements or understatements may exist
zero dollar items may exist

> **Tip**
> For a hands-on introduction to the end-to-end process of classical variables sampling in Analytics, see "Classical variables sampling tutorial" on page 1037.

> **Note**
> In addition to financial data, you can use classical variables sampling with any numeric data that has a variable characteristic – for example, quantity, units of time, or other units of measurement.

How it works

Classical variables sampling allows you to select and analyze a small subset of the records in an account. Based on the results of analyzing the subset, you can estimate the total audited value of the account, and the total amount of monetary misstatement.

The two estimates are computed as ranges:

- The **point estimate** is the midpoint of a range.
- The **upper limit** and the **lower limit** are the two end points of a range.

  You can also choose to compute a one-sided estimate or range, with a point estimate and only an upper limit, or only a lower limit.

You compare the estimated range to the book value of the account, or to the misstatement amount that you judge is material, and make a determination regarding the account.

Classical variables sampling supports making this sort of statement:

- *There is a 95% probability that the true audited value of the account is between 45,577,123.95 and 46,929,384.17, a range that contains the account book value of 46,400,198.71. Therefore the amounts in the account are fairly stated.*

- *There is a 95% probability that the misstatement in the account balance is between -813,074.76 and 539,185.46, which does not exceed the monetary precision of ±928,003.97. Therefore the amounts in the account are fairly stated.*

Overview of the classical variables sampling process

> **Caution**
>
> Do not skip calculating a valid sample size.
>
> If you go straight to drawing a sample of records, and guess at a sample size, there is a high likelihood that the projection of your analysis results will be invalid, and your final conclusion flawed.

The classical variables sampling process involves the following stages:

1. Prepare (plan) the classical variables sample
2. Draw the sample of records
3. Perform your intended audit procedures on the sampled data.
4. Evaluate the following:
   - whether the audited value of the sampled data, when projected to the account as a whole, falls within an acceptable range of the recorded book value
   - whether the observed levels of monetary misstatement in the sampled data represent an acceptable or unacceptable amount of misstatement in the account as a whole

Values are retained and prefilled between stages

Classical variables sampling in Analytics requires that you enter information in three separate dialog boxes, and run the associated commands, in this order:

1. **CVS Prepare** dialog box
2. **CVS Sample** dialog box
3. **CVS Evaluate** dialog box

As you move through this process, information from one dialog box is automatically prefilled into the next dialog box. Prefilling saves considerable labor, and removes the risk of accidentally entering incorrect values and invalidating the sample.

However, values that automatically prefill the **CVS Sample** and the **CVS Evaluate** dialog boxes are only stored temporarily, and are deleted when you close the Analytics project.

Regenerating classical variables sampling values

In a production environment, you typically perform the different stages of the classical variables sampling process at different times. You can use any of the following methods for regenerating the classical variables sampling values that are lost when you close Analytics.

The first method is the easiest.

- **Save the prefilled commands**

  The results of the CVS Prepare and the CVS Sample stages include subsequent commands in the classical variables sampling process that are prefilled with the required values. Save these prefilled commands in separate scripts to use later.

  For more information, see "Classical variables sampling tutorial" on page 1037.

- **Save the executed commands in scripts**

  After performing the CVS Prepare and CVS Sample stages, copy the `CVSPREPARE` and `CVSSAMPLE` commands from the Analytics display area and save them in separate scripts. You can run these scripts later to regenerate the classical variables sampling values.

  The drawback of this method is that you draw a redundant sample of records.

- **Retrieve the executed commands from the log**

  Copy the `CVSPREPARE` and `CVSSAMPLE` commands from the log, and re-run them in the command line to regenerate the classical variables sampling values.

  The drawback of this method is that locating the correct instances of the commands in the log can be difficult, and you draw a redundant sample of records.

Changing prefilled values

Normally you should not change any of the prefilled classical variables sampling values. Changing prefilled values can negate the statistical validity of the sampling process.

> **Caution**
> Update prefilled values only if you have the statistical knowledge to understand the effect of the change.

Numeric length limitation

Several internal calculations occur during the preparation stage of classical variables sampling. These calculations support numbers with a maximum length of 17 digits. If the result of any calculation exceeds 17 digits, the result is not included in the output, and you cannot continue with the sampling process.

Note that source data numbers of less than 17 digits can produce internal calculation results that exceed 17 digits.

Stratification

Classical variables sampling gives you the option of numerically stratifying the records in a population before drawing a sample.

The benefit of stratification is that it often dramatically reduces the required sample size while still maintaining statistical validity. A reduced sample size means less data analysis work is required to reach your goal.

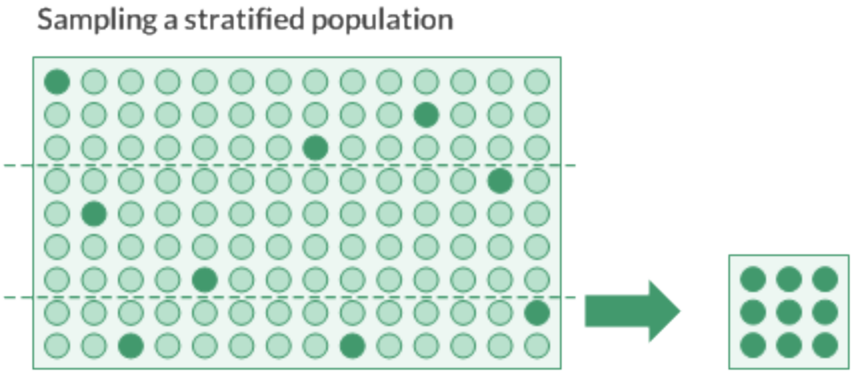
How it works

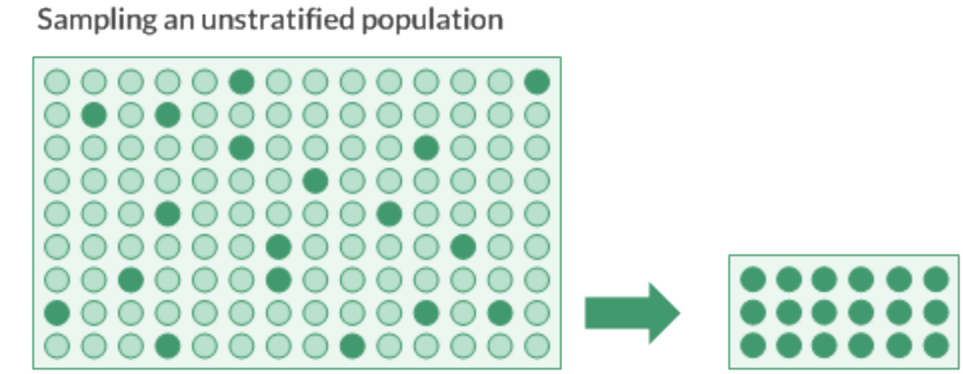
Show me more

Stratification works by dividing a population into a number of subgroups, or levels, called **strata**. Ideally, the values in each stratum are relatively homogenous.

A statistical algorithm (the Neyman method) sets the boundaries between the strata. The algorithm positions the boundaries to minimize the dispersion of values within each stratum, which decreases the effect of population variance. Reducing the variance, or 'spread', reduces the required sample size. By design, the range of each stratum is not uniform.

The required number of samples is then calculated on a per-stratum basis, and totaled, rather than on the basis of the entire, unstratified population. For the same set of data, the stratified approach typically results in a much smaller sample size than the unstratified approach.



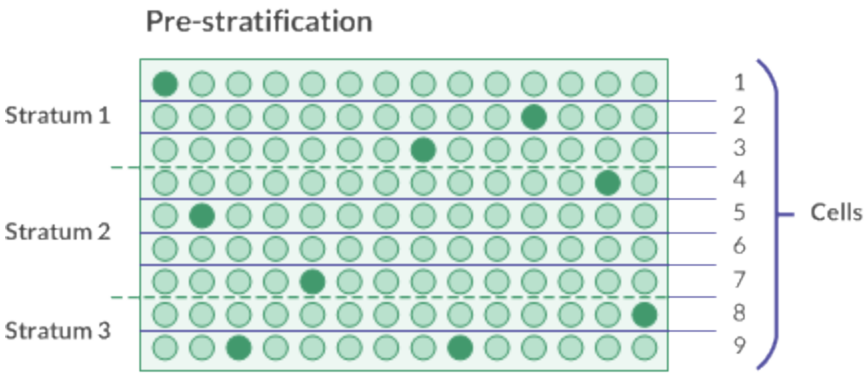


Pre-stratification using cells

As part of the stratification process, you specify the number of cells to use to pre-stratify the population. Cells are uniform numeric divisions, and narrower than strata.

A statistical algorithm uses the count of the records in each cell as part of the calculation that assigns optimal strata boundaries. Cells are not retained in the final stratified output.

At a minimum, the number of specified cells must be twice the number of specified strata.



> **Note**
> Pre-stratification cells and the cells used in the cell method of sample selection are not the same thing.

Too much of a good thing

Stratification is a powerful tool for managing sample size, but you should exercise care when specifying the number of strata and the number of cells.

As a starting point, try:

- 4 to 5 strata
- 50 cells

After a certain point, increasing the number of strata, or the number of cells, has little or no effect on sample size. However, these increases can adversely affect the design of the sample, or the performance of Analytics when stratifying large data sets.

Regarding sample design, when you reach the evaluation stage you need to have a minimum number of misstatements in each stratum in order to reliably project misstatements to the entire population. If you have too many strata in relation to the number of misstatements, problems can occur with the projection.

The certainty strata

Defining a **certainty stratum** is another available stratification option. You can define a top certainty stratum, a bottom certainty stratum, or both.

Using a certainty stratum has two benefits:

- **Automatic inclusion** – Individually significant items, or high value items, are automatically included in the sample, and not at risk of being excluded by the random selection method.
- **Reduction of variance** – Certainty stratum items are removed from the sample size calculation. Because of their nature, high value items can significantly increase population variance, and the required sample size, if they are included in the calculation.

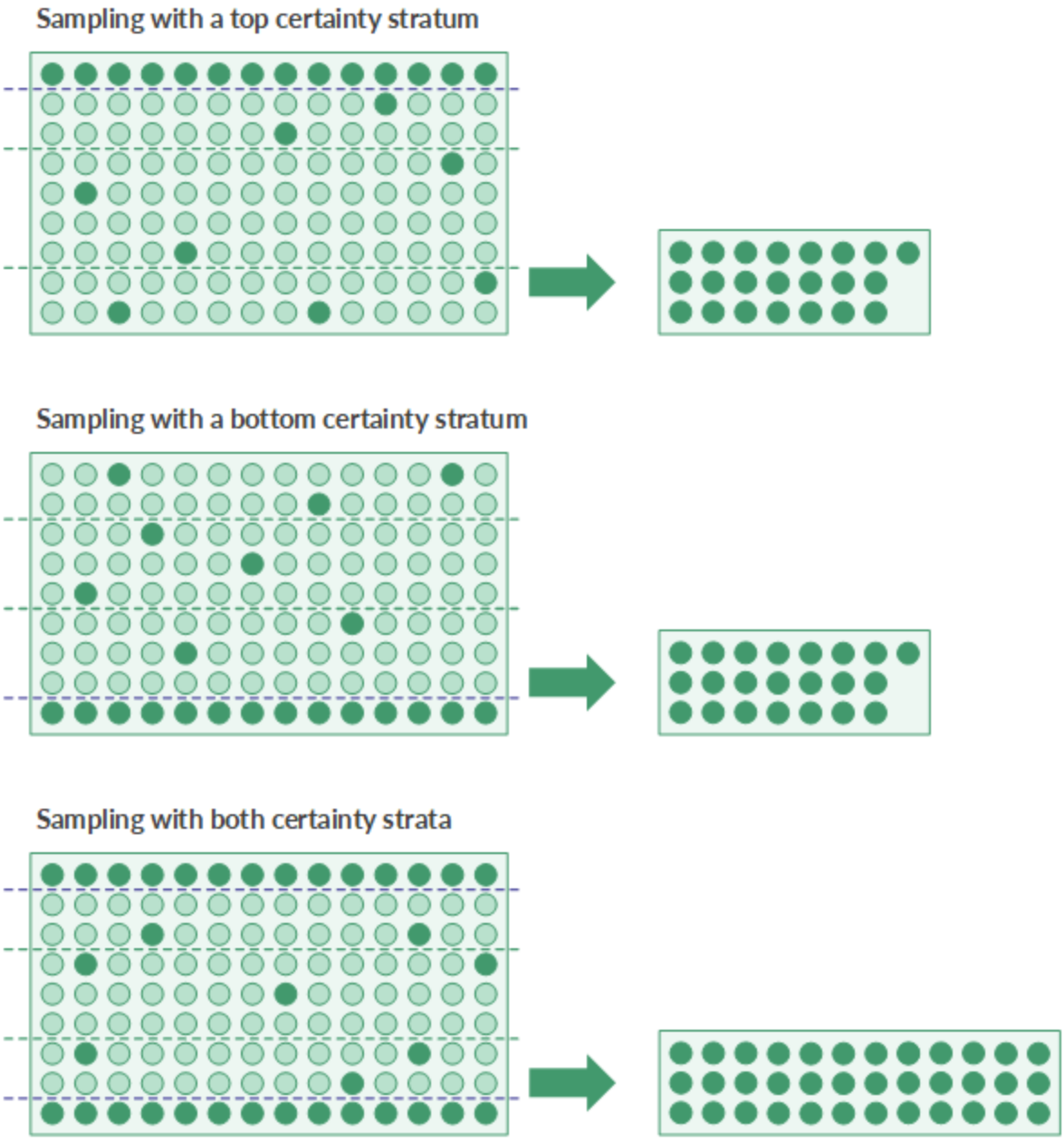
Defining a certainty stratum

To define a certainty stratum, you specify a numeric cutoff value:

- **Top certainty stratum cutoff** – All key-field book values greater than or equal to the cutoff value are automatically selected and included in the sample.
- **Bottom certainty stratum cutoff** – All key-field book values less than or equal to the cutoff value are automatically selected and included in the sample.

  Using a bottom certainty stratum is useful if large negative values are present in a population and you want to automatically include them.

The portion of the population not captured by a certainty stratum is sampled using the random selection method.



Note

Depending on the nature of the data, the overall sample size may increase as you lower the cutoff value for the top certainty stratum, or raise the cutoff value for the bottom certainty stratum.

You should avoid setting a cutoff value too generously. Consult a sampling specialist if you are unsure where to set a cutoff value.

Coordinating top and bottom certainty strata

If you decide to use both a top and a bottom certainty stratum when drawing a sample you need to consider how the top and bottom cutoff values relate:

- **Certainty strata cannot overlap** – An error occurs if you specify a top cutoff value that is less than a bottom cutoff value.
- **Leave sufficient room between the cutoff values** – If you specify cutoff values that are too close to each other, most of the population is automatically included in the sample, which defeats the purpose of sampling.

How classical variables sampling selects records

Classical variables sampling uses the following process for selecting sample records from an Analytics table:

- You specify a numeric field as the basis for the sampling. The sampling unit is an individual record in the table.
- Using the random selection method, Analytics selects samples from among the records in the table.
- If you are using stratification, a roughly equal number of records are randomly selected from each stratum.
- If you are not using stratification, records are randomly selected from the entire population.
- The selected records are included in the sampling output table.

Example

In a table with 300 records, divided into 3 strata, Analytics could select the following record numbers:

Stratum 1	Stratum 2	Stratum 3
◦ 9 ◦ 13 ◦ 40 ◦ 52 ◦ 78 ◦ 91 ◦ 99	◦ 104 ◦ 119 ◦ 132 ◦ 144 ◦ 153 ◦ 186	◦ 211 ◦ 229 ◦ 236 ◦ 248 ◦ 278 ◦ 295 ◦ 296

In an unstratified table with 300 records Analytics could select the record numbers displayed below. You can see that the selected record numbers are less evenly distributed.

> **Note**
> The record numbers below are grouped in three columns for ease of comparison, but the columns do not represent strata.

◦ 25 ◦ 64 ◦ 79 ◦ 104 ◦ 122 ◦ 127 ◦ 138	◦ 143 ◦ 175 ◦ 179 ◦ 184 ◦ 191 ◦ 201 ◦ 234	◦ 241 ◦ 257 ◦ 259 ◦ 281 ◦ 289 ◦ 299

Unbiased sample selection

Classical variables sampling is unbiased and it is not based on the amounts contained in a record. Each record has an equal chance of being selected for inclusion in the sample. A record containing a $1000 amount, a record containing a $250 amount, and a record containing a $1 amount all have the same chance of being selected.

In other words, the probability that any given record will be selected has no relation to the size of the amount it contains.

If you want to ensure that records containing the largest amounts are selected, see "The certainty strata" on page 1033.

Classical variables sampling tutorial

This tutorial introduces you to the end-to-end process of classical variables sampling in Analytics.

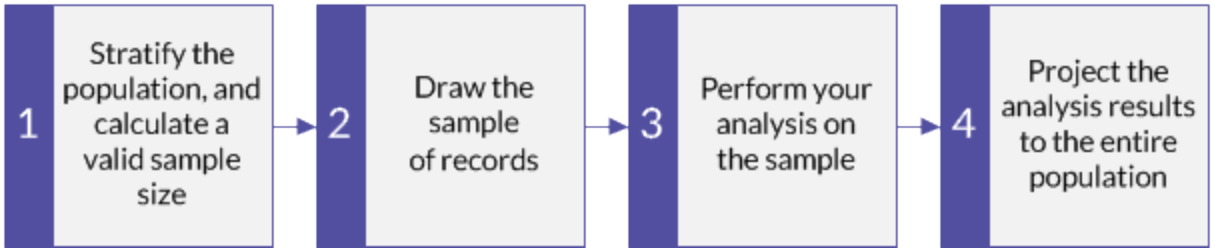
Estimated time – 30 minutes

Summary – You will draw a sample of records from an invoices table, and identify misstatements in the sample. Based on the sample results, you will make a statistical estimate of two amounts:

- the total audited value of the entire table
- the total amount of misstatement in the entire table

You then use the statistical estimate to judge whether the invoice records as a whole are fairly stated.

Main tasks – To perform classical variables sampling correctly, you need to do four main tasks:



The tutorial leaves out some optional aspects of classical variables sampling, and focuses on a single path, so you can quickly get a basic understanding of how classical variables sampling in Analytics works.

> **Tip**
> For simple definitions of sampling terms, see "A word about terminology" on page 951.

Classical variables sampling scenario

Detecting misstatement in accounts receivable

The scenario

You are examining an Invoices table with over 4000 records as part of confirming accounts receivable. You want to contact a sample of invoiced customers to confirm outstanding amounts in the account, and detect any misstatement.

You will use the customer contacts to confirm:

- receivable amounts exist
- receivable amounts are correctly recorded

How do you proceed?

How many customers should you contact? How do you decide which ones to contact? How do any misstatements you find in the sample relate to the entire account?

You can use Analytics classical variables sampling to get answers to these questions.

Analytics table used in the scenario

This scenario uses the `Invoices` table in the `ACL_Rockwood.acl` sample data file included with Analytics.

> **Note**
> Most of the amounts in the `Invoices` table in `ACL_Rockwood.acl` have a status of "Paid". For this scenario, assume they have a status of "Outstanding" and a payment amount of $0.00.

> **Note**
> Do not include the thousands separator, or the percentage sign, when you specify values. These characters prevent the command from running, or cause errors.

1 Stratify the population, and calculate a valid sample size

> **Note**
> In a production environment, the values you specify to stratify a population, and calculate a valid sample size, are dependent on your professional judgment.

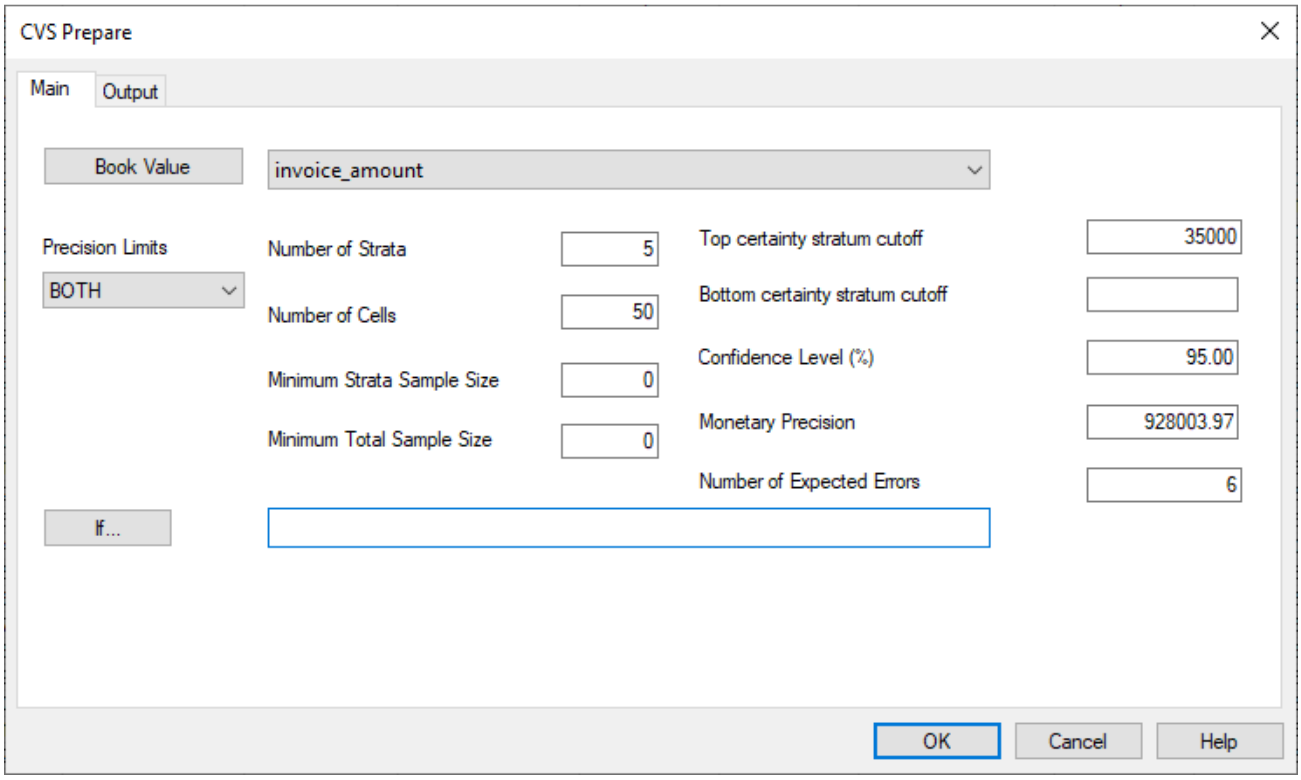
1. In `ACL_Rockwood.acl`, open the `Invoices` table, located in the `Sales_and_collection` folder.
2. Select **Sampling > Classical Variables Sampling (CVS) > Prepare**.
3. Specify the input values exactly as they appear in the screen below and click **OK**.

   Make sure the **invoice_amount** field is selected in the **Book Value** drop-down list.

   Analytics stratifies the population and calculates the sample size for each stratum, and for the population as a whole.

4. Optional. Pin the tab with the output results of the stratification and sample size calculation.

   If you pin the output results as you proceed through the classical variables sampling process, you can review the entire process once you have completed it.

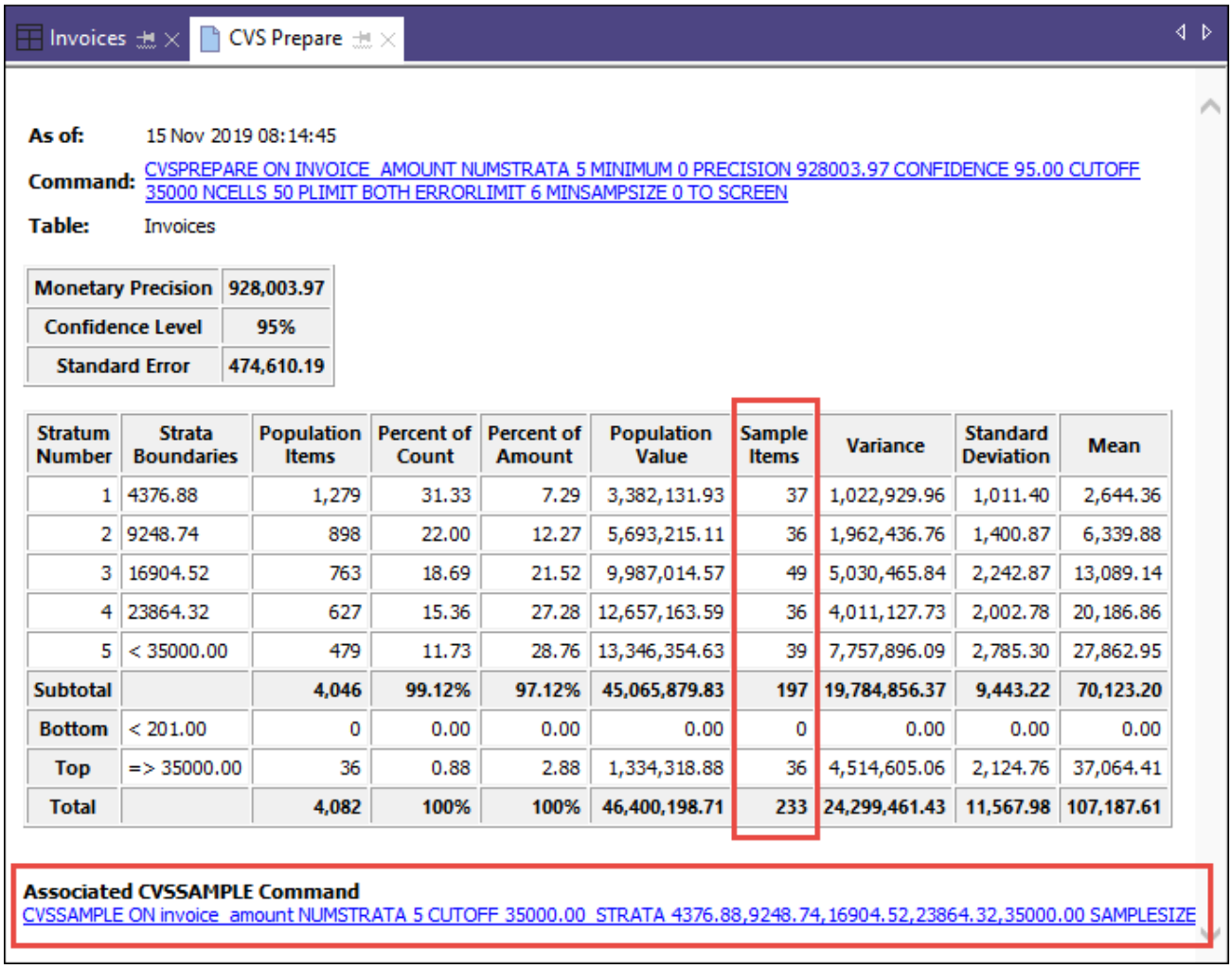


What the input values mean

Book Value	The **invoice_amount** field contains the book values you are auditing.
Precision Limits	You leave **Both** selected (the default setting) because: ◦ the account as a whole could be either overstated or understated ◦ you are interested in estimating whether misstatement in either direction could exceed the **Monetary Precision**
Number of Strata	You want to divide the population into 5 strata, or subgroups, as a way of significantly reducing the sample size.
Number of Cells	You specify 50 cells to use for pre-stratifying the population. Cells are narrower numeric divisions than strata. Pre-stratification is part of an internal process that optimizes the position of strata boundaries. Cells are not retained in the final stratified output. The number of cells must be at least twice (2 x) the number of strata.
Minimum Strata Sample	Optional.

Size	Leaving the default value of zero (0) means that you are not enforcing a minimum number of sampled records in each stratum.
Minimum Total Sample Size	Optional. Leaving the default value of zero (0) means that you are not enforcing a minimum total sample size.
Top certainty stratum cutoff	You want to sample and test 100% of the book value items greater than or equal to $35,000. Every item in the top certainty stratum will be included in the sample output table.
Bottom certainty stratum cutoff	Optional. Leaving the field empty means that you are not specifying a bottom certainty stratum.
Confidence Level (%)	You want a 95% degree of confidence that the sample you are going to draw is representative of the entire population. Put another way: if you drew the sample 100 times, it would be representative 95 times, and unrepresentative only 5 times.
Monetary Precision	Monetary precision is tolerable misstatement minus expected misstatement. You are willing to tolerate total misstatement of up to 3% of the account book value, and you expect the misstatement to be 1% of the account book value, so you are left with a monetary precision of 2%, or $928,003.97.
Number of Expected Errors	You expect the count of misstatement errors in the sample to be at least 6.

What the results mean



Invoices | CVS Prepare

As of: 15 Nov 2019 08:14:45

Command: CVSPREPARE ON INVOICE_AMOUNT NUMSTRATA 5 MINIMUM 0 PRECISION 928003.97 CONFIDENCE 95.00 CUTOFF 35000 NCELLS 50 PLIMIT BOTH ERRORLIMIT 6 MINSAMPSIZE 0 TO SCREEN

Table: Invoices

Monetary Precision	**928,003.97**
Confidence Level	**95%**
Standard Error	**474,610.19**

Stratum Number	Strata Boundaries	Population Items	Percent of Count	Percent of Amount	Population Value	Sample Items	Variance	Standard Deviation	Mean
1	4376.88	1,279	31.33	7.29	3,382,131.93	37	1,022,929.96	1,011.40	2,644.36
2	9248.74	898	22.00	12.27	5,693,215.11	36	1,962,436.76	1,400.87	6,339.88
3	16904.52	763	18.69	21.52	9,987,014.57	49	5,030,465.84	2,242.87	13,089.14
4	23864.32	627	15.36	27.28	12,657,163.59	36	4,011,127.73	2,002.78	20,186.86
5	< 35000.00	479	11.73	28.76	13,346,354.63	39	7,757,896.09	2,785.30	27,862.95
Subtotal		**4,046**	**99.12%**	**97.12%**	**45,065,879.83**	**197**	**19,784,856.37**	**9,443.22**	**70,123.20**
Bottom	< 201.00	0	0.00	0.00	0.00	0	0.00	0.00	0.00
Top	=> 35000.00	36	0.88	2.88	1,334,318.88	36	4,514,605.06	2,124.76	37,064.41
Total		**4,082**	**100%**	**100%**	**46,400,198.71**	**233**	**24,299,461.43**	**11,567.98**	**107,187.61**

Associated CVSSAMPLE Command
CVSSAMPLE ON invoice_amount NUMSTRATA 5 CUTOFF 35000.00 STRATA 4376.88,9248.74,16904.52,23864.32,35000.00 SAMPLESIZE

Sample Items	You should contact 233 customers in total. For each population stratum, and for the top certainty stratum, you should contact the specified number of customers. For example, for Stratum 3, you should contact 49 customers.
Strata Breakdown	The stratification of the population constructed by Analytics. Several descriptive values are provided for each stratum, and for the top certainty stratum, including: ◦ **Stratum Number** - a sequentially incremented number assigned to each stratum The top certainty stratum is assigned the number '0' (not displayed on this screen). ◦ **Strata Boundaries** - the upper boundaries of each stratum, and the certainty strata cutoff values ◦ **Population Items** - the count of the records in the table, broken down by stratum,

	including the top certainty stratum ◦ **Sample Items** - the total required sample size, broken down by stratum. Includes all items in the top certainty stratum.
Descriptive Statistics	Several descriptive statistics provide insight into the statistical properties of the population strata: ◦ **Standard Error** ◦ **Variance** ◦ **Standard Deviation** ◦ **Mean**
Associated CVSSAMPLE Command	The CVS Prepare results provide a prefilled version of the command that is used at the CVS Sample stage of the classical variables sampling process (the next stage). The values used to prefill the command are not saved when you close Analytics. You can manually save the prefilled command to preserve the values and avoid the work of regenerating them later. In a production environment, you may run through the CVS Prepare stage multiple times as you optimize stratification of the population and sample size. With each iteration of CVS Prepare you can manually save the associated CVSSAMPLE command.

Save the CVSSAMPLE command (optional)

> **Note**
> For the purposes of the tutorial, saving the command is not required, as long as you do not close Analytics. In a production environment, saving the command is a good idea.

Show me more

Save the `CVSSAMPLE` command in case you need to recover the values it contains.

1. At the bottom of the CVS Prepare display area, click the **CVSSAMPLE** link to load the command into the command line.
2. Copy the entire command from the command line and save it in an Analytics script with the name `CVS_Sample`.

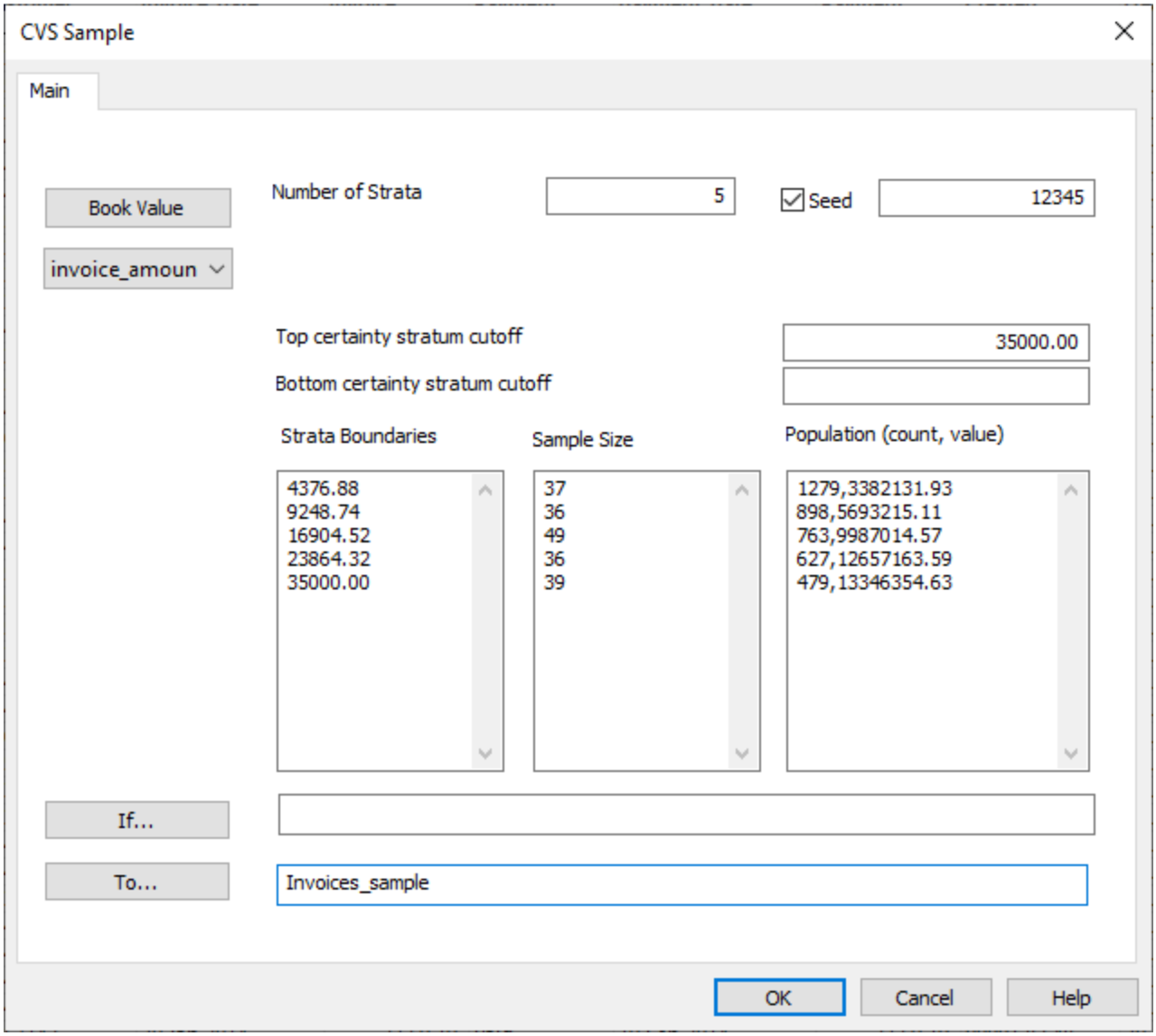
2 Draw the sample of records

1. Return to the `Invoices` table.
2. Select **Sampling > Classical Variables Sampling (CVS) > Sample**.
3. Specify the input values exactly as they appear in the screen below and click **OK** to draw the sample of records.

> **Note**
>
> Most of the values are prefilled from the output results of the CVS Prepare stage.
>
> If a number of the prefilled values are missing, see "Use the CVSSAMPLE command (optional)" on the next page.
>
> Make sure you specify the **Seed** value exactly as shown below: `12345`
>
> The seed value is used to initialize the random selection of records for the sample. If you use a different seed value, a different group of records is selected, and none of the sample amounts will match in the examples that follow.

4. Optional. Pin the tab with the summary results of the sampling process.

   If you pin the output results as you proceed through the classical variables sampling process, you can review the entire process once you have completed it.



Use the CVSSAMPLE command (optional)

> **Note**
> You can skip this section if you have kept Analytics open throughout the tutorial.

Show me more

If a number of the prefilled values are missing from the **CVS Sample** dialog box, you may have closed Analytics between the CVS Prepare and CVS Sample stages and lost the values.

Instead of using the **CVS Sample** dialog box to draw the sample of records, you can use the `CVSSAMPLE` command you saved in a script.

1. Open the CVS_Sample script and make these updates to the CVSSAMPLE command:

```
NUMSTRATA 5 SEED 12345 CUTOFF 35000.00
```

```
TO Invoices_sample
```

2. Make sure the Invoices table is open.
3. Run the script, or copy the entire command to the command line and press Enter.

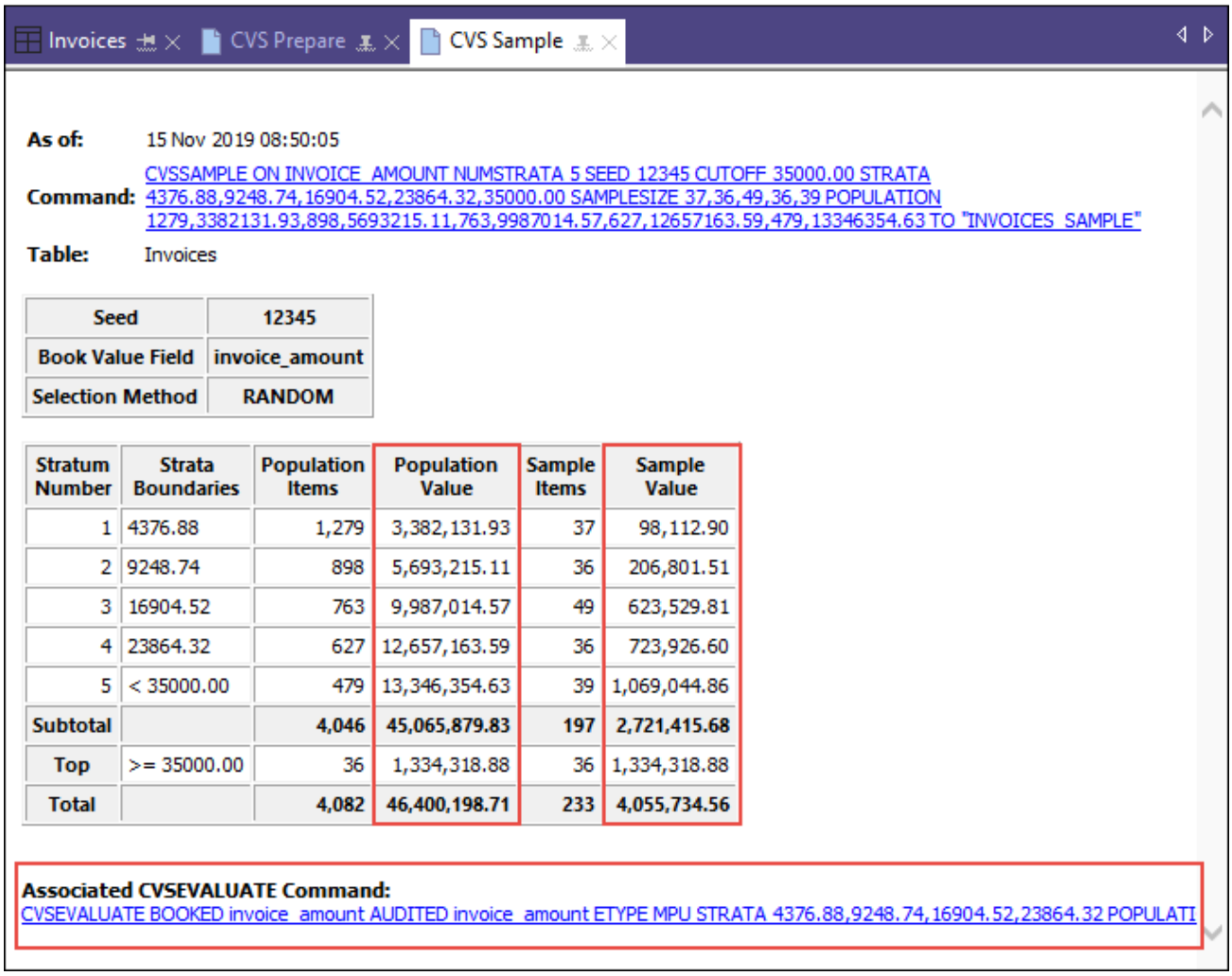
   If you run the script, double-click the CVSSAMPLE command in the log to open the CVS Sample display area.

> **Tip**
>
> If you didn't save the CVSSAMPLE command, you can perform the CVS Prepare stage again to regenerate the values required by the CVS Sample stage. You can open the Invoices table and quickly rerun the CVSPREPARE command from the log.

What the input values mean

Book Value Number of Strata Top certainty stratum cutoff Strata Boundaries Sample Size Population (count, value)	The input values are prefilled based on values that you provided, or that Analytics calculated, during the CVS Prepare stage.
Seed	Optional. Specifying a seed value is optional, but to ensure that the records included in the sample match the tutorial sample you need to specify an identical seed value.
To	The sample of records drawn from the Invoices table is output to a new table called Invoices_sample.

What the results mean



Invoices | CVS Prepare | CVS Sample

As of: 15 Nov 2019 08:50:05

Command: CVSSAMPLE ON INVOICE_AMOUNT NUMSTRATA 5 SEED 12345 CUTOFF 35000.00 STRATA 4376.88,9248.74,16904.52,23864.32,35000.00 SAMPLESIZE 37,36,49,36,39 POPULATION 1279,3382131.93,898,5693215.11,763,9987014.57,627,12657163.59,479,13346354.63 TO "INVOICES_SAMPLE"

Table: Invoices

Seed	**12345**
Book Value Field	**invoice_amount**
Selection Method	**RANDOM**

Stratum Number	Strata Boundaries	Population Items	Population Value	Sample Items	Sample Value
1	4376.88	1,279	3,382,131.93	37	98,112.90
2	9248.74	898	5,693,215.11	36	206,801.51
3	16904.52	763	9,987,014.57	49	623,529.81
4	23864.32	627	12,657,163.59	36	723,926.60
5	< 35000.00	479	13,346,354.63	39	1,069,044.86
Subtotal		**4,046**	**45,065,879.83**	**197**	**2,721,415.68**
Top	>= 35000.00	36	1,334,318.88	36	1,334,318.88
Total		**4,082**	**46,400,198.71**	**233**	**4,055,734.56**

Associated CVSEVALUATE Command:
CVSEVALUATE BOOKED invoice_amount AUDITED invoice_amount ETYPE MPU STRATA 4376.88,9248.74,16904.52,23864.32 POPULATI

Seed Book Value Field	The input values you provided.
Selection Method	Analytics used the Random selection method to draw the specified number of records from each stratum. All records in the top certainty stratum are automatically selected.
Strata Breakdown	The same breakdown that appears in the output of the CVS Prepare stage. Now that actual sample records have been drawn, Analytics can calculate the **Sample Value** for each stratum in the sample, and for the sample as a whole. Note the difference between the **Sample Value** and the **Population Value**.
Associated CVSEVALUATE Com-	The CVS Sample results provide a prefilled version of the command that is used at the CVS Evaluate stage of the classical variables sampling process (the final

mand	stage). The values used to prefill the command are not saved when you close Analytics. You can manually save the prefilled command to preserve the values and avoid the work of regenerating them later. In a production environment, several weeks could elapse between the CVS Sample and the CVS Evaluate stages during which you are performing audit procedures on the sample data.

Save the CVSEVALUATE command (optional)

> **Note**
> For the purposes of the tutorial, saving the command is not required, as long as you do not close Analytics. In a production environment, saving the command is a good idea.

Show me more

Save the `CVSEVALUATE` command in case you need to recover the values it contains.

1. At the bottom of the CVS Sample display area, click the **CVSEVALUATE** link to load the command into the command line.
2. Copy the entire command from the command line and save it in an Analytics script with the name `CVS_Evaluate`.

Add the Audit_Value field and export the sample table

Because Analytics is a read-only application, you need to export the table of sampled records to Excel so that you can add audit values.

Before you export the table, you need to add a field that replicates the **Invoice Amount** field. In Excel, you edit the replicated field.

Add the Audit_Value field

1. Close the `Invoices` table.
2. Open the `Invoices_sample` table.
3. Copy and paste this command into the command line:

```
DEFINE FIELD AUDIT_VALUE COMPUTED invoice_amount
```

If the command line is not visible, select **Window > Command Line**.

4. Press Enter to create the new field.

> **Tip**
> If you want to confirm that the AUDIT_VALUE field was created, type `display` in the command line and press Enter to see a list of the fields in the table.

Export the sample table to Excel

1. Select **Data > Export**.
2. On the **Main** tab, make sure **Fields** is selected.
3. Click **Export Fields > Add All**.
4. Select each of these fields and use the Up arrow  to move them to the top of the **Selected Fields** list:
   - SAMPLE_RECORD_NUMBER
   - STRATUM
   - invoice_amount
   - AUDIT_VALUE

   Keep the order shown here.
5. Click **OK**.
6. In the **Export As** drop-down list, select **Excel (*.xlsx)**.
7. In the **To** field, type **`Invoices_sample_audited`**, and click **OK**.
8. Click the **Output to** link to open the Excel file.

> **Note**
> Do not close Analytics.

3 Perform your analysis on the sample

For the purposes of the tutorial, assume that you do the following:

1. Contact the customers that appear in the `Invoices_sample_audited` table.
2. Confirm receivable amounts and record any misstatements.

Update the Excel file

1. Update the Excel file with the values listed in the **AUDIT_VALUE** field in the table below.
2. Save and close the Excel file.

You are specifying:

- three misstatements per stratum
- one misstatement in the top certainty stratum
- both overstatements and understatements

Note

To make it easier to update the Excel file, and to demonstrate how the subsequent evaluation works:

- the updates are all in the first few records in the file
- understatements, and a mix of understatements and overstatements, are each grouped in a particular stratum

In a production environment, the likelihood is that overstatements and understatements would be scattered across strata, and throughout the file.

Tip

Copy and paste the entire table below to an blank Excel worksheet, and then copy the audit values to the **AUDIT_VALUE** column in the **Invoices_sample** worksheet.

As an alternative, you can download a text file with a column of the audit values.

SAMPLE_RECORD_NUMBER	STRATUM	Invoice Amount	AUDIT_VALUE
1	3	9,394.55	9,494.55
2	5	27,033.66	17,033.66
3	4	22,617.90	22,917.90
4	2	4,576.24	4,575.83
5	1	4,039.67	0.00
6	3	13,753.12	31,753.12
7	4	23,633.12	23,433.12
8	5	33,663.50	33,660.00
9	2	7,136.79	6,136.79
10	2	4,495.13	0.00
11	1	1,575.87	1,075.87
12	0	44,379.67	34,379.67
13 (audit value unchanged)	0	35,159.99	35,159.99
14	5	27,204.08	27,200.00
15	4	20,156.50	20,000.00
16 (audit value unchanged)	0	37,448.07	37,448.07

SAMPLE_RECORD_NUMBER	STRATUM	Invoice Amount	AUDIT_VALUE
17	3	11,879.05	11,889.05
18	1	994.98	964.98

4 Project the analysis results to the entire population

Import the updated sample table

> **Note**
> Make sure the Excel file with the audit values that you updated is closed.

1. In Analytics, select **Import > File**.
2. In the **Select File to Define** dialog box, locate and select `..\Sample Data Files\ACL_Rockwood\Invoices_sample_audited.xlsx` and click **Open**.
3. In the **File Format** page, verify that **Excel file** is selected and click **Next**.
4. In the **Data Source** page, select the **Invoices_sample** worksheet.
5. Make sure these options are set as indicated and click **Next**:
   - **Use first row as Field Names** is selected
   - **Start On Line** is 1
   - **Import all fields as character type** is **not** selected
6. In the **Excel Import** page, verify that the **invoice_amount** field and the **AUDIT_VALUE** field have a **Type** of **Numeric**, and click **Next**.

   Select the field headers to check their assigned **Type**. If necessary, update the **Type** to **Numeric**, and specify `2` in the **Decimal** field.
7. In the **Save Data File As** dialog box, type `Invoices_sample_audited` in the **File name** field, and click **Save**.
8. In the **Final** page, click **Finish**.
9. In the dialog box for specifying a table layout name, click **OK**.

   A new Analytics table is created with the data from the imported Excel file.

Evaluate the results of the sample analysis

1. Open the `Invoices_sample_audited` table if it is not already open.
2. Select **Sampling > Classical Variables Sampling (CVS) > Evaluate**.

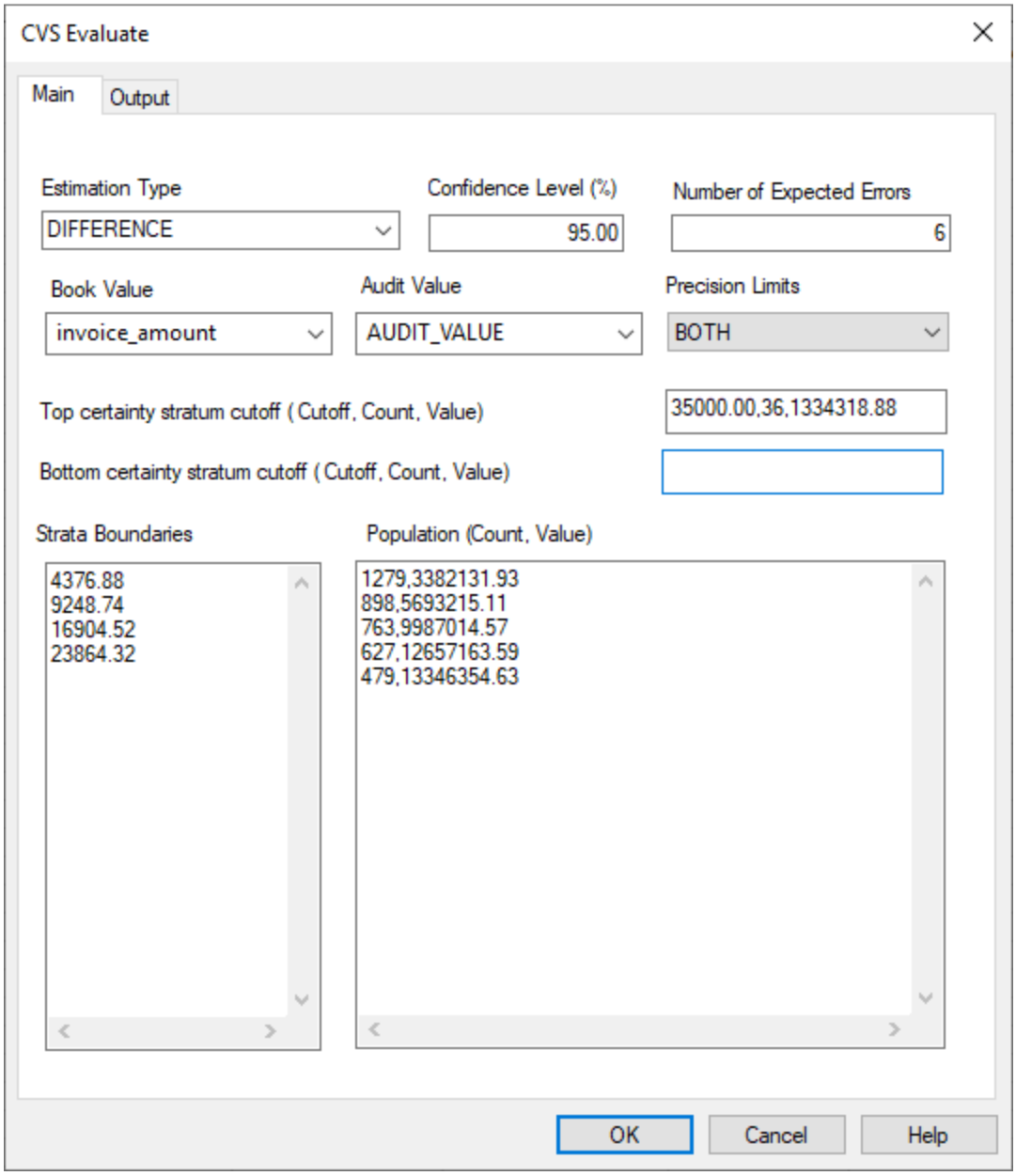
   > **Note**
   > The menu option is disabled if a table is not open.

3. Specify the input values exactly as they appear in the screen below and click **OK** to project the sample results to the entire population.

   > **Note**
   > Most of the values are prefilled from the output results of the CVS Prepare and CVS Sample stages.
   >
   > If a number of the prefilled values are missing, see "Use the CVSEVALUATE command (optional)" on the next page.

4. Optional. Pin the tab with the evaluation results.

   If you pin the output results as you proceed through the classical variables sampling process, you can review the entire process once you have completed it.



Use the CVSEVALUATE command (optional)

> **Note**
> You can skip this section if you have kept Analytics open throughout the tutorial.

Show me more

If a number of the prefilled values are missing from the **CVS Evaluate** dialog box, you may have closed Analytics between the CVS Sample and the CVS Evaluate stages and lost the values.

Instead of using the **CVS Evaluate** dialog box to evaluate results, you can use the `CVSEVALUATE` command you saved in a script.

1. Open the `CVS_Evaluate` script and make these updates to the `CVSEVALUATE` command:

   ```
   AUDITED AUDIT_VALUE ETYPE DIFFERENCE
   ```

2. Make sure the `Invoices_sample_audited` table is open.
3. Run the script, or copy the entire command to the command line and press Enter.

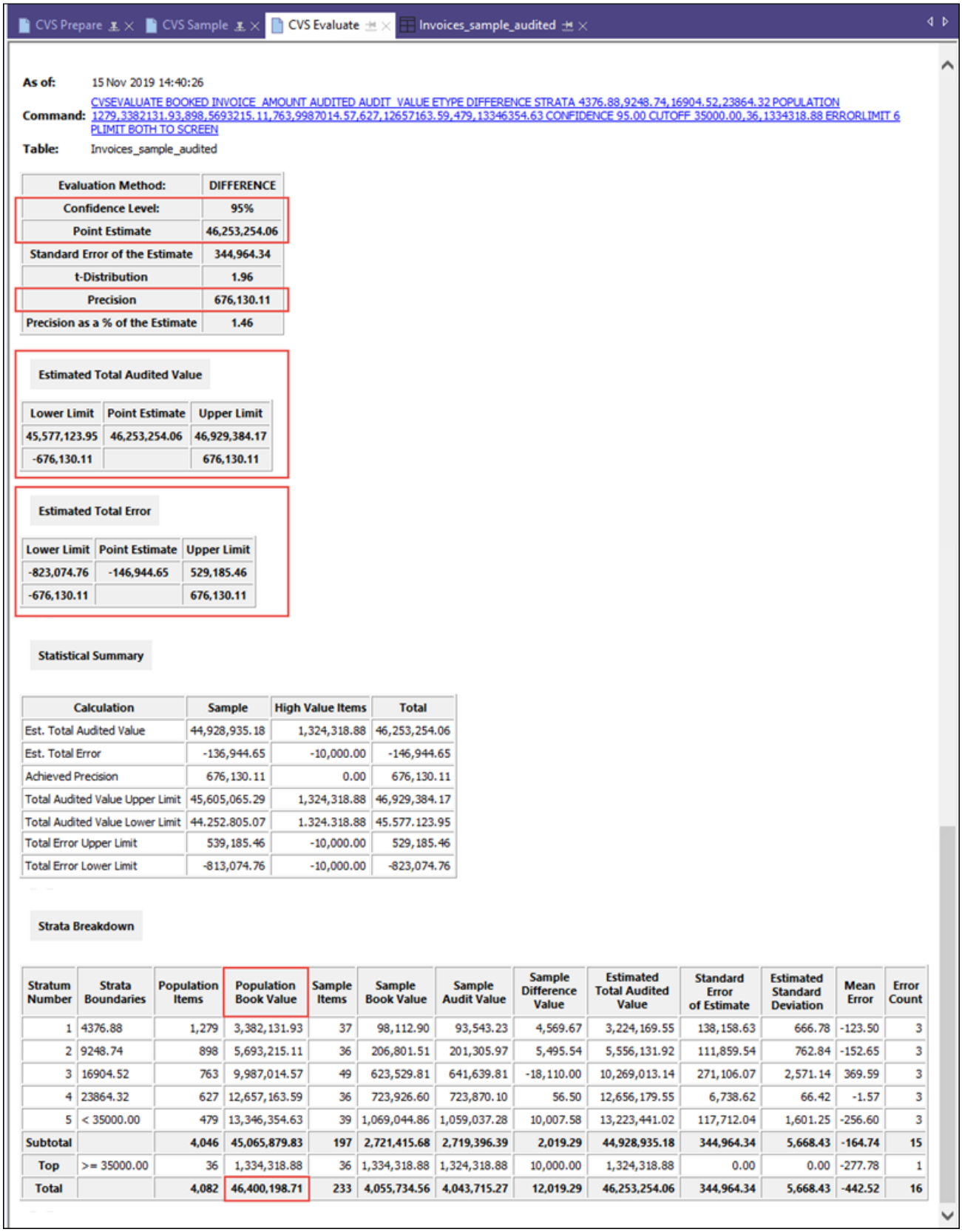
   If you run the script, double-click the `CVSEVALUATE` command in the log to open the evaluation results.

> **Tip**
>
> If you didn't save the `CVSEVALUATE` command, you can perform the CVS Prepare and CVS Sample stages again to regenerate the values required by the CVS Evaluate stage. You can open the `Invoices` table and quickly rerun the `CVSPREPARE` and `CVSSAMPLE` commands from the log.

What the input values mean

Estimation Type	You are specifying the **Difference** estimation type. Classical variables sampling in Analytics has four different ways of estimating the results of sample analysis when projected to the entire population.
Audit Value	The **AUDIT_VALUE** field contains the audit values.
Confidence Level (%) Number of Expected Errors Book Value Precision Limits Strata Boundaries Population (Count, Value) Top certainty stratum cutoff (Cutoff, Count, Value)	The input values are prefilled based on values that you provided, or that Analytics calculated, during the CVS Prepare stage.

What the projected results mean



CVS Prepare | CVS Sample | CVS Evaluate | Invoices_sample_audited

As of: 15 Nov 2019 14:40:26

Command: CVSEVALUATE BOOKED INVOICE_AMOUNT AUDITED AUDIT_VALUE ETYPE DIFFERENCE STRATA 4376.88,9248.74,16904.52,23864.32 POPULATION 1279,3382131.93,898,5693215.11,763,9987014.57,627,12657163.59,479,13346354.63 CONFIDENCE 95.00 CUTOFF 35000.00,36,1334318.88 ERRORLIMIT 6 PLIMIT BOTH TO SCREEN

Table: Invoices_sample_audited

Evaluation Method:	**DIFFERENCE**
Confidence Level:	**95%**
Point Estimate	**46,253,254.06**
Standard Error of the Estimate	**344,964.34**
t-Distribution	**1.96**
Precision	**676,130.11**
Precision as a % of the Estimate	**1.46**

Estimated Total Audited Value

Lower Limit	Point Estimate	Upper Limit
45,577,123.95	46,253,254.06	46,929,384.17
-676,130.11		676,130.11

Estimated Total Error

Lower Limit	Point Estimate	Upper Limit
-823,074.76	-146,944.65	529,185.46
-676,130.11		676,130.11

Statistical Summary

Calculation	Sample	High Value Items	Total
Est. Total Audited Value	44,928,935.18	1,324,318.88	46,253,254.06
Est. Total Error	-136,944.65	-10,000.00	-146,944.65
Achieved Precision	676,130.11	0.00	676,130.11
Total Audited Value Upper Limit	45,605,065.29	1,324,318.88	46,929,384.17
Total Audited Value Lower Limit	44.252.805.07	1.324.318.88	45.577.123.95
Total Error Upper Limit	539,185.46	-10,000.00	529,185.46
Total Error Lower Limit	-813,074.76	-10,000.00	-823,074.76

Strata Breakdown

Stratum Number	Strata Boundaries	Population Items	Population Book Value	Sample Items	Sample Book Value	Sample Audit Value	Sample Difference Value	Estimated Total Audited Value	Standard Error of Estimate	Estimated Standard Deviation	Mean Error	Error Count
1	4376.88	1,279	3,382,131.93	37	98,112.90	93,543.23	4,569.67	3,224,169.55	138,158.63	666.78	-123.50	3
2	9248.74	898	5,693,215.11	36	206,801.51	201,305.97	5,495.54	5,556,131.92	111,859.54	762.84	-152.65	3
3	16904.52	763	9,987,014.57	49	623,529.81	641,639.81	-18,110.00	10,269,013.14	271,106.07	2,571.14	369.59	3
4	23864.32	627	12,657,163.59	36	723,926.60	723,870.10	56.50	12,656,179.55	6,738.62	66.42	-1.57	3
5	< 35000.00	479	13,346,354.63	39	1,069,044.86	1,059,037.28	10,007.58	13,223,441.02	117,712.04	1,601.25	-256.60	3
Subtotal		**4,046**	**45,065,879.83**	**197**	**2,721,415.68**	**2,719,396.39**	**2,019.29**	**44,928,935.18**	**344,964.34**	**5,668.43**	**-164.74**	**15**
Top	>= 35000.00	36	1,334,318.88	36	1,334,318.88	1,324,318.88	10,000.00	1,324,318.88	0.00	0.00	-277.78	1
Total		**4,082**	**46,400,198.71**	**233**	**4,055,734.56**	**4,043,715.27**	**12,019.29**	**46,253,254.06**	**344,964.34**	**5,668.43**	**-442.52**	**16**

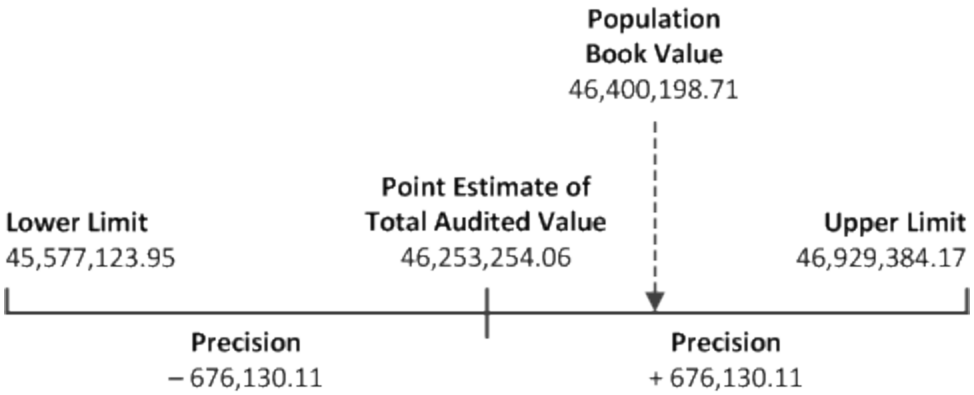
Point Estimate Precision	**Point Estimate (46,253,254.06)** - a statistical projection of the most likely audited value of the entire `Invoices` table **Precision (676,130.11)** - a statistical projection of the amount by which the **Point Estimate** could vary ◦ The statistical projections are based on the audit values in the `Invoices_sample_audited` table. ◦ The **Point Estimate** is the midpoint of an estimated range. ◦ The **Point Estimate** plus or minus the **Precision** forms the upper and lower limits of the range.
Estimated Total Audited Value	A visual presentation of the **Estimated Total Audited Value** range. **How the range works** ◦ The **Population Book Value** of the `Invoices` table is within the range: *The account is very likely to be fairly stated.* ◦ The **Population Book Value** of the `Invoices` table is outside either the upper limit or the lower limit of the range: *The account could be materially misstated.*
Estimated Total Error	A visual presentation of the **Estimated Total Error** range. **How the Error range is calculated** ◦ The **Point Estimate** of total error is **Estimated Total Audited Value** minus **Population Book Value**. ◦ The **Point Estimate** plus or minus the **Precision** forms the upper and lower limits of the range. ◦ A negative error amount indicates overstatement error, and a positive error amount indicates understatement error. **How the range works** ◦ The **Estimated Total Error** range is within the range of zero (0) ±928,003.97, the **Monetary Precision** you specified during the CVS Prepare stage: *The account is very likely to be fairly stated.* ◦ Either limit of the **Estimated Total Error** range is outside either limit of the **Monetary Precision** range: *The account could be materially misstated.*

Judging whether the invoice records as a whole are fairly stated

You have two ways of judging whether the invoice records as a whole are fairly stated. You can look at the estimated total audited value, or you can look at the estimated total error.

Estimated Total Audited Value

The **Population Book Value** of 46,400,198.71 falls within the **Estimated Total Audited Value** range, so the account as a whole is very likely to be fairly stated.

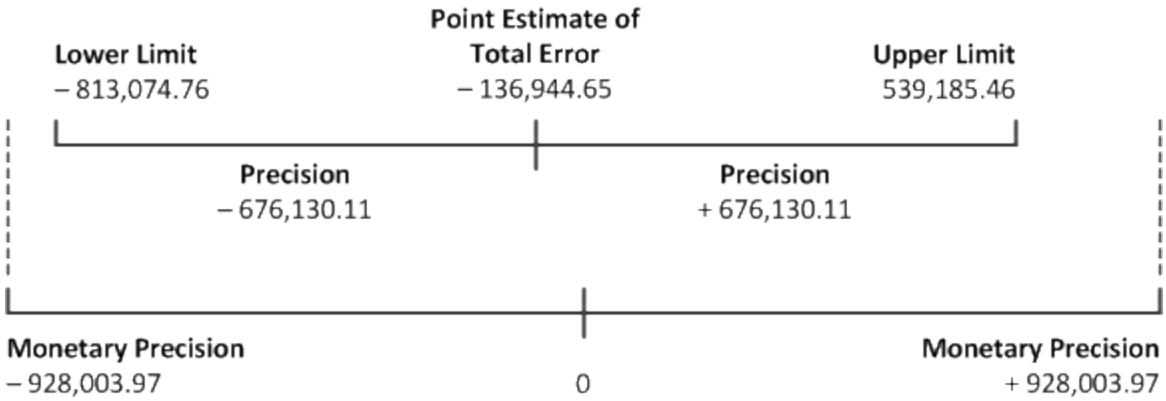


The evaluation results support making this sort of statement:

There is a 95% probability that the true audited value of the account is between 45,577,123.95 and 46,929,384.17, a range that contains the account book value of 46,400,198.71. Therefore the amounts in the account are fairly stated.

Estimated Total Error

The **Estimated Total Error** range falls within the **Monetary Precision** range, so the account as a whole is very likely to be fairly stated.



The evaluation results support making this sort of statement:

There is a 95% probability that the misstatement in the account balance is between -813,074.76 and 539,185.46, which does not exceed the monetary precision of ±928,003.97. Therefore the amounts in the account are fairly stated.

Preparing a classical variables sample

Before sampling a set of data, you must stratify the table containing the records, and calculate a statistically valid sample size for each stratum.

The **CVS Prepare** feature in Analytics calculates the required values for you based on input values you provide.

The importance of calculating a sample size

Calculating an appropriate sample size is critical to the validity of the subsequent sample. If the sample is not valid, or representative, you cannot reliably project the results of audit procedures you perform on the sample to the entire population.

Do not skip calculating a sample size, or guess at a sample size.

Most of the input values you use to calculate sample size are based on your professional judgment. Ensure that you fully understand the implications of the values before relying on the results of sampling in a production environment. Consult audit sampling resources, or an audit sampling specialist, if you are in doubt.

Numeric length limitation

Several internal calculations occur during the preparation stage of classical variables sampling. These calculations support numbers with a maximum length of 17 digits. If the result of any calculation exceeds 17 digits, the result is not included in the output, and you cannot continue with the sampling process.

Note that source data numbers of less than 17 digits can produce internal calculation results that exceed 17 digits.

How precision limits work

When you prepare a classical variables sample you must choose one of the options listed below for **Precision Limits**.

The option you choose dictates the type of estimation range you generate during the CVS Evaluate stage of the sampling process.

You need to choose an option now, during the preparation stage, because the option you choose is one of the requirements for calculating the sample size.

Precision Limits	For this type of estimate during CVS Evaluate:
Both	A two-sided range with a point estimate and an upper and lower limit
Upper	A one-sided range with a point estimate and an upper limit
Lower	A one-sided range with a point estimate and a lower limit

Show me more

The "sided-ness" of the range

The "sided-ness" of the range is derived from the normal distribution, or bell curve, that forms the basis of classical variables sampling.

A two-sided range

If you are examining an account that as a whole could be either overstated or understated then typically you are interested in whether misstatement in either direction exceeds the amount of misstatement that you judge is tolerable or acceptable.

You need a two-sided range or estimate:

- The lower limit is an estimate of the maximum amount of overstatement that could exist in the account for the confidence level you specify
- The upper limit is an estimate of the maximum amount of understatement that could exist in the account for the confidence level you specify

A one-sided range

If you are examining an account that as a whole is likely to be overstated, or likely to be understated, you may only be interested in whether misstatement in one direction exceeds the amount of misstatement that you judge is tolerable or acceptable.

You can use a one-sided range or estimate:

- A range with only a lower limit is an estimate of the maximum amount of overstatement that could exist in the account for the confidence level you specify
- A range with only an upper limit is an estimate of the maximum amount of understatement that could exist in the account for the confidence level you specify

Should I use a two-sided or one-sided range?

During the CVS Evaluate stage, using a two-sided range is the more prudent choice. A two-sided range allows you to judge whether an account is fairly stated regardless of the overall direction of misstatement in the account.

The benefit of using a one-sided range is that you can reduce the required sample size, which reduces the required labor and cost of analyzing the sample data. The amount of reduction varies, but typically is less than 50%.

The risk of using a one-sided range is that if you are wrong about the overall direction of misstatement in an account, you could miss a material misstatement in the unexamined direction and make an incorrect judgment about the fairness of an account balance.

How input values affect sample size

Input values affect the sample size calculated by Analytics. You can experiment with different input values in the **CVS Prepare** dialog box to see how they affect the sample size.

Show me more

> **Caution**
> In a production environment, do not manipulate input values solely to achieve a smaller sample size. Input values should be based on your professional judgment about what is most appropriate for the data being sampled and the audit objective.

Increasing this input value:	Decreases sample size	Increases sample size
Number of Strata	✔ After a certain point, increasing **Number of Strata** has little or no effect on sample size	
Number of Cells	✔ Increases or decreases sample size depending on the nature of the data, but generally decreases sample size After a certain point, increasing **Number of Cells** has little or no effect on sample size	
Minimum Strata Sample Size		✔ Increases sample size if the minimum threshold applies in one or more strata
Minimum Total Sample		✔

Increasing this input value:	Decreases sample size	Increases sample size
Size		Increases sample size if the minimum threshold applies
Top certainty stratum cutoff	✔ Increases or decreases sample size depending on the nature of the data If values in the data are relatively evenly distributed, a higher **Top certainty stratum cutoff** decreases sample size	
Bottom certainty stratum cutoff		✔ Increases or decreases sample size depending on the nature of the data If values in the data are relatively evenly distributed, a higher **Bottom certainty stratum cutoff** increases sample size **Note** For negative cutoff values, "higher" means closer to zero (0).
Confidence Level (%)		✔
Monetary Precision	✔	
Number of Expected Errors	Has no effect on sample size	
Precision Limits	**Both** requires a larger sample size than **Upper** or **Lower**	

Steps

> **Note**
> Do not include the thousands separator, or the percentage sign, when you specify values. These characters prevent the command from running, or cause errors.

1. Open the table containing the book values that you intend to sample.
2. Select **Sampling > Classical Variables Sampling (CVS) > Prepare**.

> **Note**
> The menu option is disabled if a table is not open.

3. On the **Main** tab, select the book value field from the **Book Value** drop-down list.
4. In the **Precision Limits** drop-down list, select an appropriate option:
   - **Both**
   - **Upper**
   - **Lower**

   > **Note**
   > The options are explained in detail below.

5. Enter the input values to use for preparing the sample design:
   - **Number of Strata**
   - **Number of Cells**
   - **Minimum Strata Sample Size**
   - **Minimum Total Sample Size**
   - **Top certainty stratum cutoff**
   - **Bottom certainty stratum cutoff**
   - **Confidence Level (%)**
   - **Monetary Precision**
   - **Number of Expected Errors**

   > **Note**
   > The input values are explained in detail below.

6. Optional. If there are records in the current view that you want to exclude from processing, enter a condition in the **If** text box, or click **If** to create an IF statement using the **Expression Builder**.

   > **Caution**
   > If you specify a conditional expression, an identical conditional expression must be used during both the calculation of the sample size, and the drawing of the sample.
   >
   > If you use a condition at one stage and not the other, or if the two conditions are not identical, the sampling results will probably not be statistically valid.

7. On the **Output** tab:
   a. In the **To** panel, select one of the following:
      - **Screen** – displays the results in the Analytics display area

        > **Tip**
        > You can click any linked result value in the display area to drill down to the associated record or records in the source table.

      - **File** – saves or appends the results to a text file

        The file is saved outside Analytics.

b. If you selected **File** as the output type, do one of the following:
   - Enter a file name in the **Name** text box.
   - Click **Name** and enter the file name, or select an existing file in the **Save** or **Save File As** dialog box to overwrite or append to the file.

     If Analytics prefills a file name, you can accept the prefilled name, or change it.

     You can also specify an absolute or relative file path, or navigate to a different folder, to save or append the file in a location other than the project location. For example: `C:\Results\Output.txt` or `Results\Output.txt`.

     > **Note**
     > **ASCII Text File** or **Unicode Text file** (depending on which edition of Analytics you are using) is the only option for **File Type**.

8. Click **OK**.

   The CVS preparation output results are displayed on screen, or saved to a file.

   Included in the display is a prefilled version of the CVSSAMPLE command.

   > **Note**
   > The output results are explained in detail below.

Save the CVSSAMPLE command for later use (optional)

As a convenience, you can save the CVSSAMPLE command to use once you have prepared a classical variables sample.

1. At the bottom of the CVS Prepare display area, click the **CVSSAMPLE** link to load the command into the command line.
2. Copy the entire command from the command line and save it in an Analytics script.

   After you have prepared the sample, you can use either the **CVS Sample** dialog box, or the CVSSAMPLE command, to draw the sample of records.

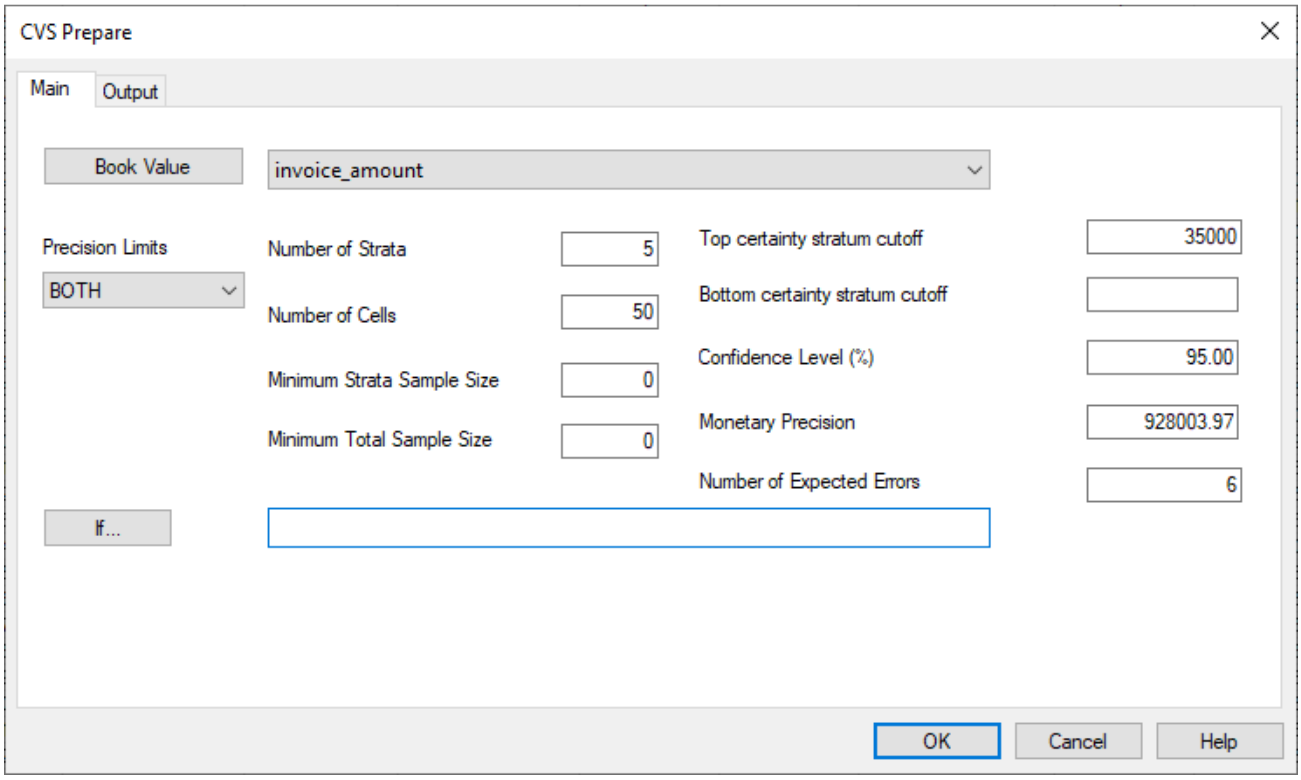
   > **Note**
   > If you use the CVSSAMPLE command, you need to add the name of the output table to the command, and a seed value (optional).
   >
   > For more information, see "CVSSAMPLE command" on page 1618.

CVS Prepare dialog box inputs and results

The tables below provide detailed information about the input values in the **CVS Prepare** dialog box, and the output results.

Main tab – input values

Example screen from classical variables sampling tutorial



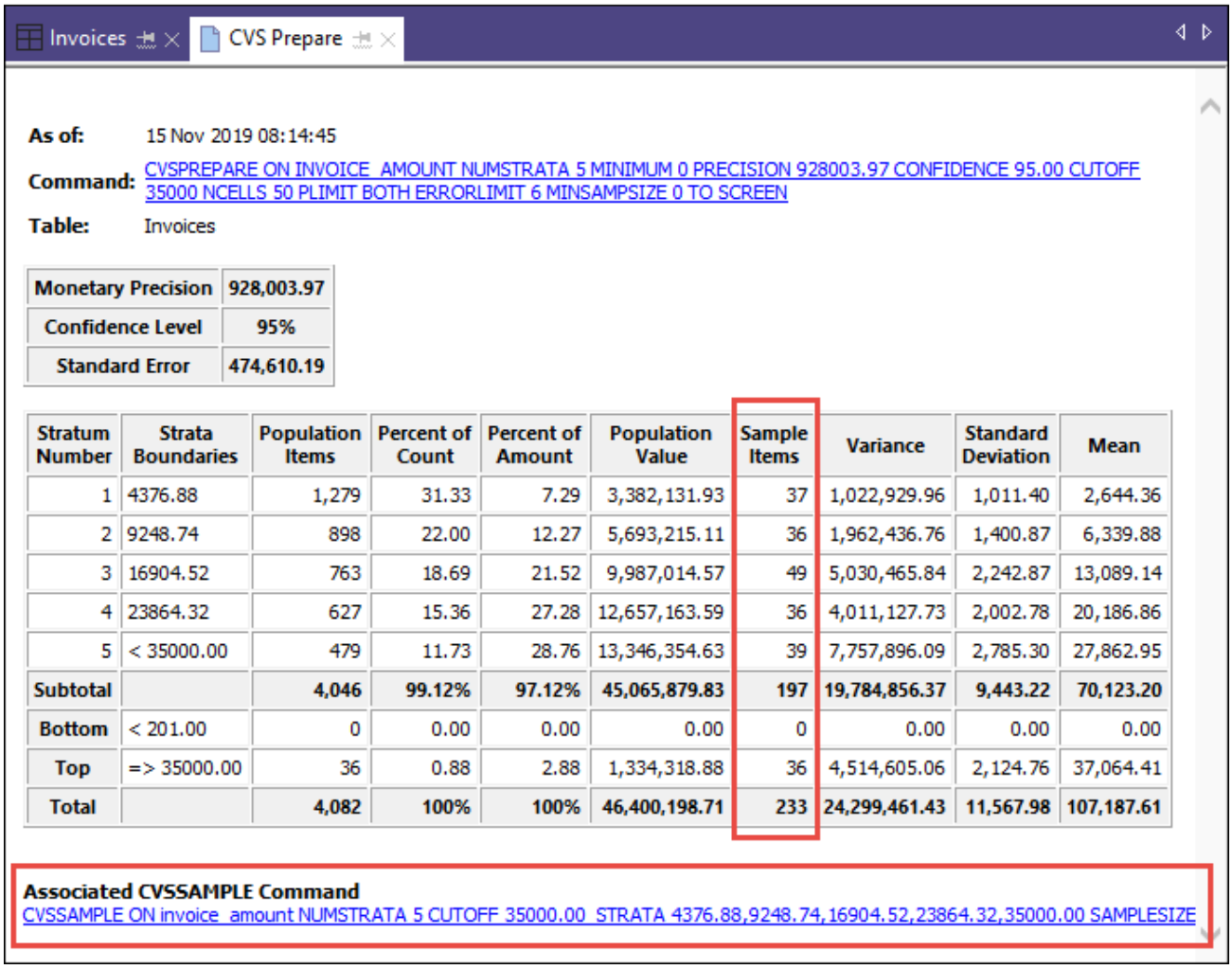
Input values – CVS Prepare dialog box	Description
Book Value	The field that contains the book values you are auditing.
Precision Limits	The type of precision limit to use. **Both** Select this option if: ◦ the account as a whole could be either overstated or understated ◦ you are interested in estimating whether misstatement in either direction exceeds the **Monetary Precision** **Upper**

Input values - CVS Prepare dialog box	Description
	Select this option if: ◦ the account as a whole is likely to be understated ◦ you are only interested in estimating whether the total amount of understatement exceeds the **Monetary Precision** **Lower** Select this option if: ◦ the account as a whole is likely to be overstated ◦ you are only interested in estimating whether the total amount of overstatement exceeds the **Monetary Precision** **Caution** Select **Both** if you are not sure which option to select. For more information see "How precision limits work" on page 1058.
Number of Strata	The number of strata (subgroups) to use for numerically stratifying the data set. The minimum number of strata is 1, and the maximum is 256. If you specify a certainty stratum, it is not included in the **Number of Strata**. For more information, see "Stratification" on page 1031. **Note** The **Number of Strata** cannot exceed 50% of the **Number of Cells**.
Number of Cells	The number of cells to use for pre-stratifying the data set. The minimum number of cells is 2, and the maximum is 999. For more information, see "Stratification" on page 1031. **Note** The **Number of Cells** must be at least twice (2 x) the **Number of Strata**.
Minimum Strata Sample Size	The minimum number of records to sample from each stratum.
Minimum Total Sample Size	The minimum number of records to sample from the entire data set.
Top certainty stratum cutoff	Optional. A top certainty stratum cutoff value. Amounts in the **Book Value** field greater than or equal to the cutoff value are automatically selected and included in the sample. If you do not specify a cutoff value, a default cutoff value is used that ensures no records are included in the top certainty stratum: `greater than (>) the maximum amount in the Book Value field`

Input values – CVS Prepare dialog box	Description
	For more information, see "The certainty strata" on page 1033.
Bottom certainty stratum cutoff	Optional. A bottom certainty stratum cutoff value. Amounts in the **Book Value** field less than or equal to the cutoff value are automatically selected and included in the sample. If you do not specify a cutoff value, a default cutoff value is used that ensures no records are included in the bottom certainty stratum: `less than (<) the minimum amount in the Book Value field` For more information, see "The certainty strata" on page 1033.
Confidence Level (%)	Your desired confidence level that the resulting sample is representative of the entire population. For example, entering 95 means that you want to be confident that 95% of the time the sample will in fact be representative. Confidence is the complement of “sampling risk”. A 95% confidence level is the same as a 5% sampling risk. ◦ If the **Precision Limit** is **Upper** or **Lower**, the minimum confidence level is 55%, and the maximum is 99.5%. ◦ If the **Precision Limit** is **Both**, the minimum confidence level is 10%, and the maximum is 99.5%.
Monetary Precision	The monetary amount that is the difference between the tolerable misstatement and the expected misstatement in the account. For example, if the tolerable misstatement is $29,000, and the expected misstatement is $5,800, enter 23200 (a difference of $23,200). The monetary precision establishes the range of acceptability for an account to be considered fairly stated.
Number of Expected Errors	Optional. The minimum number of errors you expect in the sample. This value is not used in any of the CVS calculations. Instead, it is used to trigger a notification in either of these situations: ◦ **Number of Expected Errors** is less than **Number of Strata**. ◦ During the CVS Evaluate stage, the actual number of errors you found in the sample is less than the **Number of Expected Errors** In either of these situations, the only evaluation method available is mean-per-unit.

Output results

Example screen from classical variables sampling tutorial



Invoices | CVS Prepare

As of: 15 Nov 2019 08:14:45

Command: CVSPREPARE ON INVOICE_AMOUNT NUMSTRATA 5 MINIMUM 0 PRECISION 928003.97 CONFIDENCE 95.00 CUTOFF 35000 NCELLS 50 PLIMIT BOTH ERRORLIMIT 6 MINSAMPSIZE 0 TO SCREEN

Table: Invoices

Monetary Precision	**928,003.97**
Confidence Level	**95%**
Standard Error	**474,610.19**

Stratum Number	Strata Boundaries	Population Items	Percent of Count	Percent of Amount	Population Value	Sample Items	Variance	Standard Deviation	Mean
1	4376.88	1,279	31.33	7.29	3,382,131.93	37	1,022,929.96	1,011.40	2,644.36
2	9248.74	898	22.00	12.27	5,693,215.11	36	1,962,436.76	1,400.87	6,339.88
3	16904.52	763	18.69	21.52	9,987,014.57	49	5,030,465.84	2,242.87	13,089.14
4	23864.32	627	15.36	27.28	12,657,163.59	36	4,011,127.73	2,002.78	20,186.86
5	< 35000.00	479	11.73	28.76	13,346,354.63	39	7,757,896.09	2,785.30	27,862.95
Subtotal		**4,046**	**99.12%**	**97.12%**	**45,065,879.83**	**197**	**19,784,856.37**	**9,443.22**	**70,123.20**
Bottom	< 201.00	0	0.00	0.00	0.00	0	0.00	0.00	0.00
Top	=> 35000.00	36	0.88	2.88	1,334,318.88	36	4,514,605.06	2,124.76	37,064.41
Total		**4,082**	**100%**	**100%**	**46,400,198.71**	**233**	**24,299,461.43**	**11,567.98**	**107,187.61**

Associated CVSSAMPLE Command
CVSSAMPLE ON invoice_amount NUMSTRATA 5 CUTOFF 35000.00 STRATA 4376.88,9248.74,16904.52,23864.32,35000.00 SAMPLESIZE

Output results - CVS Prepare	Description
Monetary Precision	The monetary precision that you specified as an input.
Confidence Level	The confidence level that you specified as an input.
Stratum Number	A sequentially incremented number assigned to each stratum. The certainty strata are also assigned numbers, although they are not displayed on this screen: ◦ **-1** - bottom certainty stratum ◦ **0** - top certainty stratum
Strata Boundaries	The upper boundaries of each stratum, and the bottom and top certainty stratum cutoff values. Book values are assigned to a stratum if they are: ◦ less than or equal to the upper boundary ◦ greater than the boundary immediately below

Output results - CVS Prepare	Description
	Book values are assigned to the bottom certainty stratum if they are less than or equal to the cutoff value. Book values are assigned to the top certainty stratum if they are greater than or equal to the cutoff value.
Population Items	The count of the records in the table, broken down by stratum, including the certainty strata.
Percent of Count	The percentage of the record count contained in each stratum, including the certainty strata.
Percent of Amount	The percentage of the total book value contained in each stratum, including the certainty strata.
Population Value	The total book value of the table, broken down by stratum, including the certainty strata.
Sample Items	The total required sample size, broken down by stratum. Includes all items in the certainty strata.
Associated CVSSAMPLE command	The command for performing the CVS Sample stage, prefilled with values from the CVS Prepare stage.

Performing classical variables sampling

You can create a new table that contains a representative sample of the monetary data in the active table.

Classical variables sampling is appropriate if you are interested in either:

- the total audited value of a file
- the total amount of monetary misstatement in a file

Because Analytics is a read-only application, after you have drawn the sample of records you need to export the sample table to an application that allows data entry, such as Excel, so that you can add audit values.

Follow the correct process for classical variables sampling

Before drawing a sample of records, you must stratify the table containing the records, and calculate a statistically valid sample size for each stratum.

For more information, see "Preparing a classical variables sample" on page 1058.

Draw the sample of records

> **Note**
> Do not include the thousands separator, or the percentage sign, when you specify values. These characters prevent the command from running, or cause errors.

1. Open the table containing the book values that you intend to sample.
2. Select **Sampling > Classical Variables Sampling (CVS) > Sample**.

> **Note**
> The menu option is disabled if a table is not open.

The **CVS Sample** dialog box opens. If you are using the output results of the CVS Prepare stage as input for the sampling stage, most of the fields are prefilled with the required values.

If a number of the prefilled values are missing, you can:

- rerun the CVSPREPARE command from the log to regenerate the values
- use the CVSSAMPLE command generated during the CVS Prepare stage, if you saved it

> **Note**
> If you use the saved CVSSAMPLE command, you need to add the name of the output table to the command, and a seed value (optional).
> For more information, see "CVSSAMPLE command" on page 1618.

3. If you are not using prefilled values, or you want to adjust one or more values, enter or update the required values:
   - **Book Value**
   - **Number of Strata**
   - **Top certainty stratum cutoff**
   - **Bottom certainty stratum cutoff**
   - **Strata Boundaries**
   - **Sample Size**
   - **Population (count, value)**

   > **Note**
   > The input values are explained in detail below.

   > **Caution**
   > Normally you should not change any of the prefilled values. Changing prefilled values can negate the statistical validity of the sampling process.

4. Optional. Select **Seed**, and enter a number.

   The **Seed** value is explained below.

5. If you used a conditional expression during the CVS Prepare stage, make sure that the **If** text box contains an identical expression.

   > **Note**
   > An If condition specified during the CVS Prepare stage is automatically propagated to the CVS Sample stage.
   > If you use a conditional expression, an identical expression must be used during both stages to ensure that the sampling results are statistically valid.

6. In the **To** text box, specify the name of the Analytics table that will contain the output results.

   > **Tip**
   > Use the book value table name and add the suffix `_sample`.

7. Click **OK**.

   The random sample of records is drawn from the book value table and saved to the output table you specified.

A summary of the sample process and output results is displayed on screen.

Included in the display is a prefilled version of the CVSEVALUATE command.

> **Note**
> The output results are explained in detail below.

Save the CVSEVALUATE command for later use (optional)

As a convenience, you can save the CVSEVALUATE command to use once you have performed the analysis of the sampled records.

1. At the bottom of the CVS Sample display area, click the **CVSEVALUATE** link to load the command into the command line.
2. Copy the entire command from the command line and save it in an Analytics script.

   After you have analyzed the sampled data, and added any audit values, you can use either the **CVS Evaluate** dialog box, or the CVSEVALUATE command, to project the results of the analysis to the entire population.

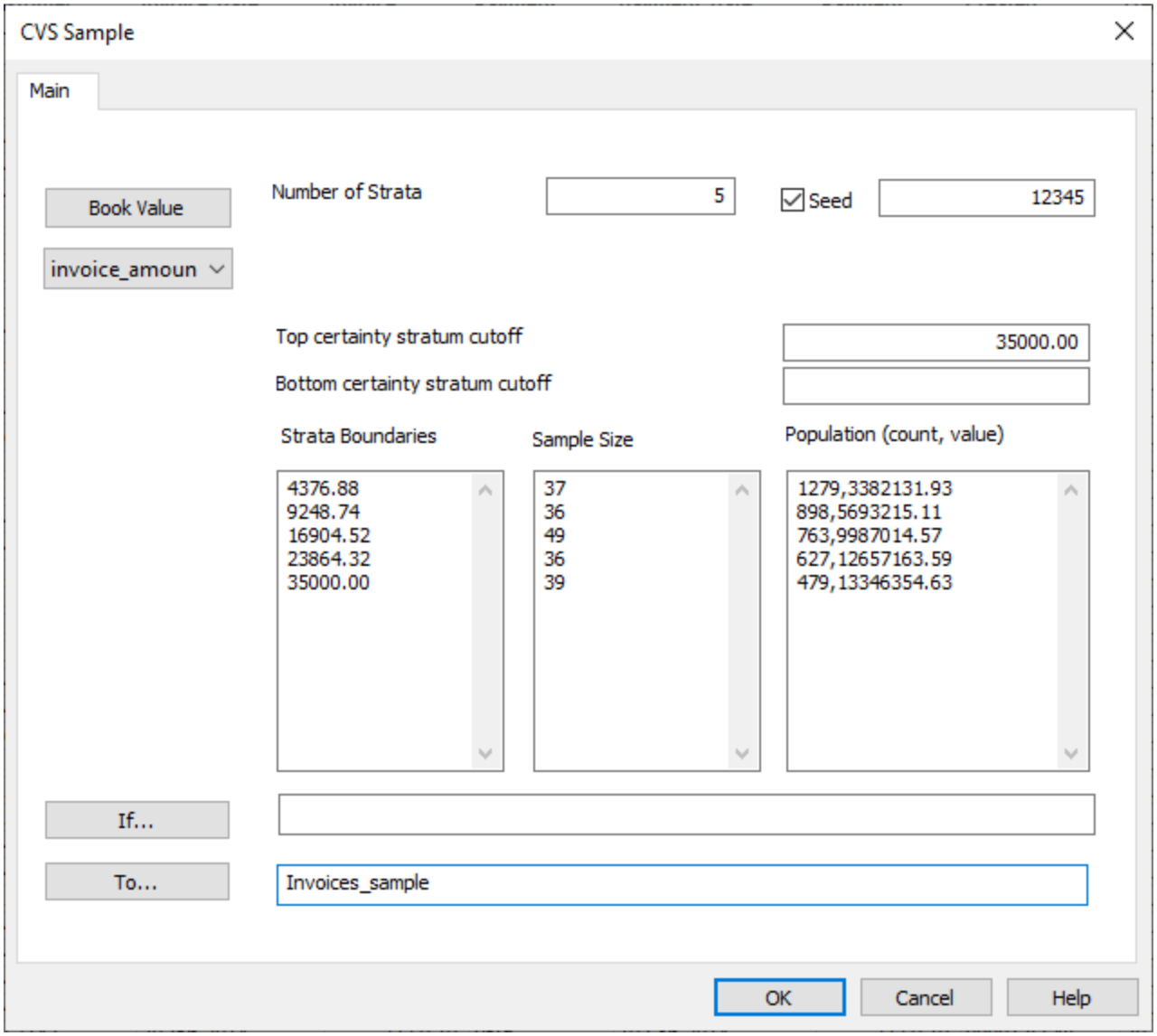
> **Note**
> If you use the CVSEVALUATE command, you need to update the name of the audit value field, and possibly the evaluation type.
> For more information, see "CVSEVALUATE command" on page 1608.

CVS Sample dialog box inputs and results

The tables below provide detailed information about the input values in the **CVS Sample** dialog box, and the output results.

Main tab – input values

Example screen from classical variables sampling tutorial

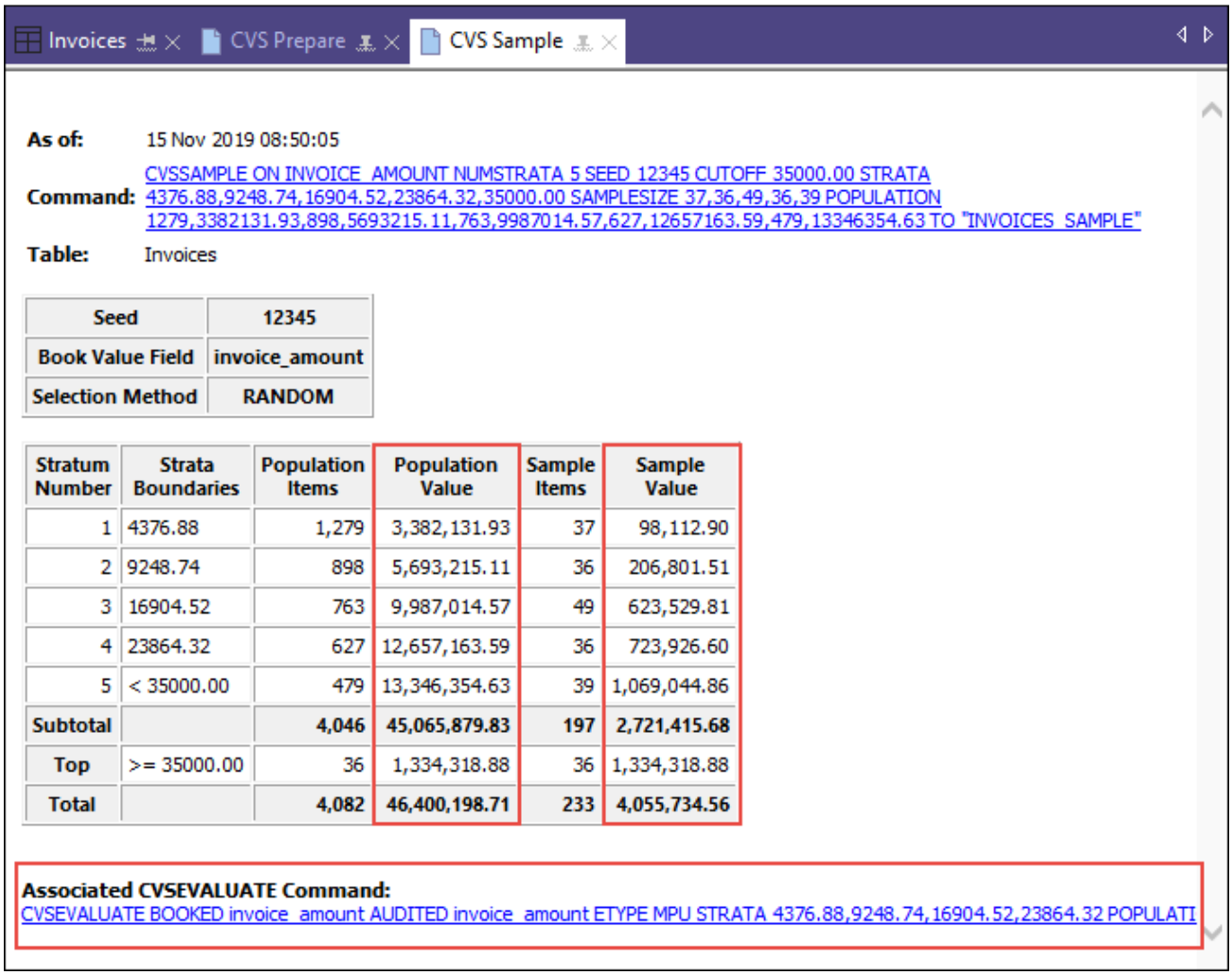


Input values – CVS Sample dialog box	Description
Book Value	The field that contains the book values you are auditing.
Number of Strata	The number of strata (subgroups) to use for numerically stratifying the data set. If you specify a certainty stratum, it is not included in the **Number of Strata**.
Top certainty stratum cutoff	A top certainty stratum cutoff value. Amounts in the **Book Value** field greater than or equal to the cutoff value are automatically selected and included in the sample.
Bottom certainty stratum cutoff	A bottom certainty stratum cutoff value.

Input values – CVS Sample dialog box	Description
	Amounts in the **Book Value** field less than or equal to the cutoff value are automatically selected and included in the sample.
Strata Boundaries	The boundary values to use for stratifying the data set.
Sample Size	The number of records to sample from each stratum.
Population (count, values)	The number of records in each stratum, and the total value for each stratum.
Seed	Optional. Can be any number. This number is used to initialize the random number generator in Analytics. **Tip** Leave the seed blank if you want Analytics to randomly select a seed value.

Output results

Example screen from classical variables sampling tutorial



Invoices | CVS Prepare | CVS Sample

As of: 15 Nov 2019 08:50:05

Command: CVSSAMPLE ON INVOICE_AMOUNT NUMSTRATA 5 SEED 12345 CUTOFF 35000.00 STRATA 4376.88,9248.74,16904.52,23864.32,35000.00 SAMPLESIZE 37,36,49,36,39 POPULATION 1279,3382131.93,898,5693215.11,763,9987014.57,627,12657163.59,479,13346354.63 TO "INVOICES_SAMPLE"

Table: Invoices

Seed	**12345**
Book Value Field	**invoice_amount**
Selection Method	**RANDOM**

Stratum Number	Strata Boundaries	Population Items	Population Value	Sample Items	Sample Value
1	4376.88	1,279	3,382,131.93	37	98,112.90
2	9248.74	898	5,693,215.11	36	206,801.51
3	16904.52	763	9,987,014.57	49	623,529.81
4	23864.32	627	12,657,163.59	36	723,926.60
5	< 35000.00	479	13,346,354.63	39	1,069,044.86
Subtotal		**4,046**	**45,065,879.83**	**197**	**2,721,415.68**
Top	>= 35000.00	36	1,334,318.88	36	1,334,318.88
Total		**4,082**	**46,400,198.71**	**233**	**4,055,734.56**

Associated CVSEVALUATE Command:
CVSEVALUATE BOOKED invoice_amount AUDITED invoice_amount ETYPE MPU STRATA 4376.88,9248.74,16904.52,23864.32 POPULATI

Output results - CVS Sample	Description
Seed	The seed value used to initialize the Analytics random number generator. The seed value was either specified by you, or randomly selected by Analytics.
Book Value Field	The book value field that you specified as an input.
Selection Method	**RANDOM** - the sample selection method used by Analytics.
Stratum Number	A sequentially incremented number assigned to each stratum. The certainty strata are also assigned numbers, although they are not displayed on this screen: ◦ **-1** - bottom certainty stratum ◦ **0** - top certainty stratum
Strata Boundaries	The upper boundaries of each stratum, and the bottom and top certainty stratum cutoff values.

Output results - CVS Sample	Description
	Book values are assigned to a stratum if they are: ◦ less than or equal to the upper boundary ◦ greater than the boundary immediately below Book values are assigned to the bottom certainty stratum if they are less than or equal to the cutoff value. Book values are assigned to the top certainty stratum if they are greater than or equal to the cutoff value.
Population Items	The count of the records in the source table, broken down by stratum, and including the certainty strata.
Population Value	The total book value of the source table, broken down by stratum, and including the certainty strata.
Sample Items	The total number of records drawn in the sample, broken down by stratum. All records in the certainty strata are automatically drawn and included in the output table.
Sample Value	The total value of the records drawn in the sample, broken down by stratum, and including the value of the certainty strata records.
Associated CVSEVALUATE Command	The command for performing the CVS Evaluate stage, prefilled with values from the CVS Prepare and CVS Sample stages.

System-generated fields

Analytics automatically generates four fields and adds them to the sample output table. For each record included in the sample, the fields contain the following descriptive information:

- **Stratum** - the number of the stratum to which the record is allocated
- **Original record number** - the original record number in the source data table
- **Selection order** - on a per-stratum basis, the order in which the record was randomly selected
- **Sample record number** - the record number in the sample output table

Add the Audit_Value field and export the sample table

Because Analytics is a read-only application, you need to export the table of sampled records to an application that allows data entry, such as Excel. In the external application you can add any audit values that you identify in the process of analyzing the sampled data.

Once the audit values have been added, you import the table back to Analytics.

Before you export the table, you need to add a field that replicates the book value field. You can then edit this replicated field as required in the external application.

Add the Audit_Value field

1. Close the book value table.
2. Open the newly created table containing the sampled records.

   For example: *<book_value_table>*_sample
3. Copy and paste this command into the command line:

   ```
   DEFINE FIELD AUDIT_VALUE COMPUTED book_value_field
   ```

   If the command line is not visible, select **Window > Command Line**.
4. Replace `book_value_field` with the actual name of the field in your table that contains the book values.

   For example:

   ```
   DEFINE FIELD AUDIT_VALUE COMPUTED invoice_amount
   ```

   > **Note**
   > Make sure to use the physical name of the field, not the display name, if the two names are different.

5. Press Enter to create the new field.

   > **Tip**
   > If you want to confirm that the AUDIT_VALUE field was created, enter `display` in the command line and press Enter to see a list of the fields in the table.

Export the sample table to Excel

Export the sample table to Excel, or to a delimited file for use in another suitable external application.

When you perform the export:

- select the **Fields** option, and select all fields to export
- specify a name for the exported table that will be easy to identify later, such as the sample table name with the added suffix `_audited`

  For example: *<book_value_table>*_sample_audited

For more information, see "Exporting data" on page 203.

Evaluating errors in a classical variables sample

After you have performed your audit procedures on a set of sampled data you can use Analytics to:

- project the sample audit value to the entire account
- project any misstatements you found to the entire account
- calculate upper and lower limits on estimated total audit value, and estimated total misstatement amount

Even if you found no errors, you still use the evaluation feature to calculate the basic allowance for sampling risk. In this situation, you must use the mean-per-unit estimation type.

Follow the correct process for classical variables sampling

Evaluating errors is the final stage in the classical variables sampling process. You must have performed the previous stages before you can evaluate errors.

For more information, see:

- "Preparing a classical variables sample" on page 1058
- "Performing classical variables sampling" on page 1069

Which estimation type should I use?

The CVS Evaluate stage gives you the option of four different estimation types (evaluation methods):

- Mean-per-unit
- Difference
- Ratio separate
- Ratio combined

The estimation type that you should use depends on the nature of the data: the sample book values, the sample audit values, and the relation between them.

Guidelines

The guidelines below help you select an estimation type.

> **Tip**
>
> If you want to compare the results produced by different estimation types, you can select **All** in the **Estimation Type** drop-down list. Selecting **All** includes all estimation types in the evaluation output.

Show me more

Estimation type	Presence of misstatements	Size of misstatements	Sign of book values	Comparison of strata ratios
Mean-per-unit	**No misstatements, or very few misstatements** The only valid estimation type if there are no misstatements, or very few misstatements, in the audited sample population.	n/a	n/a	n/a
Difference	**Misstatements required** Requires a number of misstatements in the audited sample population. For example, 5% or more of the samples contain misstatements.	**Misstatements are non-proportional** More suitable when misstatements are non-proportional: the size of a misstatement is not related to the size of the associated book value. In other words, small and large book values can have either small or large misstatements.	n/a	n/a
Ratio Separate		**Misstatements are proportional** More suitable when misstatements are proportional: the size of a misstatement is related to the size of the associated book value. In other words, small book values have small	**Book values have the same sign** All sample book values must have the same sign: either all positive, or all negative.	**Ratios vary** More suitable when the ratio of average sample audit value to average sample book value varies widely between strata.

Estimation type	Presence of misstatements	Size of misstatements	Sign of book values	Comparison of strata ratios
		misstatements, and large book values have large misstatements.		
Ratio Combined				**Ratios are consistent** More suitable when the ratio of average sample audit value to average sample book value is relatively consistent between strata.

Import the updated sample table

Import the updated sample table to Analytics from Excel, or from whatever external application you used to add audit values.

The table must contain:

- **a book value field** – the original recorded book values that existed when you drew the sample
- **an audit value field** – the audited values, updated where necessary based on the results of your analysis

For more information, see "Import Microsoft Excel data" on page 234.

Evaluate the results of the sample analysis

> **Note**
> Do not include the thousands separator, or the percentage sign, when you specify values. These characters prevent the command from running, or cause errors.

1. Open the updated sample table that you just imported.
2. Select **Sampling > Classical Variables Sampling (CVS) > Evaluate**.

> **Note**
> The menu option is disabled if a table is not open.

The **CVS Evaluate** dialog box opens. If you are using the output results of the CVS Prepare and the CVS Sample stages as input for the evaluation stage, most of the fields are prefilled with the required values.

If a number of the prefilled values are missing, you can:

- rerun the CVSSAMPLE command from the log to regenerate the values
- use the CVSEVALUATE command generated during the CVS Sample stage, if you saved it

> **Note**
> If you use the saved CVSEVALUATE command, you need to update the name of the audit value field, and possibly the evaluation type.
> For more information, see "CVSEVALUATE command" on page 1608.

3. On the **Main** tab, select one of the following options from the **Estimation Type** drop-down list:
   - **MPU**
   - **Difference**
   - **Ratio Separate**
   - **Ratio Combined**
   - **All**

   > **Note**
   > The options are explained in detail above.

4. If you are not using prefilled values, or you want to adjust one or more values, enter or update the required values:
   - **Confidence Level (%)**
   - **Number of Expected Errors**
   - **Book Value**
   - **Audit Value**
   - **Precision Limits**
   - **Top certainty stratum cutoff (Cutoff, Count, Value)**
   - **Bottom certainty stratum cutoff (Cutoff, Count, Value)**
   - **Strata Boundaries**
   - **Population (Count, Value)**

   > **Note**
   > The input values are explained in detail below.

   > **Caution**
   > Normally you should not change any of the prefilled values. Changing prefilled values can negate the statistical validity of the evaluation process.

5. On the **Output** tab:
   a. In the **To** panel, select one of the following:
      - **Screen** - displays the results in the Analytics display area

      > **Tip**
      > You can click any linked result value in the display area to drill down to the associated record or records in the source table.

- **File** – saves or appends the results to a text file

  The file is saved outside Analytics.

b. If you selected **File** as the output type, do one of the following:
   - Enter a file name in the **Name** text box.
   - Click **Name** and enter the file name, or select an existing file in the **Save** or **Save File As** dialog box to overwrite or append to the file.

     If Analytics prefills a file name, you can accept the prefilled name, or change it.

     You can also specify an absolute or relative file path, or navigate to a different folder, to save or append the file in a location other than the project location. For example: `C:\Results\Output.txt` or `Results\Output.txt`.

     > **Note**
     > **ASCII Text File** or **Unicode Text file** (depending on which edition of Analytics you are using) is the only option for **File Type**.

6. Click **OK**.

   The CVS Evaluate output results are displayed, or saved to a file.

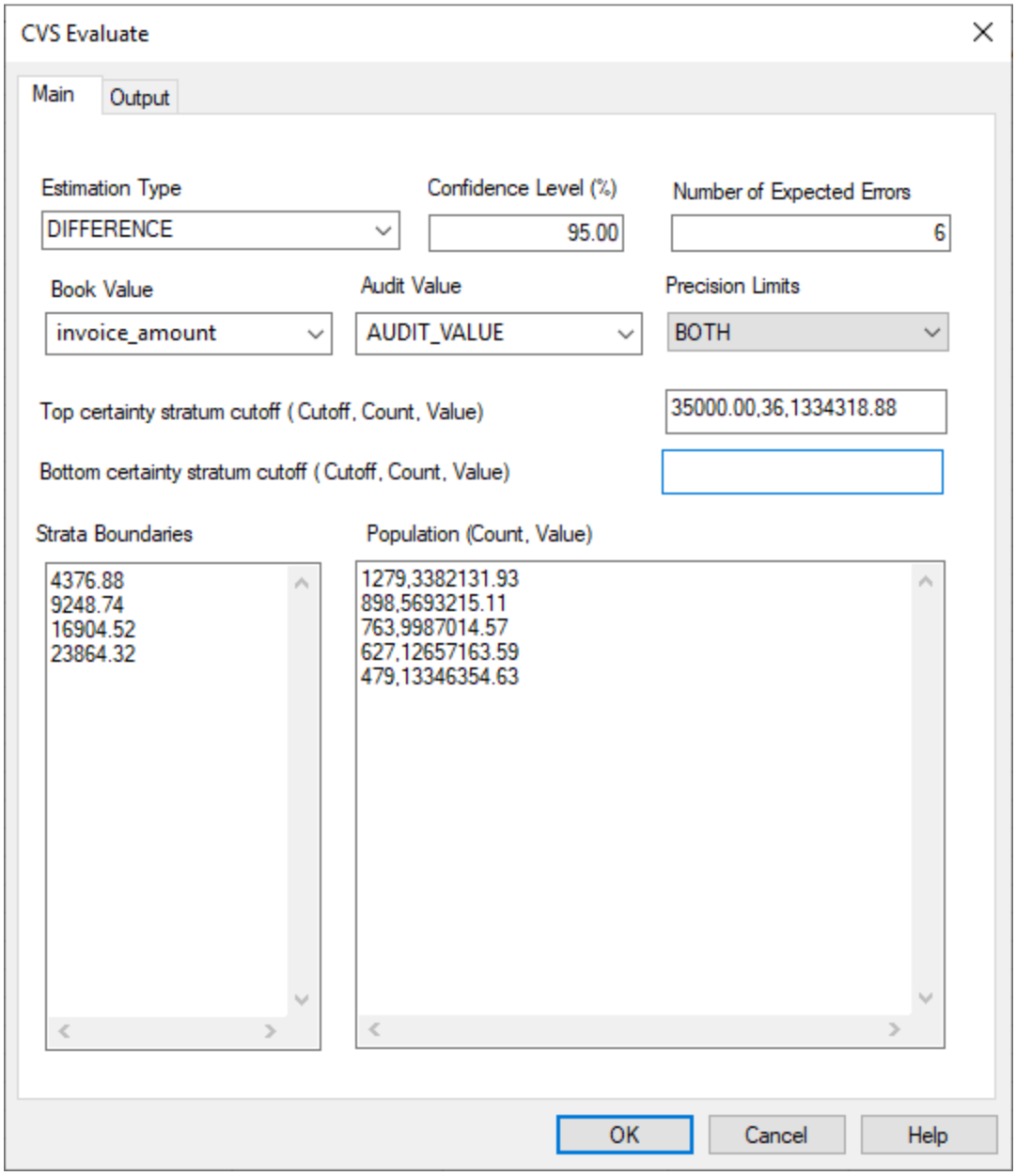
   > **Note**
   > The output results are explained in detail below.
   >
   > For additional information about interpreting output results, see "Judging whether the invoice records as a whole are fairly stated" on page 1056.

CVS Evaluate dialog box inputs and results

The tables below provide detailed information about the input values in the **CVS Evaluate** dialog box, and the output results.

Main tab – input values

Example screen from classical variables sampling tutorial

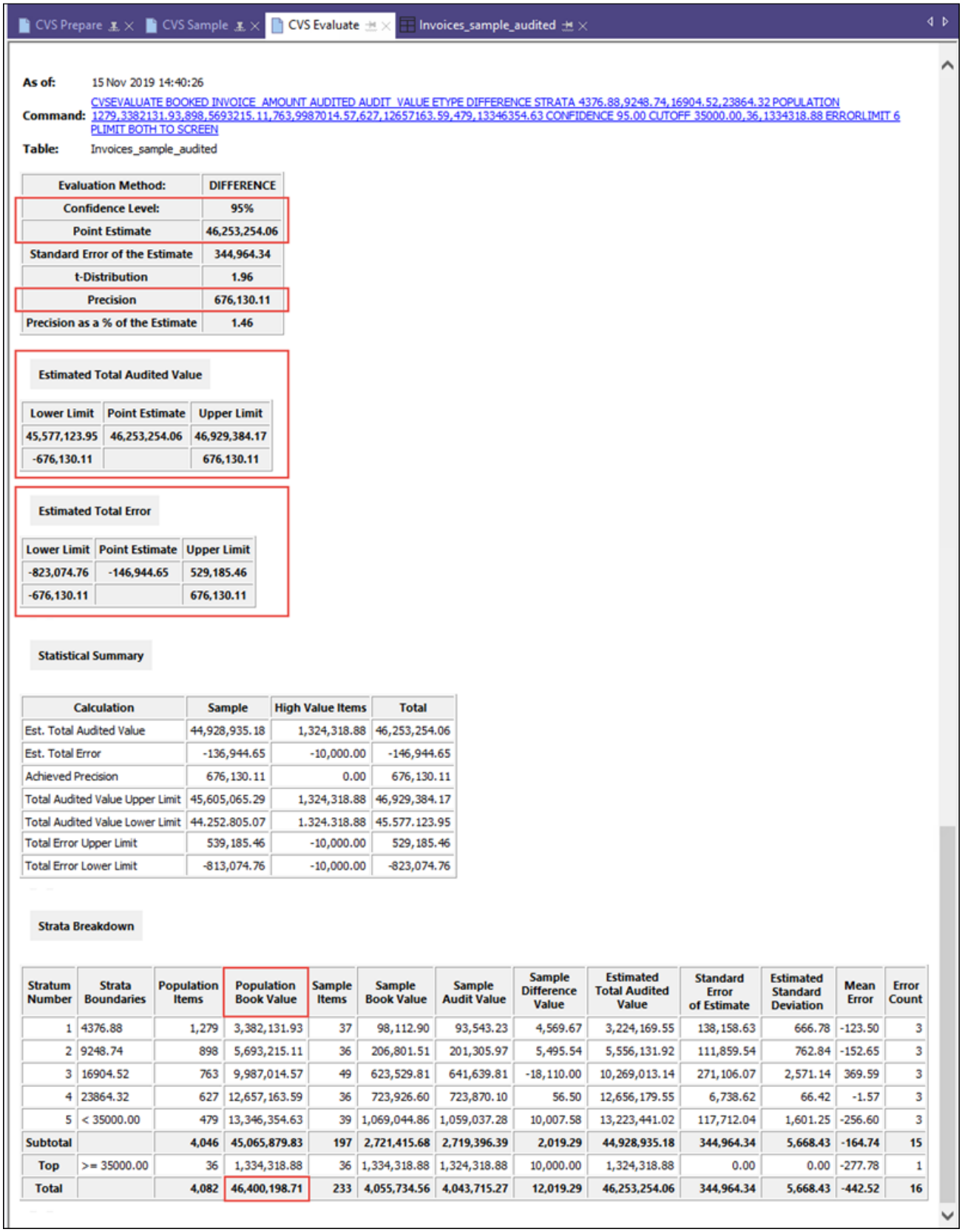


Input values – CVS Evaluate dialog box	Description
Estimation Type	The estimation type (evaluation method) to use.
Confidence Level (%)	Your desired confidence level that the resulting sample is representative of the entire population. For example, entering 95 means that you want to be confident that 95% of the time the sample will in fact be representative. Confidence is the complement of “sampling risk”. A 95% confidence level is the same as a 5% sampling risk.

Input values – CVS Evaluate dialog box	Description
Number of Expected Errors	The minimum number of errors you expected in the sample. This value is not used in the CVS Evaluate calculation. Instead, it is used to trigger a notification if the actual number of errors you found in the sample is less than the **Number of Expected Errors**. If actual errors are fewer than **Number of Expected Errors**, the only evaluation method available is mean-per-unit.
Book Value	The numeric field in the sample table containing the recorded book values.
Audit Value	The numeric field in the sample table containing the audit values.
Precision Limits	The type of precision limit to use. For more information, see "Preparing a classical variables sample" on page 1058.
Top certainty stratum cutoff (Cutoff, Count, Value)	The top certainty stratum cutoff value that was used in the CVS process, the number of records in the top certainty stratum, and their total value.
Bottom certainty stratum cutoff (Cutoff, Count, Value)	The bottom certainty stratum cutoff value that was used in the CVS process, the number of records in the bottom certainty stratum, and their total value.
Strata Boundaries	The boundary values that were used for stratifying the data set.
Population (Count, Value)	The number of records in each source table stratum, and the total value for each stratum.

Output results

Example screen from classical variables sampling tutorial



CVS Prepare | CVS Sample | CVS Evaluate | Invoices_sample_audited

As of: 15 Nov 2019 14:40:26

Command: CVSEVALUATE BOOKED INVOICE_AMOUNT AUDITED AUDIT_VALUE ETYPE DIFFERENCE STRATA 4376.88,9248.74,16904.52,23864.32 POPULATION 1279,3382131.93,898,5693215.11,763,9987014.57,627,12657163.59,479,13346354.63 CONFIDENCE 95.00 CUTOFF 35000.00,36,1334318.88 ERRORLIMIT 6 PLIMIT BOTH TO SCREEN

Table: Invoices_sample_audited

Evaluation Method:	**DIFFERENCE**
Confidence Level:	**95%**
Point Estimate	**46,253,254.06**
Standard Error of the Estimate	**344,964.34**
t-Distribution	**1.96**
Precision	**676,130.11**
Precision as a % of the Estimate	**1.46**

Estimated Total Audited Value

Lower Limit	Point Estimate	Upper Limit
45,577,123.95	46,253,254.06	46,929,384.17
-676,130.11		676,130.11

Estimated Total Error

Lower Limit	Point Estimate	Upper Limit
-823,074.76	-146,944.65	529,185.46
-676,130.11		676,130.11

Statistical Summary

Calculation	Sample	High Value Items	Total
Est. Total Audited Value	44,928,935.18	1,324,318.88	46,253,254.06
Est. Total Error	-136,944.65	-10,000.00	-146,944.65
Achieved Precision	676,130.11	0.00	676,130.11
Total Audited Value Upper Limit	45,605,065.29	1,324,318.88	46,929,384.17
Total Audited Value Lower Limit	44.252.805.07	1.324.318.88	45.577.123.95
Total Error Upper Limit	539,185.46	-10,000.00	529,185.46
Total Error Lower Limit	-813,074.76	-10,000.00	-823,074.76

Strata Breakdown

Stratum Number	Strata Boundaries	Population Items	Population Book Value	Sample Items	Sample Book Value	Sample Audit Value	Sample Difference Value	Estimated Total Audited Value	Standard Error of Estimate	Estimated Standard Deviation	Mean Error	Error Count
1	4376.88	1,279	3,382,131.93	37	98,112.90	93,543.23	4,569.67	3,224,169.55	138,158.63	666.78	-123.50	3
2	9248.74	898	5,693,215.11	36	206,801.51	201,305.97	5,495.54	5,556,131.92	111,859.54	762.84	-152.65	3
3	16904.52	763	9,987,014.57	49	623,529.81	641,639.81	-18,110.00	10,269,013.14	271,106.07	2,571.14	369.59	3
4	23864.32	627	12,657,163.59	36	723,926.60	723,870.10	56.50	12,656,179.55	6,738.62	66.42	-1.57	3
5	< 35000.00	479	13,346,354.63	39	1,069,044.86	1,059,037.28	10,007.58	13,223,441.02	117,712.04	1,601.25	-256.60	3
Subtotal		**4,046**	**45,065,879.83**	**197**	**2,721,415.68**	**2,719,396.39**	**2,019.29**	**44,928,935.18**	**344,964.34**	**5,668.43**	**-164.74**	**15**
Top	>= 35000.00	36	1,334,318.88	36	1,334,318.88	1,324,318.88	10,000.00	1,324,318.88	0.00	0.00	-277.78	1
Total		**4,082**	**46,400,198.71**	**233**	**4,055,734.56**	**4,043,715.27**	**12,019.29**	**46,253,254.06**	**344,964.34**	**5,668.43**	**-442.52**	**16**

Output results - CVS Evaluate	Description
Evaluation Method	The estimation type you selected.
Confidence Level	The confidence level that you specified as an input.
Point Estimate	A statistical projection of the most likely audited value of the entire data set in the source table. The **Point Estimate** is the midpoint of an estimated range.
Precision	A statistical projection of the amount by which the **Point Estimate** could vary. The **Point Estimate** plus or minus the **Precision** forms the upper and lower limits of the range.
Estimated Total Audited Value	A visual presentation of the **Estimated Total Audited Value** range. **How the range works** ◦ The **Population Book Value** of the source table is within the range: *The account is very likely to be fairly stated.* ◦ The **Population Book Value** of the source table is outside either the upper limit or the lower limit of the range: *The account could be materially misstated.*
Estimated Total Error	A visual presentation of the **Estimated Total Error** range. **How the Error range is calculated** ◦ The **Point Estimate** of total error is **Estimated Total Audited Value** minus **Population Book Value**. ◦ The **Point Estimate** plus or minus the **Precision** forms the upper and lower limits of the range. ◦ A negative error amount indicates overstatement error, and a positive error amount indicates understatement error. **How the range works** ◦ The **Estimated Total Error** range is within the range of zero (0) plus or minus the **Monetary Precision** you specified during the CVS Prepare stage: *The account is very likely to be fairly stated.* ◦ Either limit of the **Estimated Total Error** range is outside either limit of the **Monetary Precision** range: *The account could be materially misstated.*

Conditional sampling

> **Caution**
>
> Applying command filtering or scope parameters when sampling compromises the validity of the sample. If you do this, a note stating that the sample results may be invalid is generated in the log.
>
> Although the capability to apply command filters and scope parameters exists in the **Sampling** dialog box, the steps have been removed from the sampling procedures in this guide.

Conditional sampling is used to restrict sample selection to records that meet a specified condition – for example, transactions originating at a particular location, or products made by a particular manufacturer.

When you perform conditional sampling, you must ensure that you are working with an accurate data set. Using a command filter to refine data while sampling can yield unexpected results. A best practice is to first extract the data that meets the desired condition to a new table, and then perform the sampling, without using filters, on the new table.

Sampling filtered data versus filtering sampled data

When performing conditional sampling, be aware of the difference between:

- sampling filtered data
- filtering sampled data

Best practice: sampling filtered data

You have a table with 1000 records of which 150 meet the condition “Dept 03”. You want to draw a sample of 10 records from “Dept 03”.

The best way to achieve your goal is to filter and extract the “Dept 03” records to a new table first, before drawing the sample. You then sample the new table, so that you are drawing only from “Dept 03” records. Using this method, you are sampling filtered data.

Avoid: filtering sampled data

You have a table with 1000 records of which 150 meet the condition “Dept 03”. You want to draw a sample of 10 records from “Dept 03”.

If you draw the sample of 10 records from the original table with 1000 records, and in the process apply the command filter `IF Dept = "03"`, you are filtering sampled data.

The problem with this method is that Analytics selects 10 records from the unfiltered data set and then presents only the records that match “Dept 03”, which results in fewer than the required 10 records in the sample. The sample is not representative and is invalid.

For similar reasons, filtering an output table containing sampled records invalidates the sample.

This page intentionally left blank

Analyzing data

"Analyzing data" is a broad concept that encompasses a large range of different processes and techniques. There may be more than one way to achieve the same data analysis objective. The overall process is often iterative, requiring that you modify your initial approach based upon information you discover along the way.

Effective data analysis

At its most basic, analyzing data is the process of finding answers to questions about data. Analytics provides a number of commands and other tools that you can use to gain general insights about the data you are investigating, and to answer specific questions. However, you should resist the notion that you can click two or three buttons in Analytics and magically have all your data analysis answers.

Effective data analysis requires:

- understanding the nature of the data
- formulating specific analysis objectives
- knowledgeably applying the tools

Analytics can significantly amplify your data analysis abilities but it does not replace them.

Data analysis commands and tools in Analytics

The table below categorizes Analytics commands and tools by data analysis area. The categories are not intended to be absolute. You may find an effective use for a command outside its category. Some commands, such as Sort and Join, serve a main purpose that is not primarily analytical, but in some situations they can provide analytical insight.

> **Note**
> Data analysis, beyond the most basic, typically requires using a series of commands to progressively work toward your analysis objective, rather than using a single command in isolation.

Data analysis area	Command or tool	Description
General characteristics	Verify Count Total	Use these commands to discover general characteristics of a data set, including: ◦ data validity ◦ record counts

Data analysis area	Command or tool	Description
	Profile Statistics Outliers Sort Index	◦ total amounts ◦ minimum, maximum, and average amounts ◦ standard deviation, median, mode, and quartile values ◦ outliers ◦ ranges ◦ distribution of negative and positive values ◦ patterns
Reliability/Accuracy	computed fields	Use computed fields to recalculate and test the accuracy of calculated amounts in a data set, such as total amounts including tax
Isolation	filtering searching	Use filtering to restrict a data set or data processing to a subset of records of interest Use searching to locate specific values in a data set
Sequential Order	Sequence	Test whether data is sequentially ordered, and identify out-of-sequence items
Completeness	Gaps	Verify whether all records in a sequence, such as a sequentially ordered series of checks, are present, and identify the location of any gaps in the sequence
Uniqueness	Duplicates	Identify duplicate values or items in a field, or entire duplicate records
Inexactness	Fuzzy Duplicates	Identify nearly identical values that may refer to the same real-world entity
Frequency Distribution Concentration of Materiality	Stratify Age Classify Summarize Cross-Tabulate Histogram Cluster	Group records and determine how many records and how much value are concentrated by numeric range or cluster, by time period, or by record identifiers such as location codes, vendor/customer numbers, or product identifiers Also useful for identifying outliers
Compare	Join Fuzzy Join Relate	Combine tables to discover whether records are matched or unmatched across tables, such as an invoice table and a PO table
Numeric anomaly	Benford	Discover anomalous numeric data by testing leading digits for variance from expected Benford distribution

Profiling data

You can profile data to display the following summary statistics on one or more numeric fields in a table:

- **Total value** – The sum of all field values.
- **Absolute value** – The sum of all field values while disregarding the sign of the numbers. The absolute value is used as the default population parameter when sampling data using monetary unit sampling.
- **Minimum value** – The smallest field value.
- **Maximum value** – The largest field value.

> **Note**
> The minimum and maximum values are often used as default parameters when stratifying data and generating histograms.

Analytics includes an automatic profiling option which, if enabled, automatically produces a profile of all the numeric fields in a table each time a table is opened.

Steps

You can profile data to generate summary statistics on one or more numeric fields in a table.

Show me how

1. Select **Analyze > Profile**.
2. On the **Main** tab, do one of the following:
   - Select the field(s) to profile from the **Profile Fields** list.
   - Click **Profile Fields** to select the field(s), or to create an expression.

     The order in which you select the fields is the order in which the columns appear in the results.
3. If there are records in the current view that you want to exclude from processing, enter a condition in the **If** text box, or click **If** to create an IF statement using the **Expression Builder**.

   > **Note**
   > The **If** condition is evaluated against only the records remaining in a table after any scope options have been applied (**First**, **Next**, **While**).

   The IF statement considers all records in the view and filters out those that do not meet the specified condition.
4. Click the **More** tab.

5. Select the appropriate option in the **Scope** panel:
   - **All**
   - **First**
   - **Next**
   - **While**

   Show me more

All	This option is selected by default. Leave it selected to specify that all records in the view are processed.
First	Select this option and enter a number in the text box to start processing at the first record in the view and include only the specified number of records.
Next	Select this option and enter a number in the text box to start processing at the currently selected record in the view and include only the specified number of records. The actual record number in the leftmost column must be selected, not data in the row.
While	Select this option to use a WHILE statement to limit the processing of records in the view based on a particular criterion or set of criteria. You can enter a condition in the **While** text box, or click **While** to create a WHILE statement using the **Expression Builder**. A WHILE statement allows records in the view to be processed only while the specified condition evaluates to true. As soon as the condition evaluates to false, the processing terminates, and no further records are considered. You can use the **While** option in conjunction with the **All**, **First**, or **Next** options. Record processing stops as soon as one limit is reached.

> **Note**
> The number of records specified in the **First** or **Next** options references either the physical or the indexed order of records in a table, and disregards any filtering or quick sorting applied to the view. However, results of analytical operations respect any filtering.
> If a view is quick sorted, **Next** behaves like **First**.

6. Click **OK**.

Enable automatic profiling

You can configure Analytics to automatically profile tables when they are opened. Profile results are opened in a separate tab in the display area. If this option is enabled, it applies to all Analytics tables.

Show me how

1. Select **Tools > Options.**
2. Click the **Table** tab.
3. Select **Automatically Profile on Open**.
4. Click **OK**.

Generating statistics

You can generate detailed statistics on numeric and datetime fields in a table. Statistics provide an overview of a table, and can highlight abnormalities in the data, which can guide your subsequent analysis.

When you generate statistics, in addition to the standard output options, Analytics automatically creates a number of system variables that contain the output results. For more information, see "Variables created by Analytics commands" on page 1366.

The results of generating statistics are described in the table below.

> **Note**
> All the statistics are generated for numeric fields. Only a subset of the statistics are generated for datetime fields.

Statistic name	Results
Range	Numeric field: ◦ The difference between the highest and lowest values Datetime field: ◦ The number of days between the oldest and most recent date
Positive	◦ The number of positive values ◦ The total of all positive values (not meaningful for datetime fields) ◦ The average positive value
Negative	◦ The number of negative values ◦ The total of all negative values ◦ The average negative value
Zeros	The number of zero values
Totals	◦ The total number of positive, negative, and zero values ◦ The total of all positive, negative, and zero values ◦ The average of all positive, negative, and zero values
Abs Value	The total of all values while disregarding the sign of the numbers
Std Dev (optional)	The standard deviation from the mean value
Median (optional)	The median value ◦ Odd-numbered sets of values: the middle value ◦ Even-numbered sets of values: the average of the two values at the middle
Q25 (optional)	The first quartile value (lower quartile value)

Statistic name	Results
	◦ The result is an interpolated value based on an Analytics algorithm ◦ Produces the same result as the QUARTILE and QUARTILE.INC functions in Microsoft Excel
Q75 (optional)	The third quartile value (upper quartile value) ◦ The result is an interpolated value based on an Analytics algorithm ◦ Produces the same result as the QUARTILE and QUARTILE.INC functions in Microsoft Excel
Mode (optional)	The most frequently occurring value ◦ Displays "N/A" if no value occurs more than once ◦ In the event of a tie, displays the lowest value
Highest	The five highest values
Lowest	The five lowest values

> **Tip**
> You can use the **# of High/Low** setting on the **More** tab in the **Statistics** dialog box to specify the number of high and low values that are included in the results.

Steps

You can generate descriptive statistics on numeric and datetime fields in a table.

Show me how

1. Select **Analyze > Statistics.**
2. On the **Main** tab, do one of the following:
   - Select the field(s) to generate statistics on from the **Statistics On** list.
   - Click **Statistics On** to select the field(s), or to create an expression.

     The order in which you select the fields is the order in which the columns appear in the results.
3. If there are records in the current view that you want to exclude from processing, enter a condition in the **If** text box, or click **If** to create an IF statement using the **Expression Builder**.

   > **Note**
   > The **If** condition is evaluated against only the records remaining in a table after any scope options have been applied (**First**, **Next**, **While**).

   The IF statement considers all records in the view and filters out those that do not meet the specified condition.
4. If you want to calculate the standard deviation for the selected field or fields, select **Std Deviation**.

5. If you want to calculate the median, mode, and first and third quartile values for the selected field or fields, select **Median, Mode, Q25, Q75**.

> **Note**
> Calculating these additional statistics requires additional computer memory. You may exceed your computer's memory and get an error message if you calculate the additional statistics for very large data sets.

6. Click the **Output** tab.
7. Select the appropriate output option in the **To** panel:
   - **Screen** – Select this option to display the results in the Analytics display area.

> **Tip**
> You can click any linked result value in the display area to drill down to the associated record or records in the source table.

   If the output table contains a large number of records, it is faster and more useful to save the results to a file than to display the results on the screen.

   - **Print** – Select this option to send the results to the default printer.
   - **Graph** – Select this option to create a graph of the results and display it in the Analytics display area.
   - **File** – Select this option to save or append the results to a text file. The file is saved outside Analytics.

> **Note**
> Output options that do not apply to a particular analytical operation are disabled.

8. If you selected **File** as the output type, specify the following information in the **As** panel:
   - **File Type** – **ASCII Text File** or **Unicode Text file** (depending on which edition of Analytics you are using) is the only option. Saves the results to a new text file, or appends the results to an existing text file.
   - **Name** – Enter a file name in the **Name** text box. Or click **Name** and enter the file name, or select an existing file in the **Save** or **Save File As** dialog box to overwrite or append to the file. If Analytics prefills a file name, you can accept the prefilled name, or change it.

     You can also specify an absolute or relative file path, or navigate to a different folder, to save or append the file in a location other than the project location. For example: `C:\Results\Output.txt` or `Results\Output.txt`.

   - **Local** – Disabled and selected. Saving the file locally is the only option.
9. Click the **More** tab.
10. Select the appropriate option in the **Scope** panel:

    - **All**
    - **First**
    - **Next**
    - **While**

Show me more

All	This option is selected by default. Leave it selected to specify that all records in the view are processed.
First	Select this option and enter a number in the text box to start processing at the first record in the view and include only the specified number of records.
Next	Select this option and enter a number in the text box to start processing at the currently selected record in the view and include only the specified number of records. The actual record number in the leftmost column must be selected, not data in the row.
While	Select this option to use a WHILE statement to limit the processing of records in the view based on a particular criterion or set of criteria. You can enter a condition in the **While** text box, or click **While** to create a WHILE statement using the **Expression Builder**. A WHILE statement allows records in the view to be processed only while the specified condition evaluates to true. As soon as the condition evaluates to false, the processing terminates, and no further records are considered. You can use the **While** option in conjunction with the **All**, **First**, or **Next** options. Record processing stops as soon as one limit is reached.

> **Note**
> The number of records specified in the **First** or **Next** options references either the physical or the indexed order of records in a table, and disregards any filtering or quick sorting applied to the view. However, results of analytical operations respect any filtering.
> If a view is quick sorted, **Next** behaves like **First**.

11. If you want to change the number of highest and lowest field values included in the results, enter the number in **# of High/Low**.
12. If you selected **File** as the output type, and want to append the output results to the end of an existing text file, select **Append To Existing File**.
13. Click **OK**.
14. If the overwrite prompt appears, select the appropriate option.

Identifying outliers

Use the outlier feature in Analytics to identify records that are out of the ordinary and could require closer scrutiny.

What are outliers?

Outliers are records with numeric amounts that differ significantly from the numeric amounts in the records that they are grouped with.

Example of an outlier in a group

In an accounts payable file, invoices from a particular company typically range between $500 and $1,000. However, one invoice is for $8,500.

> **Note**
> A record can be an outlier for a legitimate reason. Typically, you need to perform additional examination of the outliers identified by Analytics to determine if any issues actually exist.

Grouping records is optional

When examining data for outliers, you do not have to group the records. You may be interested in finding outliers across an entire table, rather than within specific groups.

Example of outliers in an entire set of records

In an accounts payable file, the entire set of invoices typically ranges between $40 and $5,000. However, three invoices are greater than $20,000.

How are outliers identified?

For each group of records, or for an entire set of records, Analytics uses the standard deviation of a specified numeric field, or a multiple of the standard deviation, to establish upper and lower outlier boundaries.

Any record with a value in the numeric field that is greater than the upper boundary, or less than the lower boundary, is an outlier and included in the output results.

Standard deviation is a measure of the dispersion of a data set – that is, how spread out the values are. The outliers calculation uses population standard deviation.

Identifying outliers for a set of numbers

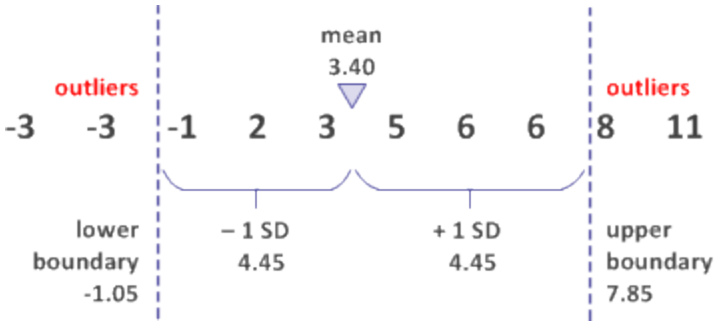
You want to identify any outliers in the following set of numbers:

-3, -3, -1, 2, 3, 5, 6, 6, 8, 11

The mean (average) of the numbers is 3.40. The mean is used to calculate the standard deviation (SD) of the set: 4.45.

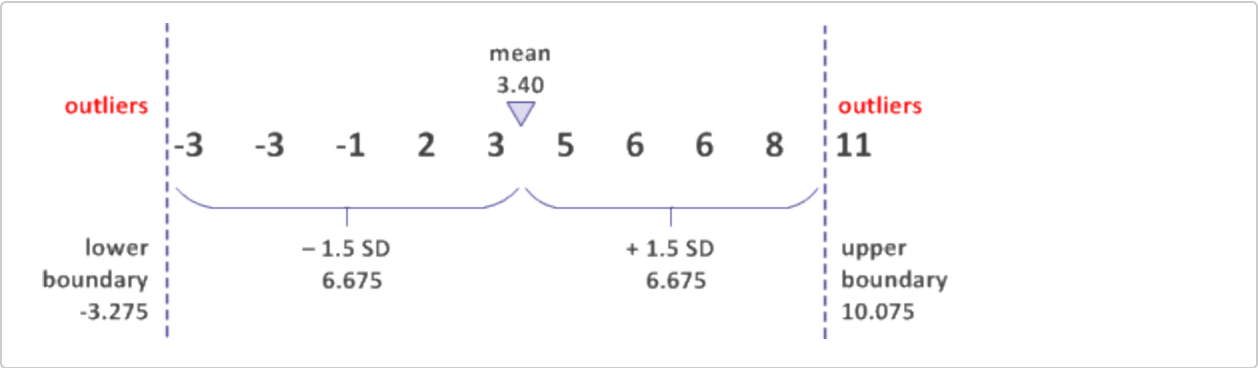
The mean ± 1 standard deviation

In the first example, you use the mean ± 1 standard deviation to establish the upper and lower outlier boundaries. Four values are identified as outliers.



The mean ± 1.5 standard deviations

In the second example, you use the mean ± 1.5 standard deviations to establish the upper and lower outlier boundaries. Now only one value is identified as an outlier.



Positioning the outlier boundaries

You can position the outlier boundaries wherever you feel is appropriate, or you can test different positions and compare results.

To position the boundaries, you specify any positive multiple of the standard deviation of the outlier field: 0.5, 1, 1.5, and so on. For example, if you specify a multiple of 1.5, the outlier boundaries are 1.5 standard deviations above and below the mean or median of the values in the outlier field.

For the same set of data, as you increase the standard deviation multiple, you potentially decrease the number of outliers in the output results.

The distribution of data

The values in a set of numeric data are typically distributed over a range from smallest to largest. In a normal distribution, the values are evenly distributed around the center point of the data, forming a bell-shaped curve. The center point is often defined as the average or the mean of the values, but it could also be the median or the mode.

Show me more

Standard deviation of a normal distribution

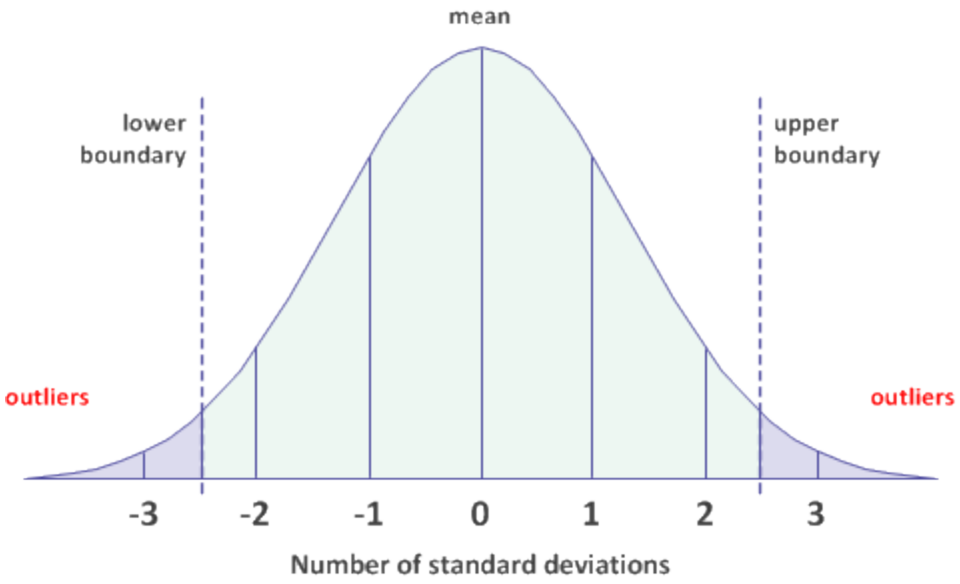
If you calculate the standard deviation for a set of normally distributed values, 68% of the values fall within one standard deviation of the mean (±), and 99.7% of the values fall within three standard deviations of the mean (±). Only a very few values exceed three standard deviations from the mean.

The distribution of values in the data sets that you analyze in Analytics may often be skewed rather than normally distributed. For example, a transaction file may have thousands of relatively small transactions, and a few large transactions. However, we can use a normal distribution for a simple illustration of how outlier boundaries work in Analytics.

As the examples below show, increasing the standard deviation multiple moves the upper and lower outlier boundaries closer to the tails of the distribution curve. As the boundaries move closer to the tails, progressively fewer values occur outside the boundaries.

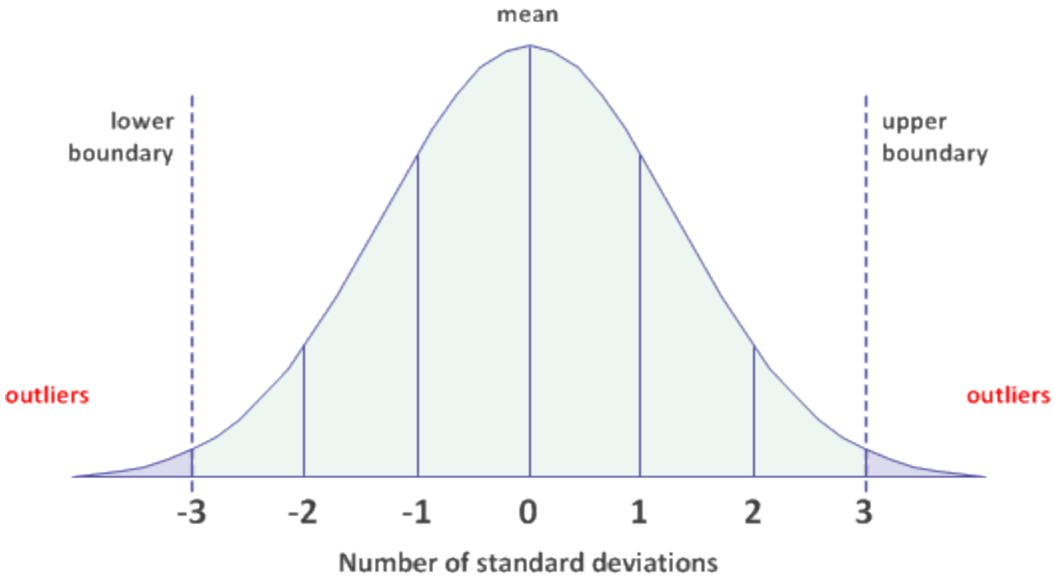
Outlier boundaries ±2.5 standard deviations from the mean

Values that are greater than +2.5 standard deviations from the mean, or less than -2.5 standard deviations, are included as outliers in the output results.



Outlier boundaries ±3 standard deviations from the mean

Values that are greater than +3 standard deviations from the mean, or less than -3 standard deviations, are included as outliers in the output results.



Guidelines

When you specify settings in the outliers feature, consider the nature of the data you are analyzing:

Nature of the data	Setting guideline
Values are clustered, with a small range	Use a smaller standard deviation multiple. Try starting with 1. Use decimal multiples such as 1.25, to make precise adjustments.
Values are dispersed, with a large range	Use a larger standard deviation multiple. Try starting with 3.
The data is skewed, with a small percentage of the values being large, or small, when compared to the rest of the data	Use **Median**, instead of **Average**, as the method for calculating the center point of the values that you are examining.

Adjusting based on the output results

- **Too many results** – increase the standard deviation multiple
- **Too few results, or no results** – reduce the standard deviation multiple

Keep in mind that you can use decimal multiples, and multiples less than 1. For example: 0.75.

Steps

1. Open the table that you want to test for outliers.
2. From the Analytics main menu, select **Analyze > Outliers**.
3. Under **Method**, select the method for calculating the center point of the values in the numeric field that you are examining:
   - **Average**
   - **Median**
4. In **Number of times of S.dev**, specify a multiple of the standard deviation to use for the outlier boundaries.

   You can specify any positive integer or decimal numeral (0.5, 1, 1.5, 2 . . .).
5. Do one of the following:
   - From the **Primary Keys** list, select one or more key fields to use for grouping the records in the table.

     > **Tip**
     > You can **Ctrl+click** to select multiple non-adjacent fields, and **Shift+click** to select multiple adjacent fields.

   - Select **No Key** to identify outliers across the entire table, rather than within specific groups.

6. From the **On Field** list, select the numeric field to examine for outliers ("the outlier field").
7. Optional. From the **Other Fields** list, select one or more additional fields to include in the output table.

> **Note**
> Key fields and the outlier field are automatically included in the output table, and do not need to be selected.

8. If there are records in the current view that you want to exclude from processing, enter a condition in the **If** text box, or click **If** to create an IF statement using the **Expression Builder**.

> **Note**
> The **If** condition is evaluated against only the records remaining in a table after any scope options have been applied (**First**, **Next**, **While**).

The IF statement considers all records in the view and filters out those that do not meet the specified condition.

9. Do one of the following:
   a. In the **To** text box, specify the name of the output table.
   b. Select **Screen** to output the results to the Analytics display area.
10. Deselect **Presort**, if appropriate.

> **Note**
> Guidance is provided below.

11. On the **More** tab:
    a. Optional. To specify that only a subset of records are processed, select one of the options in the **Scope** panel.
    b. Optional. Select **Use Output Table** if you want the output table to open automatically.
    c. Click **OK**.

Outliers dialog box options

The tables below provide detailed information about the options in the **Outliers** dialog box.

Main tab

Options - Outliers dialog box	Description
Average Median	The method used for calculating the center point of the values in the outlier field. ◦ **Average** - use the average (mean) of the values in the field ◦ **Median** - use the median of the values in the field

Options - Outliers dialog box	Description
	The center point is used in calculating the standard deviation of the values in the outlier field. **Note** If you select **Median**, the outlier field must be sorted. Select **Presort** if the outlier field is not already sorted. **Tip** If the data you are examining for outliers is significantly skewed, **Median** might produce results that are more representative of the bulk of the data.
Number of times of S.dev	In the outlier field, the number of standard deviations from the mean or the median to the upper and lower outlier boundaries. You can specify any positive integer or decimal numeral (0.5, 1, 1.5, 2 . . .) For example, specifying `2` establishes, for each key field group, or for the field as a whole: ◦ an upper outlier boundary 2 standard deviations greater than the mean or the median ◦ a lower outlier boundary 2 standard deviations less than the mean or the median Any value in the outlier field greater than an upper boundary, or less than a lower boundary, is included as an outlier in the output results. **Note** For the same set of data, as you increase the number of standard deviations, you potentially decrease the number of outliers in the output results.
Primary Keys optional	The field or fields to use for grouping the data in the table. For each key field group, a standard deviation is calculated for the group's numeric values in the outlier field. The group standard deviation is used as the basis for identifying group outliers. Key fields can be character, numeric, or datetime. Multiple fields can be any combination of data types. If you select more than one field, you created nested groups. The nesting follows the order in which you select the fields. **Note** The key field or fields must be sorted. Use **Presort** if one or more fields are not already sorted.
No Key optional	Do not group the data in the table. A standard deviation is calculated for the outlier field as a whole. The field standard deviation is used as the basis for identifying field outliers.
On Field ("the outlier field")	The numeric field to examine for outliers. You can examine only one field at a time. If you select a key field, outliers are identified at the group level. If you select **No Key**, outliers are identified at the field level.

<table>
<tr><th>Options - Outliers dialog box</th><th>Description</th></tr>
<tr><td>Other Fields
optional</td><td>One or more additional fields to include in the output.

Note
Key fields and the outlier field are automatically included in the output table, and do not need to be selected.</td></tr>
<tr><td>If
optional</td><td>Allows you to create a condition to exclude records from processing.
You can enter a condition in the If text box, or click If to create an IF statement using the Expression Builder.</td></tr>
<tr><td>To
optional</td><td>Specifies the name and location of the output table.
◦ To save the output table to the Analytics project folder - enter only the table name.
◦ To save the output table in a location other than the project folder - specify an absolute or relative file path, or click To and navigate to a different folder.
For example: <code>C:\Results\Output.fil</code> or <code>Results\Output.fil</code>.
Regardless of where you save the output table, it is added to the open project if it is not already in the project.
If Analytics prefills a table name, you can accept the prefilled name, or change it.</td></tr>
<tr><td>Screen
optional</td><td>Displays the results in the Analytics display area instead of creating an output table.</td></tr>
<tr><td>Presort
optional</td><td>Performs a sorting operation before executing the command.
<table><tr><th>If you select Presort and:</th><th>Sorts by:</th></tr><tr><td>◦ One or more key fields
◦ Average</td><td>◦ key field or fields
◦ key field or fields, then by the outlier field (if the outlier field is computed)

Note
Sorting a computed outlier field is an internal, technical requirement of Analytics.</td></tr><tr><td>◦ One or more key fields
◦ Median</td><td>key field or fields, then by the outlier field</td></tr><tr><td>◦ No Key
◦ Average</td><td>no sorting</td></tr><tr><td>◦ No Key
◦ Median</td><td>the outlier field</td></tr></table>
Tip
If the appropriate field or fields in the input table are already sorted, you can save processing time by not selecting Presort.</td></tr>
</table>

More tab

Options - Outliers dialog box	Description
Scope panel	Specifies which records are processed: ◦ **All** - (default) all records in the table are processed. ◦ **First** - select this option and enter a number in the text box to start processing at the first record in the table and include only the specified number of records. ◦ **Next** - select this option and enter a number in the text box to start processing at the currently selected record in the table view and include only the specified number of records. The actual record number in the leftmost column must be selected, not data in the row. ◦ **While** - select this option to use a WHILE statement to limit the processing of records in the table based on criteria. • You can enter a condition in the **While** text box, or click **While** to create a WHILE statement using the **Expression Builder**. • A WHILE statement allows records to be processed only while the specified condition evaluates to true. • You can use the **While** option in conjunction with the **All**, **First**, or **Next** options. **Note** The number of records specified in the **First** or **Next** options references either the physical or the indexed order of records in a table, and disregards any filtering or quick sorting applied to the view. However, results of analytical operations respect any filtering. If a view is quick sorted, **Next** behaves like **First**.
Use Output Table	Specifies whether the Analytics table containing the output results opens automatically upon completion of the operation.
OK	Executes the operation. If the overwrite prompt appears, select the appropriate option.

Sorting, filtering, and searching

Sorting, filtering, and searching are three of the most basic operations that you can perform with data in Analytics. You might need to perform one or more of these operations as preparation for your main analytical tests, or they could represent useful analysis in their own right.

With sorting and filtering, you also have the option of performing these operations as an integral part of the execution of other Analytics commands. For example, to summarize only third quarter transactions in an annual table, you could incorporate a date filter in the summarize command.

Different approaches you can take

The table below outlines the different approaches you can take with sorting, filtering, and searching.

Operation	Approaches
Sorting	◦ Quick sort - sort the values in a column to temporarily reorder the records in a view ◦ Sort command - sort records and output them to a new, physically reordered Analytics table ◦ Index command - index records to temporarily sort them in the current table ◦ Presort - sort records as an integral part of the execution of an Analytics command
Filtering	◦ Quick filter - filter data by using the mouse in a view ◦ Global filter - restrict which records in a view are displayed, or processed by Analytics operations ◦ Local filter - restrict which records are processed during a single execution of a single Analytics operation
Searching	◦ Quick search - find a word, a phrase, a number, or a date in a table ◦ Isolate all matching records - perform simple or advanced searches to include only records that match the search criteria ◦ Select the first matching record - select the first record in a table that matches the search criteria

Quick sorting data in a view

You can quick sort a column in a view to reorder the records in either ascending or descending order based on the values in the selected column.

When you quick sort a column, you create a temporary index for the view, based on the selected column, without going through the process of creating an index in the **Index** dialog box.

> **Note**
> Quick sorting works with a maximum field length of 247 characters. If a field is longer than 247 characters, the **Quick Sort** menu options are disabled.

Steps

1. Right-click the header of the column you want to use for sorting.
2. Select **Quick Sort Ascending** or **Quick Sort Descending** to sort the column in either ascending or descending order.
3. If you want to remove the quick sort, right-click the column and select **Quick Sort Off**.

   The quick sort is automatically removed when you close the table.

Quick filtering data in a view

Quick filters allow you to quickly and easily filter data by using the mouse to select values in a view. You can select a single value, or multiple adjacent values. You can use quick filters with any data type.

Quick filters auto-generate the syntax of the filter expression, which you can then modify as an easy way to create different or more complex filters.

Quick filters are easier to create than filters you create with the **Expression Builder**, or enter manually, but they are also more limited. Like other types of filters, you can name and save quick filters for subsequent reuse.

Quick filters are global filters

The quick filter you create is a **global filter**. Global filters restrict which records in a view are displayed, or processed by Analytics operations. For more information, see "Global filters (view filters)" on page 1151.

Quick filtering using non-adjacent values

You cannot use non-adjacent values in the initial application of a quick filter. Depending on the complexity of the filter, you may be able to rearrange fields in the view to make values adjacent.

You can use non-adjacent values subsequently if you apply a second quick filter to the subset of data created by the first quick filter.

If you need to use non-adjacent values in a single filter that operates on an unfiltered data set, and the values cannot be made adjacent by rearranging fields, you must enter the filter expression manually, or create it using the **Expression Builder**.

Quick filtering by blank or non-blank values

Two of the options for quick filtering character fields are **Blank** and **Not Blank**. To use either of these criteria, you must first select a value in the field, but the actual selected value is ignored. This behavior allows you to filter a very long column of data for blanks without having to first locate a blank value.

For information about filtering numeric or datetime fields by blank or non-blank values, see "Searching data" on page 1162.

Quick filtering by date, datetime, or time

Quick filtering by date, datetime, or time values can produce invalid results if you have specified a date or time display format that does not display all the available source data – for example, if you specify the format **hh:mm** for times that have hour, minute, and seconds data.

For more information about date and time display formats, see "Date and Time options" on page 133.

Quick filter options

The options available in the **Quick Filter** menu vary depending on the data type of the field you are filtering, and whether you select a single value, or multiple adjacent values.

Show me more

Data selected	Quick Filter menu options
a single character value	◦ Equal ◦ Not Equal ◦ Greater Than ◦ Greater Than Or Equal To ◦ Less Than ◦ Less Than Or Equal To ◦ Blank ◦ Not Blank
a single numeric value	◦ Equal ◦ Not Equal ◦ Greater Than ◦ Greater Than Or Equal To ◦ Less Than ◦ Less Than Or Equal To
a single date value	◦ On ◦ Not On ◦ After ◦ On or After ◦ Before ◦ On or Before
a single datetime or time value	◦ At ◦ Not At ◦ After ◦ At or After ◦ Before ◦ At or Before
a single logical value	◦ Equal ◦ Not Equal
multiple adjacent values in the same record	◦ Equal

Data selected	Quick Filter menu options
	Includes records in the filtered table if **all** selected values are matched. ◦ Not Equal Includes records in the filtered table if **all** selected values are **not** matched. ◦ Any Not Equal Includes records in the filtered table if **any** selected values are **not** matched.
multiple adjacent character values in the same column	◦ Equal ◦ Not Equal ◦ Between ◦ Not Between ◦ Blank ◦ Not Blank
multiple adjacent numeric values in the same column	◦ Equal ◦ Not Equal ◦ Between ◦ Not Between
multiple adjacent date values in the same column	◦ On ◦ Not On ◦ Between ◦ Not Between
multiple adjacent datetime or time values in the same column	◦ At ◦ Not At ◦ Between ◦ Not Between
a block of values spanning two or more records and two or more columns	◦ Equal (not shown)

Steps

Apply the initial quick filter

1. Select a single value, or two or more adjacent values, as the basis for the quick filter.

   Click and drag to select multiple adjacent values.

   If you want to quick filter a character field by blank or non-blank values, select any value in the field.
2. Right-click in the data area of the view, select **Quick Filter**, and select an appropriate submenu option.

The records in the table are filtered based on your selection. The auto-generated syntax of the filter expression appears in the Filter text box at the top of the View tab.

If you selected a block of values spanning two or more records and two or more columns, no submenu options are available and the quick filter automatically filters the view to include only those records that match the selected values.

Count the number of filtered records

After applying a quick filter, use the following method to count the number of records included by the filter. Use the same method after modifying or extending a quick filter.

1. From the Analytics main menu, click **Count**  .
2. Click **OK**.

   The number of records included by the filter, and the total number of records in the table, appear in the status bar at the bottom of the Analytics interface. For example: **Records: 108/772**

Modify the quick filter

If you want to modify the quick filter, manually edit the filter expression in the Filter text box and click **Set Filter**  .

Extend the quick filter

If you want to extend the quick filter by applying one or more additional quick filters, do the following:

1. Select a single value or two or more adjacent values.
2. Right-click in the data area of the view, select **Quick Filter > AND** or **OR**, and select an appropriate submenu option.

Example

To further limit the filter `Customer = '795401'` to records in which the transaction type is "IN", you can select the `IN` value and then right-click and select **Quick Filter > AND > Equal**. The filter expression is modified as follows:

```
(Customer = '795401') AND (Type = 'IN')
```

AND further limits the initial filtered set of data. Depending on the specific details of the extended filter expression, **OR** typically expands the initial filtered set of data.

Replace the current quick filter

If you want to replace the current quick filter or filters with a new quick filter, do the following:

1. Select a single value or two or more adjacent values.
2. Right-click in the data area of the view, select **Quick Filter > Replace**, and select an appropriate submenu option.

Remove the quick filter

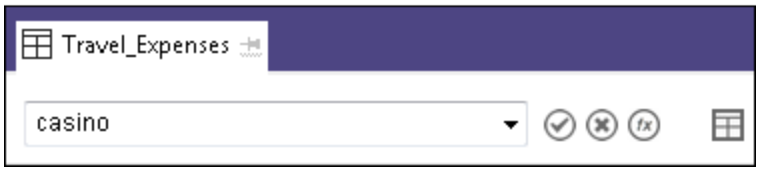
If you want to remove the quick filter or filters, click **Remove Filter**  in the filter toolbar.

Save the quick filter as a named filter

If you want to save the quick filter as a named filter associated with the table, click **Edit View Filter**  in the filter toolbar, enter a name for the filter in the **Save As** text box, and click **OK**.

Quick searching data in a table

You can enter one or more search terms in the Filter text box at the top of the View tab to perform a quick search of the data in a table.



Search scope

All the source data in the table is searched, not just the data displayed in the current view. For information about source data, tables, and views, see "The structure of Analytics tables" on page 115.

Entire records are searched, including any undefined portion of the record, rather than specific fields. For example, entering **casino** finds records that contain “casino” anywhere in the record. You can subsequently modify the search if you need to search a specific field (character data searches only).

Computed fields and related fields are not searched unless you subsequently modify the search to specify the specific field.

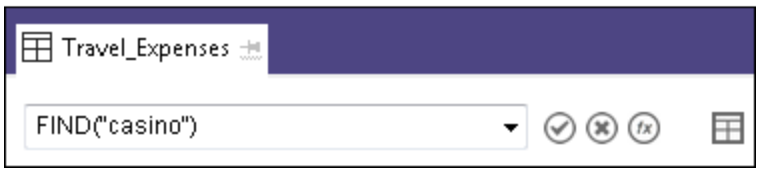
Data types searched

Searching character data is the most straightforward use of quick search. You can also search numeric and datetime data, however there are some additional considerations to take into account, explained in subsequent sections.

Automatic conversion of search term to a filter

The search term or terms you enter are automatically converted to a global filter that uses the FIND( ) function.

For example, entering **casino** results in the filter `FIND("casino")`.



The filter auto-populates the Filter text box, from where you can modify it, if required. For example, you could modify `FIND("casino")` to limit the search to a specific field: `FIND("casino", Merchant)`.

The filter is also added to the filter history and to the command log, from where you can subsequently reapply it.

Search terms and filter syntax automatically distinguished

The Filter text box automatically distinguishes between search terms and filter syntax. For example, entering `match` in the Filter text box searches for the character string “match”, whereas entering `match(City, "New York", "Washington")` creates a filter using the MATCH( ) function.

Steps

> **Note**
> When searching for numbers or datetime values, you need to match the source data formatting rather than the formatting in the view. For more information, see "Quick searching numeric or datetime data" on page 1117.

Search for one or more search terms

In the Filter text box at the top of the View tab, type one or more search terms and press Enter.

With multiple search terms, the quick search performs a logical OR operation and finds records that contain at least one of the search terms.

Search for an exact phrase

In the Filter text box at the top of the View tab, type the phrase, enclose it in double quotation marks, and press Enter.

The quick search finds only those records that contain the exact phrase.

Limit the search to a specific field (character fields only)

1. Modify the automatically generated filter in the Filter text box by typing a comma after the search term, and then add the name of the field.

   For example, modify `FIND("casino")` to `FIND("casino", Merchant)`.

2. Press Enter.

   The search is restricted to the field that you specified.

> **Note**
>
> You must use the physical name of the field, which may not be the same as the display name of the field in the view.
>
> To check the physical name, right-click the appropriate column header and select **Properties**. If necessary, copy the physical name from the text box at the top of the **Modify Column** dialog box. Do not use the **Alternate Column Title**.
>
> To search in a related field you must specify the fully qualified name of the field (that is, *table.field name*). For example: `FIND("casino", Vendor.Vendor_Name)`

Quick searching character data

When quick searching character data, you can enter whole or partial words, or exact phrases.

Show me more

- If you enter more than one word, the quick search performs a logical OR operation and finds records that contain at least one of the words.
- If you want to search for an exact phrase, enclose the phrase in double quotation marks.
- To isolate a search term, include a trailing space after the term and enclose the term and the space in double quotation marks.

  For example, `"cash "` returns “cash” but not “cashier”, assuming that in the data the string “cash” is followed by at least one space

Search terms	Return records that contain:
cas	◦ **cas**ino ◦ **cas**h ◦ Ameri**cas** ◦ Lan**cas**hire ◦ etcetera . . .
casino	◦ casino ◦ casinos
casino liquor	◦ casino ◦ casinos ◦ liquor ◦ liquors ◦ casino (and) liquor (order not considered) ◦ etcetera . . .

Search terms	Return records that contain:
“Diamond Casino”	◦ Diamond Casino
“Diamond Casino” “Golden Casino”	◦ Diamond Casino ◦ Golden Casino ◦ Diamond Casino (and) Golden Casino (order not considered)
casino, “ABC Liquors”	◦ casino ◦ casinos ◦ ABC Liquors ◦ casino (and) ABC Liquors (order not considered) ◦ etcetera . . .
“ABC L”	◦ ABC Liquors ◦ ABC Limousine ◦ ABC Learning ◦ etcetera . . .
“cash ” (the word ‘cash’ followed by one space)	◦ cash (in the data, requires that the string ‘cash’ is followed by at least one space) ◦ does not return ‘cashier’ or ‘Lancashire’

Quick searching numeric or datetime data

> **Note**
> If you want to search for numeric or datetime data in a specific field, use quick filtering. For more information, see "Quick filtering data in a view" on page 1109.

When quick searching numeric or datetime data, you need to remember that you are searching the underlying source data rather than the data displayed in a view.

Numbers, dates, and times are often formatted differently in the source data than they are in the view. **Search terms need to match the source data formatting rather than the formatting in the view.**

You can select **Edit > Table Layout** to view the source data for a table.

Quick searching numeric data

The numeric format in the source data affects which records are returned for a specific search term.

Show me more

Search term	Numeric format in view	Numeric format in source data	Returns records that contain:
1234.00	9999.99	9999.99	1234.00
		9,999.99	no records returned
1,234.00	9,999.99	9999.99	no records returned
		9,999.99	1,234.00
		9.999,99	no records returned
(1234.00)	(9999.99)	(9999.99)	(1234.00)
		-9999.99	no records returned
1234.01	9999.99 (number rounded) for example: 1234.01	9999.9999 for example: 1234.0085	no records returned
1234.0085			1234.0085
123 456	9999.99	9999.99	◦ 123 ◦ 456 ◦ 123 (and) 456 (order not considered)

Quick searching datetime data

The datetime format in the source data affects which records are returned for a specific search term.

Show me more

Search term	Datetime format in view	Datetime format in source data	Returns records that contain:
12/31/2015	MM/DD/YYYY	MM/DD/YYYY	12/31/2015
		DD/MM/YYYY	no records returned
		YYYYMMDD	no records returned
31/12/2015		MM/DD/YYYY	no records returned
		DD/MM/YYYY	31/12/2015
		YYYYMMDD	no records returned
20151231		MM/DD/YYYY	no records returned
		DD/MM/YYYY	no records returned
		YYYYMMDD	20151231
2015-12-31	YYYY-MM-DD	YYYY-MM-DD	error message no records returned
FIND("2015-12-31")			2015-12-31
23:59:59	hh:mm:ss	hh:mm:ss	23:59:59
		hhmmss	no records returned
20151231.235959	MM/DD/YYYY hh:mm:ss	YYYYMMDD.hhmmss	20151231.235959
		MM/DD/YYYY hh:mm:ss	no records returned

Additional characteristics of quick searching

Quick searching has these additional characteristics:

Characteristic	Description
Case-sensitivity	The search is not case-sensitive.
Wildcards	Wildcard characters in search terms are not supported.
Spaces	Leading, trailing, and intervening spaces in search terms are considered only if you enclose the search term or terms and the spaces inside double quotation marks. When spaces are enclosed by double quotation marks they are treated like characters and must be exactly matched in the data.

Characteristic	Description
Quotation marks	Only double quotation marks can be used for enclosing phrases. Single quotation marks are not supported for this purpose and are treated like a regular character.
Computed fields	Computed fields are not searched.
Related fields	Related fields are not searched.
Limiting search by field	When modifying the auto-populated filter to limit the search to a specific field you can specify only character fields. Specifying a numeric or datetime field causes an error.

Unsupported characters	If used in quick search terms, the following characters may give inconsistent results, or cause an error message, because they are the operators used in Analytics expressions:
	`^ * ( ) - + = < >`
	If you want to search for one of these characters, manually enter the FIND( ) function in the Filter text box. For example:
	`FIND("a+b")` or `FIND("2015-12-31")`

Field boundaries Trailing spaces	Field boundaries in records are ignored, which means it is possible for a search term to match a string of characters across a field boundary. Trailing spaces in fields are treated like characters. Search results found across field boundaries may not be valid results, unless you specifically intend this type of search. For example, the last digits of an account number and the first digits of an amount in an adjacent field could match a numeric search term, but would probably be a false positive. **Note** The field boundaries under discussion are the ones that appear in the table layout, based on the physical order of fields in the layout. The order of fields in the layout can differ from the order of columns in the associated view, creating different adjacencies in the layout and the view. If it is unclear why quick searching is returning a particular record, select **Edit > Table Layout** to view the source data being searched.

Additional searching and filtering information

- For many other data searching options, including using wildcards, see "Searching data" on page 1162.
- For more information about filtering, see "Filtering data" on page 1147.
- For more information about the FIND( ) function, see "FIND( ) function" on page 2145.

Sorting and indexing

Sorting and indexing are two different methods for sequentially ordering data in tables. Some Analytics commands require that the input is first sorted or indexed. Ordering data can also be a useful analytical operation in itself, bringing patterns and anomalies into focus.

Operation	Description
Sorting	Sorting a table physically reorders data into a sequential order and outputs the results to a new Analytics table.
Indexing	Indexing does not make any change to the underlying physical order of data. Instead, it creates a separate index file that references records in a table, allowing access to the records in a sequential order rather than a physical order. Data in a view is reordered in accordance with an index only while the index is active.

Ordering data as a prerequisite for other operations

Because computers process files in sequence, starting with the first record, sequentially ordering data is a prerequisite for several analytical tests and other operations in Analytics. Multiple-table operations, such as joins or relations, may require key fields be sorted or indexed.

Other Analytics tests and operations may not require ordered data, but they execute much more quickly if the data is first sorted or indexed.

Should I sort or index?

The decision whether to sort or index may depend on the particular task you want to perform. For example:

- **Sorting** – might be a better choice for investigative work because it outputs a new table that can serve as the basis for subsequent analysis
- **Indexing** – might be a better choice for informational or preliminary work because it allows you to quickly switch between different representations of the data in the active table

Benefits and drawbacks of sorting and indexing

The table below compares the benefits and drawbacks of sorting and indexing, and lists operations that require either sorting or indexing as a prerequisite.

	Sorting	Indexing
Outputs results to a new, physically separate Analytics table	Yes	No
Physically reorders data	Yes	No
Speed of operation	Slower	Faster
Required disk space for processing	More	Less
Resulting file size	Larger	Smaller
Subsequent processing of the sorted or indexed file	Faster	Slower
Searching character fields	Slower	Faster
Prerequisite for	◦ Join (recommended, but not enforced, for the primary table) ◦ Merge ◦ Duplicates ◦ Gaps	◦ Define Relation (indexing of the child table key field performed automatically by Analytics) ◦ Join (applying an index to the secondary table can only be performed from the command line or in a script) ◦ Merge (applying an index to the secondary table can only be performed from the command line or in a script) ◦ Duplicates ◦ Gaps ◦ Find ◦ Find Literal search option ◦ Seek ◦ Seek Expression search option

The Sort Order option and sort sequences

The **Sort Order** option (**Tools > Options > Table**) specifies the sort sequence (collation) for character data. The option you specify defines what sort sequence is used when you sort or index records, or test sequential order, using a character field.

What is a sort sequence?

A sort sequence is like a template against which Analytics compares the first character or characters of each value in a character field when sorting, indexing, testing sequential order, or performing a quick sort.

The table below shows the default Analytics **Sort Order** setting and the associated sort sequence.

Analytics Edition	Sort Order default	Associated sort sequence
non-Unicode	System Default (-ASCII-	◦ numbers, then uppercase, then lowercase 0, 1, 2... A, B, C... a, b, c... For example, “Z” sorts before “a”. ◦ Special characters occur at different points in the sequence, depending on the character. ◦ Characters with diacritical marks occur at the end of the sequence and use the same uppercase before lowercase internal sequence. Show all characters sorted (non-Unicode edition) `! " # $ % & ' ( ) * + , - . / 0 1 2 3 4 5 6 7 8 9 : ; < = > ? @ A B C D E F G H I J K L M N O P Q R S T U V W X Y Z [ \ ] ^ _ ` a b c d e f g h i j k l m n o p q r s t u v w x y z { | } ~ € ‚ ƒ „ … ˆ Š ‹ Ž ‘ ’ “ ” – — ˜ ™ š › ž Ÿ ¡ ¢ £ ¥ ¦ ¨ © « ® ± ´ ¸ » ¿ À Á Â Ã Ä Å Ç È É Ê Ë Ì Í Î Ï Ñ Ò Ó Ô Õ Ö Ø Ù Ú Û Ü Ý Þ ß à á â ã ä å ç è é ê ë ì í î ï ð ñ ò ó ô õ ö ø ù ú û ü ý þ ÿ`

Analytics Edition	Sort Order default	Associated sort sequence
	l-)	
Unicode	Mix Language (-UCA-) (-Unicode collation	◦ numbers, then lowercase and uppercase intermixed 0, 1, 2... a, A, b, B, c, C... For example, “a” sorts before “Z”. ◦ Special characters occur before numbers. ◦ Characters with diacritical marks are intermixed with characters without diacritical marks. For example: e, E, é, É, f, F Show all characters sorted (Unicode edition) _ - – — , ; : ! ¡ ? ¿ . … ' ‘ ’ , ‹ › “ ” „ « » ( ) [ ] { } @ * / \ & # % ` ´ ~ ^ ¨ ¸ ˆ © ® + ± < = > | ¦ ~ ¢ $ £ ¥ € 0 1 2 3 4 5 6 7 8 9 a A á Á à À â Â å Å ä Ä ã Ã b B c C ç Ç d D ð e E é É è È ê Ê ë Ë f F ƒ g G h H i I í Í ì Ì î Î ï Ï j J k K l L m M n N ñ Ñ o O ó Ó ò Ò ô Ô ö Ö õ Õ ø Ø p P q Q r R s S š Š ß t T ™ u U ú Ú ù Ù û Û ü Ü v V w W x X y Y ý Ý ÿ Ÿ z Z ž Ž þ Þ

Analytics Edition	Sort Order default	Associated sort sequence
	ation algorithm)	

Changing the Sort Order

You can change the **Sort Order** to a different language if it better matches the data you are analyzing. In the Unicode edition of Analytics, you can also make this change on a command basis by using the ISOLOCALE parameter in the command line or a script.

Modifying a sort sequence

In the non-Unicode edition of Analytics, when you select a different language, you have the option of modifying the associated sort sequence by changing the order of the characters in the **Sort Order** text box.

You also have the option of creating a custom sort sequence by selecting **Custom** in the **Sort Order** field and specifying a sequence, or by entering `SET ORDER <TO> values` in the command line or a script and specifying a sequence. Whatever characters you specify will be sorted before all other

characters, and in the sequence you specify. For example, you could specify that lowercase and uppercase letters are intermixed by entering the values `aAbBcC...`. Specifying `SET ORDER` returns the sort sequence to its default setting.

Default sort sequence based on byte order

The default sort sequence for individual languages is derived from the byte order of each character in its character set. You can view the byte order of characters in character sets using the Windows Character Map.

Sorting records

You can sort records into an ascending or descending sequential order and output the results to a new, physically reordered Analytics table. Outputting to an Analytics table is the only output option.

Sorting records is a prerequisite for several Analytics operations. For more information, see "Should I do an explicit sort or use Presort?" on page 1129

Sorting can also be a useful analytical operation in itself, bringing patterns and anomalies into focus.

> **Note**
> Indexing records is an alternative to sorting them, and in some situations may be a better choice. For more information, see "Should I sort or index?" on page 1121

Should I output the entire record or only specified fields?

When sorting, you have the option of including the entire record in the sorted output table, or only specified fields. There are implications associated with each option, summarized below.

The option you choose can also affect sorting speed. For more information, see "How to speed up sorting" on the next page.

> **Tip**
> If you want some of the characteristics of outputting by field, but you need the entire record, output by field and select all fields.

Output type	Implications
Record	◦ The entire record is included in the sorted output table. ◦ Computed fields are preserved as computed expressions. ◦ Related fields cannot be included. However, the new output table is automatically related to the original child table and you can add fields from the child table to the output table view.
Fields	◦ Only specified fields are included in the sorted output table. Key fields are automatically included and do not need to be specified. ◦ Computed fields are converted to physical fields and populated with the actual computed values. ◦ Related fields can be included. They become permanent physical fields in the output table. The new output table is no longer related to the original child table.

Sorting on multiple key fields

You can sort records using one key field, or you can create nested sorting schemes by sorting on multiple key fields – primary key field, secondary key field, and so on. Nested sorting supports mixing data types, and mixing ascending and descending order, across key fields.

Example

You want to sort a transaction table in ascending order on a key date field, and within each date in descending order on a key amount field.

The result below illustrates nested sorting that mixes data type (datetime and numeric), and ascending and descending order.

Date field (ascending order)	Amount field (descending order, nested)
15 Jan 2011	$2300.00
15 Jan 2011	$1200.00
15 Jan 2011	$600.00
16 Jan 2011	$900.00
16 Jan 2011	$100.00
17 Jan 2011	$4700.00
17 Jan 2011	$900.00
17 Jan 2011	$500.00

How to speed up sorting

Sorting very large tables, with millions of records, can be time consuming. Sorting requires a significant amount of system resources and can be slowed down if you are performing other tasks simultaneously.

Improve sorting speed

Two options can improve sorting speed:

- **Output a subset of fields** – If you need only a portion of the data contained in a record, do not include the entire record in the sorted output table. Select only the fields you need, which in most cases speeds up the sorting process.

  The smaller the subset of fields, as a percentage of the total number of fields, the greater the performance improvement.
- **Increase the memory available for sorting** – You can allocate a specific amount of memory for sorting, to a maximum of 2000 MB. Go to **Tools > Options > Table > Sort Memory**, or use the SET SORTMEMORY command.

Additional suggestions

If the amount of time required to sort large tables remains an issue for you, consider:

- upgrading your computer hardware
- creating a script to sort data on a scheduled basis overnight

Should I do an explicit sort or use Presort?

Sorting records prior to any of the following operations is either a prerequisite, or recommended:

- joining tables
- merging tables
- summarizing (if you want a single group for each set of identical values in the key field)
- testing for duplicates
- testing for gaps

All these operations include the **Presort** option, which allows you to incorporate a preliminary sequential sorting of records as part of the operation.

If you are performing two or more of these operations on the same table, explicitly sorting the table first, rather than repeatedly using **Presort**, may be more efficient, especially if the table contains a large number of records.

Verifying that all source records are in the output table

If you are sorting and outputting all the records in a table, you can set a control total on a numeric field to verify that all the records are in fact output to the new table.

You set a control total for a field in the **Table Layout** dialog box. Once you have sorted and output the records, in the new table select **Tools > Table History** to compare the input and output control totals. For more information, see "Define a physical field" on page 717.

Steps

You can sort records by one or more key fields in the active table and output the results to a new Analytics table. You can include the entire record in the sorted output table, or only specified fields.

Show me how

> **Note**
> You need free disk space at least 2.5 times the size of the file being sorted for the creation of a temporary file used during the sorting process.

1. In the Navigator, open the table you want to sort.
2. Select **Data > Sort**.
3. On the **Main** tab, do one of the following:
   - Select the key field(s) from the **Sort On** list.
   - Click **Sort On** to select the key field(s), or to create an expression.

   > **Tip**
   > If you click **Sort On**, you can optionally specify a descending sort order in the output results for one or more of the key fields by clicking the sort arrow  (the default sort order is ascending).

4. To output entire records, or only specified fields, do one of the following:
   - Leave **Record** selected if you want to include the entire record in the sorted output table.
   - Select **Fields** if you want to include only a subset of fields in the sorted output table.

   > **Note**
   > If you need only a portion of the data contained in a record, select **Fields**, especially if the table is large.
   > For more information, see "How to speed up sorting" on page 1128.

5. If you selected **Fields**, do one of the following:
   - Select the appropriate non-key fields from the **Other Fields** list.
   - Click **Other Fields** to select the non-key field(s), or to create an expression.

   > **Tip**
   > You can **Ctrl+click** to select multiple non-adjacent fields, and **Shift+click** to select multiple adjacent fields.

6. In the **To** text box, specify the name of the output table.

7. On the **More** tab:
   a. (Optional) To specify that only a subset of records are processed, select one of the options in the **Scope** panel.
   b. Click **OK**.

Sort dialog box options

The tables below provide detailed information about the options in the **Sort** dialog box.

Main tab

Options - Sort dialog box	Description
Sort On	Specifies the key field or fields to use to sort the table. ◦ You can select the field or fields from the **Sort On** list. ◦ You can also click **Sort On** to select the field or fields, to create an expression, or to specify a descending sort order, then click **OK**. **Note** Sorting on logical fields requires that **Include Filters in Field Lists** is selected (**Tools > Options > Interface**). **Key field guidelines:** ◦ **Single key field** - If you select only one key field, records within each sorted group retain their original sort order relative to one another. ◦ **Multiple key fields** - If you select more than one key field, the order in which you select the fields dictates the nested sorting priority. The records are sorted by the first field you select, and if there are multiple occurrences of a value in the first field, the records within the group are then sorted by the second field you select, and so on. For more information, see "Sorting on multiple key fields" on page 1128. ◦ **Key field column order** • If you include the entire record in the output table, the key field column order in the resulting table is the same as the column order in the source table, regardless of the order in which you select the key fields. • If you include a subset of fields in the output table, the key field column order in the resulting table is the order in which you select them. As a group, key fields appear before non-key fields in the output table. ◦ **Related key field** - If you want to sort by a field in a child table in a table relation: • Click **Sort On**. The **From Table** drop-down list in the **Selected Fields** dialog box allows you to select the appropriate child table. **Caution** If you sort by a related key field when using the **Record** option, be aware that the related key field is not included in the sorted output table, which can be confusing.
Record Fields	Specifies whether to include the entire record in the sorted output table, or only a subset of fields.

Options – Sort dialog box	Description
	◦ **Record** – includes entire records. The fields in the output table maintain the same order as the source table. ◦ **Fields** – includes a selection of individual fields. The fields in the output table appear in the order you select them. If you are including one or more computed fields: ◦ select **Record** to preserve the fields as computed expressions ◦ select **Fields** to convert the fields to physical fields of the appropriate data type and populate them with the actual computed values If you want to include fields from a child table in a table relation: ◦ select **Fields** You cannot include child table fields using the **Record** option.
Other Fields	If you selected **Fields**, specifies the non-key fields to include in the sorted output table. ◦ You can select the appropriate fields from the **Other Fields** list. ◦ You can also click **Other Fields** to select the appropriate fields, or to create an expression, then click **OK**. **Note** Key fields are automatically included in the output table. They are ignored when specified as **Other Fields**. As a group, key fields appear before other fields in the output table. If you want to select fields from a child table in a table relation: ◦ Click **Other Fields**. The **From Table** drop-down list in the **Selected Fields** dialog box allows you to select the appropriate child table.
If	(Optional) Allows you to create a condition to exclude records from processing. You can enter a condition in the **If** text box, or click **If** to create an IF statement using the **Expression Builder**.
To	Specifies the name and location of the output table. ◦ **To save the output table to the Analytics project folder** – enter only the table name. ◦ **To save the output table in a location other than the project folder** – specify an absolute or relative file path, or click **To** and navigate to a different folder. For example: `C:\Results\Output.fil` or `Results\Output.fil`. Regardless of where you save the output table, it is added to the open project if it is not already in the project. If Analytics prefills a table name, you can accept the prefilled name, or change it.
Local	If you are connected to a server table, specifies where to save the output table. ◦ **Local selected** – saves the output table to the same location as the Analytics project, or to a specified path, or location you navigate to. ◦ **Local deselected** – saves the output table to the Prefix folder on AX Server.
Use Output Table	Specifies whether the Analytics table containing the output results opens automatically

Options – Sort dialog box	Description
	upon completion of the operation.

More tab

Options – Sort dialog box	Description
Scope panel	Specifies which records in the source table are processed: ◦ **All** – (default) all records in the source table are processed. ◦ **First** – select this option and enter a number in the text box to start processing at the first record in the source table and include only the specified number of records. ◦ **Next** – select this option and enter a number in the text box to start processing at the currently selected record in the source table view and include only the specified number of records. The actual record number in the leftmost column must be selected, not data in the row. ◦ **While** – select this option to use a WHILE statement to limit the processing of records in the source table based on criteria. • You can enter a condition in the **While** text box, or click **While** to create a WHILE statement using the **Expression Builder**. • A WHILE statement allows records to be processed only while the specified condition evaluates to true. • You can use the **While** option in conjunction with the **All**, **First**, or **Next** options. **Note** The number of records specified in the **First** or **Next** options references either the physical or the indexed order of records in a table, and disregards any filtering or quick sorting applied to the view. However, results of analytical operations respect any filtering. If a view is quick sorted, **Next** behaves like **First**.
Append To Existing File	Specifies that the output results are appended (added) to the end of an existing Analytics table. The resulting combined table is considered unsorted because the sorted records are appended to the end of the target table, without consideration of any existing sort order in the target table. ◦ You can select **Append To Existing File** if you are certain the records or fields and the target table are identical in structure. ◦ You can leave **Append To Existing File** deselected if you want Analytics to compare the record lengths of the output results and the existing table. If the record lengths are not identical, the data structure is not identical, and the append will not work correctly. **Note** Leaving **Append To Existing File** deselected is recommended if you are uncertain whether the output results and the existing table have an identical data structure. For more information about appending and data structure, see "Appending output results to an existing table" on page 200.

Options – Sort dialog box	Description
OK	Executes the operation. If the overwrite prompt appears, select the appropriate option. If you are expecting the **Append** option to appear and it does not, click **No** to cancel the operation and see "Appending output results to an existing table" on page 200.

Indexing records

Indexing creates a separate index file (.inx file) that allows access to the records in an Analytics table in a sequential order rather than a physical order (that is, the raw data order).

Indexing does not physically reorder data in tables. However, when a table's index is active, the data in the view is reordered in accordance with the order specified by the index, and analytical operations process the data based on this order. If a table has more than one view, all views are subject to an active index.

When an index is active, the word **Indexed** prefaces the record count in the status bar. For example: **Indexed Records: 500**.

When the index is inactive, the records in a view revert to their original physical order. Upon opening an Analytics table, any existing indexes are inactive by default.

> **Note**
> Sorting records is an alternative to indexing them, and in some situations may be a better choice. For more information, see "Should I sort or index?" on page 1121

Indexing and field type

You can index any type of field, including computed fields and ad hoc expressions, regardless of data type.

Indexing on logical fields requires that **Include Filters in Field Lists** is selected (**Tools > Options > Interface**).

Multiple indexes for a single table

You can create multiple indexes for a single table, and switch between indexes as required, which can be useful when initially assessing a set of data. Only one index can be active at a time.

Nested indexing

You can index records using one key field, or you can create nested indexing schemes by indexing on multiple key fields (primary key field, secondary key field, and so on).

Nested indexing supports mixing ascending and descending order, and mixing data types, across key fields.

Nested indexing with mixed ascending and descending order

You want to see the largest transaction amounts for each day in an unsorted transaction table. You index the table in ascending order on a date key field, and within each day in descending order on an amount key field.

Date field (ascending)	Amount field (descending, nested)
15 Jan 2011	$2300.00
15 Jan 2011	$1200.00
15 Jan 2011	$600.00
16 Jan 2011	$900.00
16 Jan 2011	$100.00
17 Jan 2011	$4700.00
17 Jan 2011	$900.00
17 Jan 2011	$500.00

Indexing is restricted to Analytics tables

Indexing is restricted to Analytics tables – that is, tables with a .fil source data file. You can index both local and server-based Analytics tables if they have .fil files.

You cannot index database tables that you connect to using a database profile, because there is no .fil file. Data is read directly from the database. To order data in this situation, you can use a SQL ORDER clause in the **Data Definition Wizard** while accessing the database.

Indexing required for some Analytics commands

Indexing is a prerequisite for using the **Find Literal** and **Seek Expression** options in the **Search** dialog box when searching Analytics tables. (The options are the equivalent of the FIND and SEEK

commands.)

These options are available only if:

- a table is indexed
- the index is active
- the index's primary key field is a character field indexed in ascending order

The table can have a nested index, but only the primary key field is searched.

Conditional indexes

Indexes can include If, First, Next, and While parameters, in which case they become conditional indexes. Only those records that match the condition are indexed, or are displayed or available for analysis when the conditional index is active.

Show me more

Every time you activate the index the condition is automatically reapplied. You can facilitate certain types of analysis by using conditional indexes to create subsets of larger tables.

When a conditional index with an If parameter is active, the words **Filtered Index** preface the record count in the status bar. For example: **Filtered Index Records: 500**. When conditional indexes with First, Next, and While parameters are active, the word **Indexed** prefaces the record count, like indexes without any conditions.

Indexes and filters

When creating a conditional index with an If parameter or a filter, you can include a global filter (a filter on a view), a local filter (a filter within a command), or both.

The table below provides examples of including filters in indexes, and shows the effect that the indexes have on the sample data.

The filters are part of the indexing syntax, which you can view in the log, or in the index details. For more information, see "View index details" on page 1141.

Type of filter	Description/Indexing syntax	vendor_ID	trans_ amount
None	No index (physical order)	212 108 359 108 359 212 359 212	1400.00 3400.00 1600.00 1100.00 3400.00 1200.00 2200.00 1700.00

Type of filter	Description/Indexing syntax	vendor_ID	trans_ amount
		359 108	1400.00 2300.00
Global	Index contains only vendor #359 records `INDEX ON trans_amount TO "vendor 359 transactions"` Global filter: `vendor_ID = "359"`	359 359 359 359	1400.00 1600.00 2200.00 3400.00
Local	Index contains only transaction amounts $2000 or greater `INDEX ON trans_amount IF trans_amount >= 2000 TO "trans amount 2000 or greater"`	359 108 108 359	2200.00 2300.00 3400.00 3400.00
Global-Local	Index contains only vendor #359 records with transaction amounts of $2000 or greater `INDEX ON trans_amount IF trans_amount >= 2000 TO "vendor 359 transactions 2000 or greater"` Global filter: `vendor_ID = "359"`	359 359	2200.00 3400.00

Steps

Index records

You can index records by one or more key fields in the active table, and use the resulting index to temporarily reorder the records without affecting the underlying physical order of the data.

Show me how

1. Select **Data > Index**.
2. On the **Main** tab, do one of the following:
   - Select the field(s) to index from the **Index On** list.
   - Click **Index On** to select the field(s), or to create an expression.

   If you select more than one field, the order in which you select the fields dictates the nested indexing priority. The records are indexed by the first field you select, and if there are multiple occurrences of a value in the first field, the records within the group are then indexed by the second field you select, and so on. If you do not select additional fields, records within a group retain their original sort order relative to one another.

   For information about indexing using expressions and computed fields, see "Sorting or indexing using a computed key field" on page 1143.

> **Note**
> The combined length of the fields being indexed cannot exceed 247 characters.

3. If you clicked **Index On**, you can optionally specify a descending index order for one or more selected fields by clicking the sort arrow  (the default is ascending).
4. If there are records in the current view that you want to exclude from processing, enter a condition in the **If** text box, or click **If** to create an IF statement using the **Expression Builder**.

   > **Note**
   > The **If** condition is evaluated against only the records remaining in a table after any scope options have been applied (**First**, **Next**, **While**).

   The IF statement considers all records in the view and filters out those that do not meet the specified condition.
5. Do one of the following:
   - In the **To** text box, specify the name of the index file.
   - Click **To** and specify the index file name, or select an existing index file in the **Save** or **Save File As** dialog box to overwrite the file.

   If Analytics prefills an index file name, you can accept the prefilled name, or change it.

   > **Note**
   > Index names are limited to 64 alphanumeric characters. The name can include the underscore character (_), but no other special characters, or any spaces. The name cannot start with a number.

   > **Tip**
   > A best practice is to give indexes meaningful names that describe the nature of the ordering imposed by the index. For example, “Date_Amount_D” could be the name of an index that orders a table by Date in ascending order, and within each day by Amount in descending order.

6. Select or deselect **Use Output Table** depending on whether or not you want to activate the index immediately.

   You can activate a table’s index at any time by selecting it from the **Index** drop-down list at the top right of the view.
7. Click the **More** tab.
8. Select the appropriate option in the **Scope** panel:
   - **All**
   - **First**
   - **Next**
   - **While**

Show me more

All	This option is selected by default. Leave it selected to specify that all records in the view are processed.
First	Select this option and enter a number in the text box to start processing at the first record in the view and include only the specified number of records.
Next	Select this option and enter a number in the text box to start processing at the currently selected record in the view and include only the specified number of records. The actual record number in the leftmost column must be selected, not data in the row.
While	Select this option to use a WHILE statement to limit the processing of records in the view based on a particular criterion or set of criteria. You can enter a condition in the **While** text box, or click **While** to create a WHILE statement using the **Expression Builder**. A WHILE statement allows records in the view to be processed only while the specified condition evaluates to true. As soon as the condition evaluates to false, the processing terminates, and no further records are considered. You can use the **While** option in conjunction with the **All**, **First**, or **Next** options. Record processing stops as soon as one limit is reached.

> **Note**
> The number of records specified in the **First** or **Next** options references either the physical or the indexed order of records in a table, and disregards any filtering or quick sorting applied to the view. However, results of analytical operations respect any filtering.
> If a view is quick sorted, **Next** behaves like **First**.

9. Click **OK**.
10. If the overwrite prompt appears, select the appropriate option.

    An entry for the index is added to the **Index** drop-down list in the View tab. If you selected **Use Output Table**, the index is activated and the table is sorted according to the index.

Activate or deactivate indexes

You can activate an index at the time you create it, or at any time after creating it. Upon opening an Analytics table, any existing indexes are inactive by default.

Show me how

- To activate an index, do one of the following:
  - When creating an index, select **Use Output Table** in the **Index** dialog box to activate the index right away.
  - Select the index from the **Index** drop-down list at the top right of the view.
- To deactivate an index, do one of the following:
  - Select **(None)** in the **Index** drop-down list at the top right of the view.
  - Switch to another index.
  - Close the table.

View index details

You can view the details of an index – that is, the actual syntax of the specific Index command. The command syntax includes the key field(s), and any parameters, filters, or expressions. Index details reveal exactly how a particular index is processing the records in a table.

Show me how

1. Open the table containing the index.
2. Right-click the table in the **Navigator** and select **Properties**.
3. Click the **Indexes** tab, select the index name, and click **Details**.

   The **Index Properties** dialog box displays the index details:

   - **Command** – displays the syntax of the specific Index command, including any local filter.
   - **Filter** – displays the syntax of any global filter that is part of the index.
4. Click **OK** and **OK** again to exit the **Table Properties** dialog box.

Maintain indexes

You can copy, rename, or delete an index in the **Indexes** tab of the **Table Properties** dialog box. You can also add additional indexes from the same location.

Show me how

> **Note**
>
> You can perform these maintenance tasks only through Analytics. If you directly rename an index file (.inx file) in a Windows folder the index file is automatically recreated with the original name the next time you activate the index in Analytics. If you directly delete an index file, the index file is automatically recreated the next time you activate the index.

1. Open the table containing the index.
2. Right-click the table in the **Navigator** and select **Properties**.
3. Click the **Indexes** tab, select the index name, and do one of the following:
   - Click **Copy** to copy the index.

     The index is copied with an incrementing number added to the end of the index name.
   - Click **Rename**, enter a new name, and click **OK** to rename the index.

     > **Note**
     >
     > Index names are limited to 64 alphanumeric characters. The name can include the underscore character (_), but no other special characters, or any spaces. The name cannot start with a number.

   - Click **Delete**, then click **Delete** again to delete the index.
4. If you want to add a new index, click **Add**.

   The **Index** dialog box appears, allowing you to create an index in the usual manner.

5. Click **OK** to exit the **Table Properties** dialog box.

Sorting or indexing using a computed key field

If the format of the data in a physical key field prevents the accurate sorting or indexing of a table, you may be able to achieve accurate results by creating a computed key field. Examples of data format impeding sorting or indexing include numbers that inconsistently contain non-numeric prefixes or leading blanks, and names that are inconsistently capitalized. You can create a computed key field to make the data consistent, and then sort or index using the computed key field.

You can also sort or index using a computed key field to order tables in ways other than a strictly sequential ordering of key field values. For example, you can use a computed key field to reverse the physical order of records in a table, group even dollar values, or group names that contain a particular string.

Make the computed key field values visible

A best practice when indexing using a computed key field is to add the computed field to the view so that you can see exactly how the computed values are being used to index the table. When you sort using a computed key field, the computed field is automatically included in the new, sorted table.

It is possible to directly incorporate expressions in the sorting and indexing operations, and achieve the same results that you can achieve with a computed key field. However, this approach is not recommended because it hides the computed values that are being used to order the table.

Examples of sorting or indexing using a computed key field

Several examples of sorting or indexing using a computed key field appear below. For comparison, the non-computed, sequential order of the physical key field is also included.

Show me more

Description/Computed key field expression	Analytics Function	Key field (physical order)	Key field (sequential order, non-Unicode edition of Analytics)	Computed key field (sequential order)	Key field (order based on computed key field order)
Ignore non-numeric characters and blanks, sort/index on numeric	INCLUDE ( )	92 12	20 #85	12 20	12 20

Description/Computed key field expression	Analytics Function	Key field (physical order)	Key field (sequential order, non-Unicode edition of Analytics)	Computed key field (sequential order)	Key field (order based on computed key field order)
characters only `INCLUDE(dept_ID, "1234567890")`		T-38 20 #85	12 92 T-38	38 85 92	T-38 #85 92
Avoid sorting/indexing discrepancies caused by differing upper and lower case `UPPER(last_name)`	UPPER( )	Smythe JONES Smith JOHNSON SMYTHE Jones SMITH Johnson	JOHNSON JONES Johnson Jones SMITH SMYTHE Smith Smythe	JOHNSON JOHNSON JONES JONES SMITH SMITH SMYTHE SMYTHE	JOHNSON Johnson JONES Jones Smith SMITH Smythe SMYTHE
Invert the physical order of records in a table **Note** Requires computed key field in descending order. Record numbers always display in ascending order even if a descending order is applied. `RECNO( )`	RECNO( )	82.12 87.00 62.79 97.47 43.00	43.00 62.79 82.12 87.00 97.47	1 2 3 4 5	43.00 97.47 62.79 87.00 82.12
Group even dollar amounts, sequentially order amounts within the group	MOD( )	82.12 **87.00** 62.79 97.47 **43.00**	**43.00** 62.79 82.12 **87.00** 97.47	0.00 0.00 0.12 0.47 0.79	**43.00** **87.00** 82.12 97.47 62.79

Description/Computed key field expression	Analytics Function	Key field (physical order)	Key field (sequential order, non-Unicode edition of Analytics)	Computed key field (sequential order)	Key field (order based on computed key field order)
Note Requires a nested sort/index of key field inside computed key field. `MOD(trans_amount, 1.00)`					
Group values containing a particular string, sequentially order values within the group **Note** Requires computed key field in descending order. Requires a nested sort/index of key field inside computed key field. `FIND("Hardware", vendor_name)`	FIND( )	**Lilydale Hardware**	Binford Tools	T	**Global Trade Hardware**
		Triathalon Group	Bolton Distribution	T	**Lilydale Hardware**
		Wholesome Hardware	**Global Trade Hardware**	T	**Wholesome Hardware**
		Steel Case Manufacturing	Industrial Equipment Co-Op	F	Binford Tools
		Industrial Equipment Co-Op	**Lilydale Hardware**	F	Bolton Distribution
		Global Trade Hardware	Steel Case Manufacturing	F	Industrial Equipment Co-Op
		Binford Tools	Triathalon Group	F	Steel Case Manufacturing
		Bolton Distribution	**Wholesome Hardware**	F	Triathalon Group

Steps

The steps for sorting or indexing a table using a computed key field are explained in general terms below.

For detailed information about creating computed fields, and performing sorting or indexing, see the relevant sections of this guide.

1. Using an appropriate expression, create a computed key field based on the sort or index physical key field.
2. If you are going to perform an indexing operation, add the computed key field to the view.
3. Using the computed field as the key field, perform the regular sorting or indexing procedure.
   - If necessary, specify a descending order for the computed key field. Some expressions require a descending order to position results at the top of the table.
   - If you want to sequentially order results within groupings created by the computed key field expression, select both the computed key field and the physical key field when sorting or indexing. Make sure to select the computed key field first, so that it takes precedence over the physical key field.

Filtering data

Filters are an essential tool when analyzing data. They allow you to exclude the data in a table that is not currently relevant and focus analysis on a targeted subset of records.

Excluded data is not displayed in the View tab, or processed by Analytics operations such as extracting. Excluded data is only hidden, not deleted, and it can be displayed at any time by removing the filter. Filters are associated with a specific table in an Analytics project rather than the entire project.

How filters work

A filter is a logical expression that evaluates each record in a table and returns a value of **True** (T) or **False** (F) – for example, `Invoice_Amount > 1000.00`

Data that evaluates to True is included in the filtered table, or in an Analytics operation, and data that evaluates to False is excluded.

You can apply filters from a number of different locations in Analytics, and you can use them in conjunction with one another.

Types of filters

You can create several different types of filters in an Analytics project:

- Global filters, also called view filters
- Quick filters
- Local filters, also called command filters
- Data filters

Filters can be **ad hoc** – that is, not permanently saved with a table – or they can be **named** and saved with a table for later reuse. Named filters can also be saved in a workspace so they can be shared among multiple tables.

Global filters (view filters)

A global filter applies to the view, or views, associated with a table layout, and restricts which records are displayed, or processed. When a global filter is applied, any operations performed on the table are performed on only the records that the filter includes.

A global filter remains active until you remove it, replace it with another global filter, or close the table. You can make a global filter the default filter for a table so that it is automatically applied every time you open the table.

For more information, see "Global filters (view filters)" on page 1151.

Quick filters

A quick filter is a global filter that is applied by right-clicking in a view and using the **Quick Filter** option on the context menu. Quick filters are convenient because they allow you to select filter values and criteria with the mouse, rather than specifying them manually, and they automatically populate the Filter text box with valid filter syntax.

Because of the need to select filter values and criteria with the mouse, quick filters have certain limitations. They typically cannot be used to create complex filters with multiple criteria.

For more information, see "Quick filtering data in a view" on page 1109.

Local filters (command filters)

A local filter applies to a single execution of a single Analytics command, restricting which records in a table the command processes. When the command completes its processing, the local filter is no longer active.

For more information, see "Local filters (command filters)" on page 1160.

Data filters

Data filters serve a specific purpose. They provide a method for selectively defining data in data sources that contain more than one record type, such as Print Image (Report) files and Multiple Record Type files. Unlike the other types of filters, they are not intended for general use when analyzing data in Analytics.

For more information, see "About data filters" on page 764.

Ad hoc and named filters

Ad hoc filters

You can apply a global or local filter using the filter syntax alone – for example, `Invoice_Amount > 1000.00` – in which case the filter is ad hoc. Ad hoc filters are not permanently saved with a table.

Ad hoc global filters are retained while they appear in the filter history associated with a specific table.

Ad hoc local filters are retained for the duration of a single Analytics operation – although they can be retrieved from the command log, if necessary.

Named filters

You can name and save a global or local filter for subsequent reuse, in which case it is permanently saved with the associated Analytics table. For example, you could name and save the ad hoc filter `Invoice_Amount > 1000.00` as “Inv_greater_than_1K”.

When you reapply the filter, you specify the filter name, rather than recreating the syntax, which saves labor. Naming filters also makes it easy to differentiate between a number of saved filters. For example:

- “Inv_less_than_1K”
- “Inv_1K_to_5K”
- “Inv_greater_than_5K”

You can name and save filters when you create them, or at any subsequent point if the ad hoc filter still appears in the filter history. Once a filter is named and saved, it is available to use as a global filter with any view associated with the table, or as a local filter with any operation performed on the table.

For information about how to name and save a filter, or convert an ad hoc filter to a named filter, see "Apply a global filter to a view" on page 1156.

Filter history

When you apply a global filter to a table it is saved in the filter history associated with the table. As long as the filter remains in the filter history, it can be reapplied by selecting it from the Filter drop-down list at the top of the view.

Both ad hoc and named global filters are saved in the filter history. Local filters are not saved in the filter history.

Additional filter history details:

Tables, views, and filter history	Each table has a separate filter history. Multiple views of the same table share the same filter history.
Persistence of the filter history	The filter history persists when you close the table, close the project, or close Analytics.
Sequence of filters in the list	The most recently applied filter appears at the top of the Filter drop-down list.
Maximum number of stored filters	A maximum of 10 filters are stored. If you exceed the maximum, the oldest filter is removed from the bottom of the list, and the most recent filter is added to the top.
Redundant filters	Filters in the filter history are unique. Applying a filter multiple times does not cause redundant filter history entries.
Deleted named filters	Deleted named filters are not removed from the filter history but they no longer function.

Clearing the filter history	You can clear the entire filter history by right-clicking the Filter text box and selecting **Clear History**. You cannot selectively remove filters from the filter history. Clearing the filter history does not delete named filters.

Summary of filter retention

The table below summarizes filter retention:

	Permanently saved with table	Added to filter history
Ad hoc global filter	No	Yes
Ad hoc local filter	No	No
Named global filter	Yes	Yes
Named local filter	Yes	No

Configurable filter options

Two configurable options allow you to control aspects of filter behavior:

Include Filters in Field Lists	Controls whether named filters appear in field lists. For more information, see "Interface options" on page 120.
Hide Filtered Records	Controls whether filtered records in views are hidden, or displayed but visually de-emphasized. For more information, see "View options" on page 127.

Global filters (view filters)

Global filters restrict which records in a view are displayed, or processed by Analytics operations.

You can build simple filters with a single criterion to broadly filter records, or complex filters with multiple criteria to isolate very specific subsets of data.

Simple versus complex filters

A simple filter

You can create a simple filter with a single criterion to isolate records related to a particular entity, such as:

- a name
- a date
- an account number

For example, you could filter an Accounts Payable table by vendor number so that only the records associated with a particular vendor are displayed or processed:

```
Vendor_No = "14438"
```

A more complex filter

If you need to isolate more specific subsets of data, you can create more complex filters with multiple criteria.

For example, you could create a filter that restricts an Accounts Payable table to invoices that meet all three of these requirements:

- vendor 14438
- submitted in 2014
- $1000.00 or greater

```
(Vendor_No = "14438") AND (BETWEEN(Invoice_Date,`20140101`,`20141231`)) AND
(Invoice_Amount >= 1000.00)
```

You can apply only one filter at a time to a view, but as the example above shows, you can use Boolean operators such as AND and OR to combine multiple criteria in a single filter.

For more information about Boolean operators, see "Operators in Analytics expressions" on page 797.

Filter expressions specify the requirements for inclusion

When you create a filter expression such as `Vendor_No = "14438"` you are specifying the requirements or criteria for records **to be included** in the filtered table.

From the standpoint of Boolean logic, records for which the filter expression evaluates to **True** are included in the filtered table. Records that evaluate to **False** are excluded.

So in this example:

- all records with vendor number 14438 evaluate to True, and are included
- all records with vendor number 90215 evaluate to False, and are excluded

> **Tip**
> To help visualize which records are included by a filter, imagine prefacing the filter expression with the phrase "Include records if". This technique can be helpful when constructing complex expressions, or when using Boolean operators that negate, such as NOT, and Not Equal To (<>).

Examples of filter expressions

The examples below provide four variations of filtering using the same group of filter values and the same set of data.

Include records if:

- all values are matched
- any values are matched
- all values are not matched
- any values are not matched

Include records if ALL values are matched

The filter expression below includes records in the filtered table if they belong to vendor 14438, and they are dated 15 July 2014, and the invoice amount is $1,000.

In other words, all three criteria must be met for a record to be included in the filtered table.

```
(Vendor_No = "14438") AND (Invoice_Date = `20140715`) AND (Invoice_Amount =
1000.00)
```

Included?	Vendor Number	Invoice Date	Invoice Amount
YES	**14438**	**15 Jul 2014**	**$1000**
no	90215	**15 Jul 2014**	**$1000**
no	**14438**	25 May 2015	**$1000**
no	**14438**	**15 Jul 2014**	$500
no	90215	25 May 2015	$500

Include records if ANY values are matched

The filter expression below includes records in the filtered table if they belong to vendor 14438, or if they are dated 15 July 2014, or if the invoice amount is $1,000.

In other words, if any one of the three criteria is met a record is included in the filtered table.

```
(Vendor_No = "14438") OR (Invoice_Date = `20140715`) OR (Invoice_Amount =
1000.00)
```

Included?	Vendor Number	Invoice Date	Invoice Amount
YES	**14438**	**15 Jul 2014**	**$1000**
YES	90215	**15 Jul 2014**	**$1000**
YES	**14438**	25 May 2015	**$1000**
YES	**14438**	**15 Jul 2014**	$500
no	90215	25 May 2015	$500

Include records if ALL values are NOT matched

The filter expression below includes records in the filtered table if they do not belong to vendor 14438, and they are not dated 15 July 2014, and the invoice amount is not $1,000.

In other words, all three criteria must be met for a record to be included in the filtered table.

```
(Vendor_No <> "14438") AND (Invoice_Date <> `20140715`) AND (Invoice_Amount <>
1000.00)
```

Included?	Vendor Number	Invoice Date	Invoice Amount
no	14438	15 Jul 2014	$1000
no	**90215**	15 Jul 2014	$1000
no	14438	**25 May 2015**	$1000
no	14438	15 Jul 2014	**$500**
YES	**90215**	**25 May 2015**	**$500**

Include records if ANY values are NOT matched

The filter expression below includes records in the filtered table if they do not belong to vendor 14438, or if they are not dated 15 July 2014, or if the invoice amount is not $1,000.

In other words, if any one of the three criteria is met a record is included in the filtered table.

```
(Vendor_No <> "14438") OR (Invoice_Date <> `20140715`) OR (Invoice_Amount <>
1000.00)
```

Included?	Vendor Number	Invoice Date	Invoice Amount
no	14438	15 Jul 2014	$1000
YES	**90215**	15 Jul 2014	$1000
YES	14438	**25 May 2015**	$1000
YES	14438	15 Jul 2014	**$500**
YES	**90215**	**25 May 2015**	**$500**

Partial matching

Partial matching is supported when filtering character data – that is, the filter value can be contained by a longer value in the field you are using for filtering.

For example:

- `Vendor_Name = "R"` restricts a table to vendors with names beginning with “R”.
- `Address = "PO Box"` restricts a table to addresses that start with “PO Box”.

> **Note**
> Filter values must appear at the start of fields to constitute a match.

Partial matching is enabled when the **Exact Character Comparisons** option is off (the default setting). If the option is on, partial matching is disabled and the filter value must exactly match a value in a field to constitute a match. For more information, see "Table options" on page 122.

Filter retention

A global filter remains active until you remove it, replace it with another global filter, or close the table. You can make a global filter the default filter for a table so that it is automatically applied every time you open the table.

Global filters differ from local filters, which are active only during a single execution of a single Analytics operation.

When a global filter is active, the **Global Filter** indicator appears in the status bar followed by the filter syntax or the filter name, depending on whether the filter is ad hoc or named:

- **an ad hoc filter** - Global Filter: (Vendor_No = "14438")
- **a named filter** - Global Filter: Vend_14438

Different ways to create and apply a global filter

There are several different ways to create and apply a global filter:

- Manually enter the filter syntax in the Filter text box
- Create a quick filter
- Create a filter, or select an existing filter, using the **Expression Builder**
- Select an existing filter from the Filter drop-down list

Apply a global filter to a view

You can create a global filter, or select an existing filter, and apply it to a view to restrict which records are displayed, or processed by Analytics operations.

You can also specify that a global filter is the default filter for a table so that it is automatically applied every time you open the table.

Simple filters can be manually entered in the Filter text box, or created using quick filtering. More complex filters, using multiple criteria, are easier to create in the **Expression Builder**.

Create a new filter

To create a new global filter, use one of the following:

- **Filter text box** – Enter a filter expression in the Filter text box (for example, `Invoice_Amount > 1000.00`) and click **Set Filter**  .

  The filter is ad hoc and retained only while it appears in the filter history associated with the table.

- **Quick filter** – Create a quick filter. For more information, see "Quick filtering data in a view" on page 1109.

  The filter is ad hoc and retained only while it appears in the filter history associated with the table.

  > **Tip**
  > You can use quick filtering to automatically create valid filter syntax, and then manually edit the filter values to create the filter you want.

- **Expression Builder** – Click **Edit View Filter**  to open the **Expression Builder**, create the filter expression, optionally enter a name for the filter in the **Save As** text box, and click **OK**.

  If you enter a name for the filter, it is permanently saved with the table. If you do not enter a name, the filter is ad hoc and retained only while it appears in the filter history associated with the table.

  Filter names are limited to 256 alphanumeric characters and cannot begin with a number.

  For information about using the **Expression Builder**, see "Creating expressions using the Expression Builder" on page 801.

Count the number of filtered records

After applying a global filter, use the following method to count the number of records included by the filter.

1. From the Analytics main menu, click **Count**.
2. Click **OK**.

   The number of records included by the filter, and the total number of records in the table, appear in the status bar at the bottom of the Analytics interface. For example: **Records: 108/772**

Specify a global filter as the default table filter

1. Click **Edit View Filter** to open the **Expression Builder**, create the filter expression, enter a name for the filter in the **Save As** text box, and click **OK**.

   Filter names are limited to 256 alphanumeric characters and cannot begin with a number.
2. Select **Edit > Table Layout**.
3. In the **Edit Fields/Expressions** tab, double-click the name of the filter.
4. Select **Default Filter**.
5. Click **Accept Entry** and click **Close** to exit the **Table Layout** dialog box.

   The filter is permanently saved with the table, and is automatically applied every time you open the table. Only one default filter can be specified at a time for a table.

   To remove a default filter, select **Edit > Table Layout**, double-click the name of the filter, and deselect **Default Filter**. The filter is still applied to the table, but it is no longer the default.

Select an existing filter

If you want to select an existing filter, use one of the following:

- **Filter drop-down list** – Select the filter from the Filter drop-down list.

  The 10 filters most recently applied to the table appear in the list.
- **Expression Builder** – Click **Edit View Filter** to open the **Expression Builder**, double-click a named filter in the **Filters** list, and click **OK**.

  Only named filters permanently saved with the table appear in the **Filters** list. Ad hoc filters do not appear in the **Filters** list.

Convert an ad hoc filter to a named filter

1. Apply the ad hoc filter to the table.
2. Click **Edit View Filter** to open the **Expression Builder**, enter a name for the filter in the **Save As** text box, and click **OK**.

   The filter is permanently saved with the table.

Remove a currently applied filter

Click **Remove Filter**.

Removing a filter does not delete it. Ad hoc filters are retained while they appear in the filter history associated with the table. Named filters are permanently saved with the table.

Maintain a named filter

You can use the **Filters** dialog box to perform the following actions with the named filters associated with a table:

- Add
- Modify
- Duplicate
- Rename
- Delete

You can also view the syntax for a named filter without making any modifications.

> **Note**
> You cannot rename or delete a filter if it is currently applied to a table.

1. Open the table with the named filter that you want to maintain.
2. Select **Edit > Filters**.

   Analytics displays the list of named filters associated with the table.
3. If you want to add a new filter, click **New** and use the **Expression Builder** to create the filter.

   > **Note**
   > Only logical expressions can be used to define filters.

   For information about using the **Expression Builder**, see "Creating expressions using the Expression Builder" on page 801.

4. If you want to work with an existing filter, select it in the list and do one of the following:
   - Click **OK** to view or modify the selected filter in the **Expression Builder**. After viewing or modifying the existing filter syntax click **OK**.

     If you modified the filter it is updated when you click **OK** to close the **Expression Builder**.
   - Click **Duplicate** to duplicate the selected filter, and click **Done** to create an exact copy of the filter, or click **OK** to modify the expression used by the duplicated filter.

     > **Tip**
     > Duplicating a complex filter and modifying it can be easier than creating a filter from scratch.

   - Click **Rename**, enter a new name in the text box, and click **OK**.

     Filter names are limited to 256 alphanumeric characters and cannot begin with a number. Click **Done** to use the existing value of the filter, or click **OK** to modify the expression used by the filter.
   - Click **Delete** to delete the filter, click **Delete** again in the confirmation dialog box, and click **Done** to close the dialog box.

Local filters (command filters)

Local filters can be applied as part of an Analytics operation to restrict which records are processed by the operation. For example, you could include a local filter in a totaling operation on an Accounts Payable table to total only the invoices submitted by a particular vendor.

The expressions you use to create local filters work the same way as the expressions for global filters. For more information and specific examples, see "Global filters (view filters)" on page 1151.

Duration of local filters

A local filter applies to a single execution of a single Analytics operation only, and does not alter the display of records in a view. When the operation is complete, the local filter is no longer active.

Local filters differ from global filters, which remain active until you remove them, replace them with another global filter, or close the table.

Applying a local filter and a global filter at the same time

You can apply a local filter and a global filter at the same time. In this situation, an Analytics operation processes only those records that meet the criteria of both filters.

The two filters should be logically consistent. Records excluded by one filter cannot be included by the other filter. For example, if a global filter contains the criteria `(Invoice_Amount >= 1000.00)`, and a local filter contains the criteria `(Invoice_Amount >= 500.00)`, invoice amounts less than $1000.00 are excluded from processing, despite what the local filter specifies.

Create a local filter

As part of an Analytics operation, you can create a local filter, or select an existing named filter, to restrict which records are processed.

Simple filters can be manually entered in the **If** text box in Analytics command dialog boxes.

More complex filters, using multiple criteria, are easier to create in the **Expression Builder**. If you want to name and save a local filter, you must create it in the **Expression Builder**.

1. In an Analytics command dialog box, do one of the following:
   - Enter the filter expression in the **If** text box.

     For example, `Invoice_Amount > 1000.00`. The filter is ad hoc and retained only during processing of the Analytics operation.

- Click **If** to create a filter, or select an existing filter, using the **Expression Builder**.

2. If you are using the **Expression Builder**, do one of the following:
   - Create the filter expression, optionally enter a name for the filter in the **Save As** text box, and click **OK**.

     If you enter a name for the filter, it is permanently saved with the table. If you do not enter a name, the filter is ad hoc and retained only during processing of the Analytics operation.

     Filter names are limited to 256 alphanumeric characters and cannot begin with a number.

     For information about using the **Expression Builder**, see "Creating expressions using the Expression Builder" on page 801.
   - Double-click a named filter in the **Filters** list, and click **OK**.

     Only named filters permanently saved with the table appear in the **Filters** list. Ad hoc filters do not appear in the **Filters** list.

Searching data

You can use different methods for searching data in Analytics tables:

Desired result	Location to enter search expression	Method
Isolate all matching records	Filter text box	◦ **Basic search** - Do a basic search using operators such as Equals =, Less Than <, or Greater Than > ◦ **Quick search** - Do a quick search using the quick search or quick filter features ◦ **Functions** - Search using Analytics functions
Select the first matching record	**Search** dialog box	◦ **Commands** - Search using Analytics commands

> **Tip**
> Typically, users search to isolate all matching records, which returns a set of results. So normally you use the Filter text box and you can ignore the Search dialog box.

Searching to isolate all matching records

Basic search

To perform a basic search, enter an expression in the Filter text box at the top of a table view and press Enter.

The example below isolates all records that contain the name “United Equipment” in the Vendor_Name field:

Other basic searches using operators

- Isolates invoices of $5,000.00 or more:

```
Invoice_Amount >= 5000
```

- Isolates invoices in the third quarter of 2017:

```
(Invoice_Date >= `20170701`) AND (Invoice_Date <= `20170930`)
```

> **Tip**
> Build search expressions from scratch only if the expressions are simple. For more involved searching, use the quick filter method, or advanced searching using functions.

Search for blank, empty, or invalid values

You can search for blank text or numeric values, or blank or invalid datetime values. You can search for non-blank values by changing the operator you use in the expression.

Blank or non-blank text values

- Isolates all records in which the Vendor_Name field is blank:

```
Vendor_Name = " "
```

```
ISBLANK(Vendor_Name)
```

- Isolates all records in which the Vendor_Name field is not blank:

```
Vendor_Name <> " "
```

```
NOT(ISBLANK(Vendor_Name))
```

Blank or non-blank numeric values

- Isolates all records in which the Invoice_Amount field is blank, or zero (0):

```
Invoice_Amount = 0
```

- Isolates all records in which the Invoice_Amount field is not blank, and not zero (0):

```
Invoice_Amount <> 0
```

Blank or non-blank datetime values

- Isolates all records in which the Invoice_Date field is blank, or the value is invalid:

```
Invoice_Date = `19000101`
```

```
NOT VERIFY(Invoice_Date)
```

A date value can be invalid if it does not match the date format used by the field, or if the date does not exist. For example: 31 April 2020.

- Isolates all records in which the Invoice_Date field is not blank, and the value is valid:

```
Invoice_Date <> `19000101`
```

```
VERIFY(Invoice_Date)
```

Guidelines for basic searches

Field names	You must specify a field name to search in, and it must be the physical field name in the table layout, not the display name in the table view. **Tip** To see the physical field name, right-click a column header in the table view and select **Properties**.
Search in more than one field	You can build an expression that searches in more than one field. The easiest way to search across all fields in a record is to search using a function. For more information, see "Search and filter using Analytics functions" on page 1173.
Partial matching	Partial matching of search terms is not supported. For information about using partial matching, see "Search and filter using Analytics functions" on page 1173.
Quotation marks	Text search terms must be enclosed in "quotation marks".
Back quotes	Datetime search terms must be enclosed in `back quotes`.
Datetime format	◦ Datetime search terms must use YYYYMMDD or YYMMDD format. ◦ Any time portion must use hhmmss format, and be preceded a single blank space, the letter 't', or the letter 'T'. For example: `t183000` ◦ Do not use any separators such as slashes (/) or colons (:) between the individual components of dates or times.

Operators	For the list of valid operators, see "Operators in Analytics expressions" on page 797.
Related fields	To search in a related field you must specify the fully qualified field name: *table name.field name*.

Quick search and quick filter

Quick search and quick filter are two Analytics features that make searching easier by building the search expression for you in the Filter text box.

- **Quick search** – you enter a text term in the Filter text box
- **Quick filter** – you use the mouse to select search criteria

For more information, see:

- "Quick searching data in a table" on page 1114
- "Quick filtering data in a view" on page 1109
- "Global filters (view filters)" on page 1151

> **Note**
> Some limitations exist with quick search and quick filter, which are explained in the topics about these features.

Search using Analytics functions

Using functions to search gives you the greatest degree of power and flexibility. Similar to a basic search, you enter a search expression in the Filter text box, but the expression contains a function.

The example below uses the `FINDMULTI( )` function to isolate all records that contain at least one of the search terms anywhere in the record.

For detailed information about searching using functions, see "Search and filter using Analytics functions" on page 1173.

Select the first matching record

You can use an Analytics command, accessed from the main menu, to select the first record in a table that meets the search criteria. This capability is primarily useful in Analytics scripts, where it can be used in conjunction with other commands to perform certain tasks.

One of the commands allows you to go directly to a specific record number, which can be helpful in the Analytics user interface when navigating large tables.

For more information, see "Selecting the first matching record" on the facing page.

Selecting the first matching record

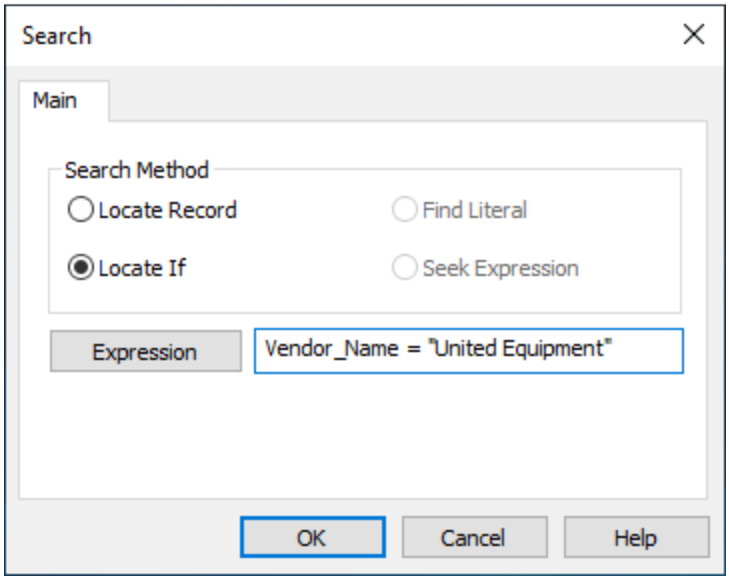
You can use an Analytics command to select the first record in a table that matches the search criteria. The record is selected, but not isolated, unlike the other types of searching in Analytics. The rest of the records are still present in the table view.

Usefulness in scripts

The ability to select the first matching record is primarily useful in Analytics scripts. For example, in conjunction with other scripting techniques, the commands below can be used to move sequentially through records in a table as a precursor to performing a repeated action based on the contents of each selected record.

Search dialog box

In the Analytics user interface, you access the commands in the **Search** dialog box (**Data > Search**).



The table below explains the different options in the **Search** dialog box. It also provides the equivalent ACLScript commands on the assumption that the options are primarily useful in Analytics scripts.

> **Note**
> You can click any command name below for detailed information about the command.

Search dialog box option	Equivalent Analytics command	Description
Locate Record	LOCATE RECORD	Selects a specific record number in a table.
Locate If	LOCATE	Selects the first occurrence of any type of literal, or of an expression that uses any data type, or mix of data types. The table does not have to be indexed. For example: ◦ Vendor_City = "New York" ◦ Invoice_Amount = 296.50 ◦ Invoice_Date = `20141231` ◦ Vendor_City = *v_city* ◦ Vendor_City = *v_city* AND Invoice_Amount > 1000
Find Literal	FIND	Selects the first occurrence of a character literal (for example, New York) in a character field indexed in ascending order. **Note** The FIND command and the FIND( ) function are two separate Analytics features with significant differences.
Seek Expression	SEEK	Selects the first occurrence of a character literal (for example, “New York”), or a character expression (for example, *v_city*), in a character field indexed in ascending order.

Index requirement

To use the **Find Literal** or **Seek Expression** options, you must first index the character field that you want to search, in ascending order. Both options search in the indexed field only.

If a table is indexed by more than one field (a nested index), only the primary key field is searched, assuming it is a character field indexed in ascending order. If an index is conditional, any records excluded from the view are also excluded from the search.

Guidelines

Data type	All options can be used with character fields. Only the **Locate If** option can be used with datetime or numeric fields.

Partial matching	Partial matching is supported when searching character fields, however the search string must appear at the start of the field. For example, `Vendor_Name = "Uni"` finds "United Equipment", but `Vendor_Name = "Equip"` does not.
Case sensitivity	When used to search character fields, all options are case-sensitive.
Performance	The **Locate If** option searches a table sequentially and therefore is slower than the **Find Literal** or **Seek Expression** options, which search indexed tables. However, the **Locate If** option does not require the time spent to index a table.
Record order	The **Locate If** option maintains the original order of the records in a table, which depending on the nature of your analysis may be desirable.

Select a specific record number in a table

1. From the Analytics main menu, select **Data > Search > Locate Record**.
2. Type the record number in the **Expression** text box and click **OK**.

   If the record number is found, it is selected and the table is positioned at the record.

Select the first occurrence of any type of literal or expression

1. From the Analytics main menu, select **Data > Search > Locate If**.
2. Do one of the following:
   - Enter an expression in the **Expression** text box and click **OK**.
   - Click **Expression** to open the **Expression Builder**, create an expression, click **OK**, and click **OK** again.

   The expression can be as simple or as complex as required, can involve one field or multiple fields, and can mix data types. For example:

   - `Vendor_Name = "United Equipment"`
   - `Invoice_Amount > 1000`
   - `Vendor_Name = "United Equipment" AND Invoice_Amount > 1000 AND Invoice_Date > `20140930``

   You must enclose character literal values in quotation marks, and datetime values in backquotes.

   If the specified value is found, the table is positioned at that record.

   If the specified value is not found, the table is positioned at the first record in the table.

Select the first occurrence of a character literal in an indexed table

1. Activate an index for the table you want to search.

   The table must be indexed by the character field you want to search.
2. From the Analytics main menu, select **Data > Search > Find Literal**.
3. Type a character literal value in the **Expression** text box and click **OK**.

   Do not enclose the character literal value in quotation marks unless the quotation marks are part of the data in the field. For example:

   - `United Equipment`
   - `R` (to find the first value starting with "R")

   If the specified value is found, the table is positioned at that record.

   If the specified value is not found, the message "No index matched key" is displayed. The table is positioned at the first record with a key field value greater than the specified value, or at the first record in the table if no value is greater than the specified value.

Select the first occurrence of a character literal or expression in an indexed table

1. Activate an index for the table you want to search.

   The table must be indexed by the character field you want to search.
2. From the Analytics main menu, select **Data > Search > Seek Expression**.
3. Do one of the following:
   - Enter a character-type expression or a character literal value in the **Expression** text box and click **OK**.
   - Click **Expression** to open the **Expression Builder**, create an expression, click **OK**, and click **OK** again.

   For example:

   - *v_vendor_name*
   - `"United Equipment"`

   You must enclose character literal values in quotation marks.

   If the specified value is found, the table is positioned at that record.

   If the specified value is not found, the message "No index matched key" is displayed. The table is positioned at the first record with a key field value greater than the specified value, or at the first record in the table if no value is greater than the specified value.

A comparison of Analytics search commands

The table below provides a high-level comparison of Analytics search commands. If you use any of the commands in an Analytics script, it can be useful to know how the specific rules that govern each command may differ.

	Locate Record / Locate If	Find Literal	Seek Expression
Data types searchable	◦ Character ◦ Datetime ◦ Numeric (you can also search by record number)	Character	
Searches in	◦ Field ◦ Fields	Field	
Searches in related fields	Yes (fully qualified field name must be specified)	Yes	
Index required	No	Yes (ascending order required)	
Leading spaces search-able	Yes (spaces in data or search string treated like a character)	No	Yes (spaces in data or search string treated like a character)
Case-sensitive	Yes		
Partial matching	Yes (search string must appear at the start of the field, Character only)	Yes (search string must appear at the start of the field)	
Quotation marks around search term required	◦ Yes (for Character) ◦ No (for Numeric) ◦ Optional (for record number) ◦ backquotes (for Datetime)	No (search term must not be enclosed in quotation marks, unless the quotation marks are part of the data)	Yes
Affected by Exact Char-acter Comparisons option	Yes	No	

	Locate Record / Locate If	Find Literal	Seek Expression
(SET EXACT ON/OFF)			
Expressions supported	Yes	No	Yes
Additional remarks	The **Locate Record** and the **Locate If** operations in the **Search** dialog box, and the LOCATE RECORD/LOCATE command, are identical.	The **Find Literal** operation in the **Search** dialog box, and the FIND command, are identical.	The **Seek Expression** operation in the **Search** dialog box, and the SEEK command, are identical.

Search and filter using Analytics functions

You can use Analytics functions to perform powerful and effective searching and filtering of data in tables.

To use a function to search or filter, you create a filter in the Filter text box at the top of the table view. The filter uses one of the Analytics functions explained below.

For example, this filter uses the `FINDMULTI( )` function to isolate all records that contain at least one of the search terms anywhere in the record:

Guidelines for searching or filtering using functions

Field names	When you specify a field name to search in, it must be the physical field name in the table layout, not the display name in the table view. **Tip** To see the physical field name, right-click a column header in the table view and select **Properties**.
Quotation marks	Text search terms must be enclosed in "quotation marks".
Back quotes	Datetime search terms must be enclosed in `back quotes`.
Datetime format	◦ Datetime search terms must use YYYYMMDD or YYMMDD format. ◦ Any time portion must use hhmmss format, and be preceded a single blank space, the letter 't', or the letter 'T'. For example: `` `t183000` `` ◦ Do not use any separators such as slashes (/) or colons (:) between the individual components of dates or times.
Related fields	To search in a related field you must specify the fully qualified field name: *table name.field name*.
Function rules	Each function has specific rules that govern how it works - things like supported data types, and case-sensitivity.

	For a high-level comparison of the rules governing Analytics search functions, see "A comparison of Analytics search functions" on page 1183. For detailed information about any function, click the linked function name below.

Types of searches

You can use a function to search or filter text, numeric, or datetime data. However, you need to use the right function for the type of data that you are searching or filtering:

- **Data type supported by a function** – Functions are designed to work with a specific data type, or in some cases they can work with more than one data type.

  For example, you can use the ISBLANK( ) function with text data (character data) but not with numeric or datetime data. You can use the MATCH( ) or BETWEEN( ) functions with character, numeric, or datetime data.

- **Data type of the data** – You need to be aware of the data type of the data that you are searching or filtering and use a function appropriate for the data type. Numbers and dates typically have a numeric or datetime data type. However, they may be using a character data type.

> **Note**
> You can click any function name below for detailed information about the function.

> **Tip**
> You can copy and paste any of the examples below directly into the Filter text box, and modify the search terms and other inputs to match your data.

Text searches (character data type)

Search for a single text term

Use: "FIND( ) function" on page 2145

Description: The search function with the fewest restrictions. Not case-sensitive. Allows searching entire records in addition to searching an individual field or fields.

Example	Result
`FIND("United Equipment")`	Isolates all records that contain the name “United Equipment” anywhere in the record.
`FIND("Equip")`	Isolates all records that contain the string “Equip” anywhere in the record.

Example	Result
`FIND("United Equipment", Vendor_Name)`	Isolates all records that contain the name “United Equipment” in the Vendor_Name field.
`FIND("United Equipment", Vendor.Vendor_Name)`	Isolates all records that contain the name “United Equipment” in the Vendor_Name field in the related Vendor table.

Search for blank text values

Use: "ISBLANK( ) function" on page 2194

Description: Allows you to search for blank values in a character field.

Example	Result
`ISBLANK(First_Name)`	Isolates all records with a blank First_Name field.

Search for multiple text terms

Use: "FINDMULTI( ) function" on page 2150

Description: The same as FIND( ), but allows specifying multiple search terms.

Example	Result
`FINDMULTI(RECORD, "United Equipment", "Muller Corp.")`	Isolates all records that contain the name “United Equipment” or "Muller Corp." anywhere in the record.
`FINDMULTI(RECORD, "equip", "supp")`	Isolates all records that contain the strings “equip” or "supp" anywhere in the record.
`FINDMULTI(Vendor_Name, "United Equipment", "Muller Corp.")`	Isolates all records that contain the name “United Equipment” or "Muller Corp." in the Vendor_Name field.

Example	Result
`FINDMULTI(Vendor.Vendor_Name, "United Equipment", "Muller Corp.")`	Isolates all records that contain the name “United Equipment” or "Muller Corp." in the Vendor_Name field in the related Vendor table.

Use: "MATCH( ) function" on page 2231

Description: A versatile search function that allows you to search a field for multiple search terms simultaneously, or search multiple fields for the same search term. Also allows you to find matching values in two fields.

Example	Result
`MATCH(Vendor_City, "Phoenix", "Austin", "Los Angeles")`	Isolates all records in which the value in the Vendor_City field exactly matches, or begins with, “Phoenix”, “Austin”, or “Los Angeles”.
`NOT MATCH(Vendor_City, "Phoenix", "Austin", "Los Angeles")`	Isolates all records in which the value in the Vendor_City field does not exactly match, or begin with, “Phoenix”, “Austin”, or “Los Angeles”.
`MATCH(Product_Code, "A", "D", "F")`	Isolates all records that have product codes “A”, “D”, or “F”, or product codes beginning with “A”, “D”, or “F”, in the Product_Code field.
`MATCH(Product_Code, "A", "D", "F")`	Isolates all records that have one-character product codes “A”, “D”, or “F” in the Product_Code field. The **Exact Character Comparisons** option must be on.
Note MATCH( ) examples assume that the **Exact Character Comparisons** option is off, except where noted.	

Search for case-sensitive text terms

Use: "MATCH( ) function" on page 2231

Description: A versatile search function that allows you to search a field for multiple search terms simultaneously, or search multiple fields for the same search term. Also allows you to find matching values in two fields.

Example	Result
`MATCH(Last_Name, "SMITH")`	Isolates all records in which the value in the Last_ Name field is "SMITH", all uppercase.
`MATCH(Last_Name, "smith")`	Isolates all records in which the value in the Last_ Name field is "smith", all lowercase.
`MATCH(Last_Name, "Smith")`	Isolates all records in which the value in the Last_ Name field is "Smith", proper case.

Search for a text term in multiple fields

Use: "MATCH( ) function" on page 2231

Description: A versatile search function that allows you to search a field for multiple search terms simultaneously, or search multiple fields for the same search term. Also allows you to find matching values in two fields.

Example	Result
`MATCH("Phoenix", Vendor_City, City, City_2)`	Isolates all records in which at least one of the values in the Vendor_City, City, or City_2 fields exactly matches, or begins with, “Phoenix”.

Search for matching text terms

Use: "MATCH( ) function" on page 2231

Description: A versatile search function that allows you to search a field for multiple search terms simultaneously, or search multiple fields for the same search term. Also allows you to find matching values in two fields.

Example	Result
`MATCH(Vendor_Address, Employee_Address)`	Isolates all records with identical vendor and employee addresses.

Example	Result
	You may need to use additional functions to standardize the format of vendor and employee addresses.

Search for one or more occurrences of a specific character or substring

Use: "OCCURS( ) function" on page 2263

Description: Allows you to search for one or multiple occurrences of a substring in a character field.

Example	Result
`OCCURS(Invoice_Number, "-") > 1`	Isolates all records in which the invoice number contains 2 or more hyphens.
`OCCURS(Full_Name, ALLTRIM(Last_Name))=1`	Isolates all records in which the value in the Last_Name field appears in the Full_Name field. Including the ALLTRIM( ) function in the expression removes any leading or trailing spaces from the Last_Name field, ensuring that only text values are compared.
`OCCURS(Vendor_Name, "UNITED EQUIPMENT") > 0`	Isolates all records that contain the name “UNITED EQUIPMENT”, in uppercase, in the Vendor_Name field. Unlike the FIND( ) function, the OCCURS( ) function is case sensitive.

Search for a substring starting at a specific character position

Use: "AT( ) function" on page 2045

Description: Allows you to search for a substring, or a subsequent occurrence of the substring, in a character field, and specify the starting byte position of the target substring.

Example	Result
`AT(2, "-", Invoice_Number) > 10`	Isolates all records in which the invoice number contains 2 or more hyphens, and the second hyphen occurs after the tenth character in the string.

Search for text in a range

Use: "BETWEEN( ) function" on page 2048

Description: Allows you to search for text values that fall within a range.

Example	Result
`BETWEEN(Last_Name, "C", "K")`	Isolates all records in which the value in the Last_ Name field begins with one of the letters from “C” to “K”, inclusive. The **Exact Character Comparisons** option must be off.

Search for nearly identical text values (fuzzy duplicates)

Use: "ISFUZZYDUP( ) function" on page 2198

Description: Allows you to search for nearly identical values (fuzzy duplicates), as well as identical values. Not case-sensitive.

Use: "LEVDIST( ) function" on page 2214

Description: Similar to ISFUZZYDUP( ), but case-sensitive by default.

Example	Result
`ISFUZZYDUP(Last_Name, "Braun", 2)`	Isolates all records with the name “Braun”, or fuzzy duplicates of the name “Braun”, in the Last_ Name field. The Levenshtein distance (degree of fuzziness), set to 2 in this example, can be increased or decreased.
`LEVDIST(TRIM(Last_Name), "Braun") < 3`	Isolates all records with the name “Braun”, or fuzzy duplicates of the name “Braun”, in the Last_ Name field. The Levenshtein distance (degree of fuzziness), set to < 3 in this example, can be increased or decreased. Including the TRIM( ) function in the expression removes any trailing spaces from the last name field, ensuring that only text values are compared.

Search for a basic pattern

Use: "MAP( ) function" on page 2224

Description: Allows you to search using wildcard characters, literal characters, or a mix of both.

Example	Result
`MAP(Invoice_Number, "XX99999")`	Isolates all records with invoice numbers that consist of, or that start with, two letters followed by five numbers.
`MAP(Invoice_Number, "AB12345")`	Isolates all records with invoice numbers that are exactly “AB12345”, or that start with “AB12345”.
`MAP(Invoice_Number, "AB99999")`	Isolates all records with invoice numbers that consist of, or that start with, “AB” followed by five numbers.
`NOT MAP(SSN, "999-99-9999")`	Isolates all records that do not match the standard format of social security numbers in the SSN field.

Search for a more complicated pattern

Use: "REGEXFIND( ) function" on page 2327

Description: The most powerful and flexible search function. Allows you to search using regular expressions that combine literal characters and metacharacters. Can be more complicated to use than other search functions.

Example	Result
`REGEXFIND(Vendor_City, "Phoenix|Austin|Los Angeles")`	Isolates all records in which the value in the Vendor_City field contains “Phoenix”, “Austin”, or “Los Angeles”.
`REGEXFIND(Product_Code, "\b\d{3}-[a-zA-Z]{6}\b")`	Isolates all records with a product code that starts with 3 numbers, followed by a hyphen and 6 letters.
`REGEXFIND(Product_Code, "\b\d{3,}-[a-zA-Z]{6}")`	Isolates all records with a product code that starts with 3 or more numbers, followed by a hyphen and 6 or more letters.

Numeric searches

Search for a number

Use: "MATCH( ) function" on page 2231

Description: A versatile search function that allows you to search a field for multiple search terms simultaneously, or search multiple fields for the same search term. Also allows you to find matching values in two fields.

Example	Result
`MATCH(Invoice_Amount,154.00)`	Isolates all records with an invoice amount of $154.00.
`MATCH(Invoice_Amount,154.00, 522.00)`	Isolates all records with an invoice amount of $154.00 or $522.00.
`NOT MATCH(Inventory_Value_at_Cost, Cost_x_Quantity)`	Isolates all records with different amounts in the Inventory_Value_at_Cost field and the computed Cost_x_Quantity field.

Search for numbers in a range

Use: "BETWEEN( ) function" on page 2048

Description: Allows you to search for numeric values that fall within a range.

Example	Result
`BETWEEN(Invoice_Amount, 1000, 5000)`	Isolates all records with an invoice amount from $1000 to $5000, inclusive.

Search for a number throughout an entire table

Use: "FIND( ) function" on page 2145

Description: Allows searching entire records in addition to searching an individual field or fields.

Use: "FINDMULTI( ) function" on page 2150

Description: The same as FIND( ), but allows specifying multiple search terms.

> **Note**
>
> Using the FIND( ) or FINDMULTI( ) functions to search for a numeric value can be tricky. The functions search the exact characters in the source data file (.fil), which can be presented differently in the table view.
>
> If search results seem inconsistent to you, examine the source data in the **Table Layout** dialog box.

Example	Result
`FIND("154.00")`	Isolates all records that contain the exact characters **154.00** anywhere in the record in the source data file.

Datetime searches

Search for a datetime value

Use: "MATCH( ) function" on page 2231

Description: A versatile search function that allows you to search a field for multiple search terms simultaneously, or search multiple fields for the same search term. Also allows you to find matching values in two fields.

Example	Result
`MATCH(Invoice_Date, `20170731`)`	Isolates all records with an invoice date of 31 Jul 2017.
`MATCH(Invoice_Date, `20170731`, `20170831`, `20170930`)`	Isolates all records with an invoice dated the last day of the month in each month of the third quarter.

Search for blank or invalid date values

Use: "VERIFY( ) function" on page 2441

Description: Allows you to search for blank or invalid values in a date field.

Example	Result
`NOT VERIFY(Invoice_Date)`	Isolates all records with a blank or invalid date in the Invoice_Date field.

Search for datetime values in a range

Use: "BETWEEN( ) function" on page 2048

Description: Allows you to search for datetime values that fall within a range.

Example	Result
`BETWEEN(Invoice_Date, `20140930`, `20141030`)`	Isolates all records with an invoice date from 30 Sep 2014 to 30 Oct 2014, inclusive.
`NOT BETWEEN(Invoice_Date, `20140930`, `20141030`)`	Isolates all records with an invoice date that does not fall between 30 Sep 2014 and 30 Oct 2014, inclusive.

Search for a datetime value throughout an entire table

Use: "FIND( ) function" on page 2145

Description: Allows searching entire records in addition to searching an individual field or fields.

Use: "FINDMULTI( ) function" on page 2150

Description: The same as FIND( ), but allows specifying multiple search terms.

> **Note**
>
> Using the FIND( ) or FINDMULTI( ) functions to search for a datetime value can be tricky. The functions search the exact characters in the source data file (.fil), which can be presented differently in the table view.
>
> If search results seem inconsistent to you, examine the source data in the **Table Layout** dialog box.

Example	Result
`FINDMULTI(RECORD, "31/07/2017", "31/08/2017")`	Isolates all records that contain the exact characters **31/07/2017** or **31/08/2017** anywhere in the record in the source data file. The normal restriction regarding datetime format (YYYYMMDD, YYMMDD, hhmmss, hhmm) does not apply when using FIND( ) or FINDMULTI ( ) to search for a datetime value.

A comparison of Analytics search functions

The tables below provide a high-level comparison of Analytics search functions. As you construct search expressions in Analytics it can be useful to know how the specific rules that govern each function may differ.

Data types when searching

Show me more

Supported data types	Function
Character	AT( ) FIND( ) FINDMULTI( ) ISFUZZYDUP( ) LEVDIST( ) MAP( ) OCCURS( ) REGEXFIND( )
Character Datetime Numeric	BETWEEN( ) MATCH( )

Search locations (field, fields, record)

Show me more

Supported search locations	Function
Single field	BETWEEN( ) ISFUZZYDUP( ) LEVDIST( )
One or more fields	AT( ) MAP( ) MATCH( ) OCCURS( ) REGEXFIND( )
One or more fields Record	FIND( ) FINDMULTI( )

Leading spaces searchable

Show me more

Leading spaces searchable	Function
Yes Leading spaces in data can optionally be matched in search string	AT( ) BETWEEN( ) FIND( ) FINDMULTI( ) OCCURS( )
Yes Leading spaces in data must be exactly matched in search string	MAP( ) MATCH( )
Yes Spaces in data or search string treated like a character	ISFUZZYDUP( ) LEVDIST( ) REGEXFIND( )

Case-sensitivity

Show me more

Function is case-sensitive	Function
Yes	AT( ) BETWEEN( ) MAP( ) (literal characters) MATCH( ) OCCURS( ) REGEXFIND( )
No	FIND( ) FINDMULTI( ) ISFUZZYDUP( ) MAP( ) (wildcard characters)
Optional	LEVDIST( )

Partial matching

Show me more

Partial matching supported	Function
Yes	AT( )

Partial matching supported	Function
Search string can appear anywhere in the field	FIND( ) FINDMULTI( ) OCCURS( ) REGEXFIND( )
Yes Search string must appear at the start of the field, character data type only	BETWEEN( ) MATCH( )
Yes Search string must be same length as data value, or shorter	MAP( )
Yes	ISFUZZYDUP( ) LEVDIST( )

Multiple search terms

Show me more

Multiple search terms supported	Function
Yes	FINDMULTI( ) MATCH( ) REGEXFIND( )
No	AT( ) BETWEEN( ) FIND( ) ISFUZZYDUP( ) LEVDIST( ) MAP( ) OCCURS( )

Affected by Exact Character Comparisons option (SET EXACT ON/OFF)

Show me more

Affected by Exact Character Comparisons option (SET EXACT ON/OFF)	Function
Yes	BETWEEN( ) MATCH( )
No	AT( ) FIND( ) FINDMULTI( ) ISFUZZYDUP( ) LEVDIST( ) MAP( ) OCCURS( ) REGEXFIND( )

Testing sequential order

Testing sequential order (the Examine Sequence option) allows you to verify whether data has already been sorted or indexed, or whether it needs to be before you perform certain analytical tests, or data combining operations.

Several tests and operations in Analytics require that data is in sequential order for the results to be valid, or for the operation to execute successfully. Instead of unnecessarily sorting or indexing a table, you can first test it to find out if sorting or indexing is required. Testing first can save time, as sorting especially can require considerable time and system resources with large tables.

You can test the sequential order of character, numeric, datetime, or computed fields, or a combination of fields and data types if the data is sorted or indexed by more than one field.

> **Note**
> Sequentially ordered data does not mean that the data has no gaps. For example, the numeric series (1, 3, 5) is sequentially ordered. Testing for gaps is a different operation. For more information, see "Testing for gaps" on page 1195.

Testing sequential order does not sort records

Testing sequential order does not sequentially order or sort records, or in any way modify the order of records in the table being tested. It verifies whether or not specified fields in a table are currently sequentially ordered, and reports any errors of sequence. To sort or sequentially order records requires performing separate sorting or indexing operations.

Testing for out-of-sequence items

Testing sequential order also allows you to identify out-of-sequence items in data that should have an inherent sequential order, such as invoice or check numbers, indicating possible irregularities. For example, you could sort the invoice data for a particular vendor by date, and then test the sequential order of the invoice numbers. Out-of-sequence invoice numbers could warrant further scrutiny.

The sort sequence for testing character data

The sequential order of character fields is tested against whatever **sort sequence** is specified for character data in the **Sort Order** option (**Tools > Options > Table**). Typically the default sort sequence is specified (0,1,2... A,B,C...) unless you have changed it. Some sorting subtleties exist in the non-Unicode edition of Analytics – for example, the default sort sequence dictates that all uppercase alpha characters are sorted before all lowercase alpha characters.

How sequence errors are reported

The Examine Sequence option compares the first value in a column against the second value, the second value against the third, and so on, progressing down the column by comparing paired values. A sequence error is reported if a pair of values constitutes a break in the sequence.

Following a break, the sequence begins anew, using the second of the paired values as the new starting point. Any values after the break that are out of sequence when compared to values before the break are not reported as sequence errors. For example, when testing the following column of values for ascending order, Analytics reports two sequence errors (4, 1), not five (4, 4, 5, 1, 2).

```
1
3
6
4    sequence error
4
5
6
9
1    sequence error
2
```

Testing nested sort orders

If you are testing the sequential order of two or more fields in combination, for the results to be valid, you need to select the fields for testing in the same order of priority as the sort order or index order priority – primary key field, secondary key field, and so on. You also need to match the direction of the sequential order – ascending or descending – for each field.

Valid and invalid results when testing nested sort orders

The examples below demonstrate results that are valid or invalid based on whether the testing order matches the nested sort order and the direction of the values in the **Date** and **Amount** key fields.

Date (primary key field, ascending)	Amount (secondary key field, nested, descending)
15 Jan 2011	$2300.00
15 Jan 2011	$1200.00
15 Jan 2011	$600.00
16 Jan 2011	$900.00
16 Jan 2011	$100.00
17 Jan 2011	$4700.00
17 Jan 2011	$900.00
17 Jan 2011	$500.00

Valid result

Returns 0 sequence errors:

```
SEQUENCE ON Date Amount D
```

The sequence test uses the same order of priority and direction as the fields being tested.

Invalid result

Returns 2 sequence errors:

```
SEQUENCE ON Amount D Date
```

Record Number	Amount	Date
4	900	01/16/2011
6	4,700	01/17/2011

The sequence test uses a different order of priority from the fields being tested, and treats the **Amount** field as unnested.

Invalid result

Returns 5 sequence errors:

```
SEQUENCE ON Date Amount
```

Record Number	Date	Amount
2	01/15/2011	1,200
3	01/15/2011	600
5	01/16/2011	100
7	01/17/2011	900
8	01/17/2011	500

The sequence test uses a different direction from one of the fields being tested, and treats the **Amount** field as sorted ascending.

Steps

You can use the Examine Sequence option to determine if one or more fields in the active table are ordered sequentially, or to identify out-of-sequence items.

> **Note**
> Ensure that a quick sort is not currently applied to the active table. The view must display the actual physical order of the underlying Analytics table for the Examine Sequence option to provide valid results.

Show me how

1. Select **Analyze > Sequence**.
2. On the **Main** tab, do one of the following:
   - Select the field(s) to test from the **Sequence On** list.
   - Click **Sequence On** to select the field(s), or to create an expression.

If you select more than one field, the order in which you select the fields dictates the testing priority. The records are tested by the first field you select, and if there are multiple sequential occurrences of the same value in the first field, the records within the group are then tested by the second field you select, and so on. If you do not select additional fields, records within a group are not subject to any secondary testing.

> **Note**
> When testing tables sorted or indexed by more than one field (nested sorting or indexing), testing priority needs to match sorting or indexing priority (primary key field, secondary key field, and so on) for results to be valid.

The order in which you select the fields is the order in which the columns appear in the results.

3. If you clicked **Sequence On**, you can optionally specify a descending sort order for one or more selected fields by clicking the sort arrow  (the default is ascending).

> **Note**
> When testing fields that have been previously sorted or indexed, the direction of the sort order you specify – ascending or descending – must match the direction of the field you are testing for results to be valid.

4. If there are records in the current view that you want to exclude from processing, enter a condition in the **If** text box, or click **If** to create an IF statement using the **Expression Builder**.

> **Note**
> The **If** condition is evaluated against only the records remaining in a table after any scope options have been applied (**First**, **Next**, **While**).

The IF statement considers all records in the view and filters out those that do not meet the specified condition.

5. Click the **Output** tab.
6. Select the appropriate output option in the **To** panel:
   - **Screen** – Select this option to display the results in the Analytics display area.

> **Tip**
> You can click any linked result value in the display area to drill down to the associated record or records in the source table.

   If the output table contains a large number of records, it is faster and more useful to save the results to a file than to display the results on the screen.

   - **Print** – Select this option to send the results to the default printer.
   - **Graph** – Select this option to create a graph of the results and display it in the Analytics display area.
   - **File** – Select this option to save or append the results to a text file. The file is saved outside Analytics.

> **Note**
> Output options that do not apply to a particular analytical operation are disabled.

7. If you selected **File** as the output type, specify the following information in the **As** panel:
   - **File Type** – **ASCII Text File** or **Unicode Text file** (depending on which edition of Analytics you are using) is the only option. Saves the results to a new text file, or appends the results to an existing text file.
   - **Name** – Enter a file name in the **Name** text box. Or click **Name** and enter the file name, or select an existing file in the **Save** or **Save File As** dialog box to overwrite or append to the file. If Analytics prefills a file name, you can accept the prefilled name, or change it.

     You can also specify an absolute or relative file path, or navigate to a different folder, to save or append the file in a location other than the project location. For example: `C:\Results\Output.txt` or `Results\Output.txt`.
   - **Local** – Disabled and selected. Saving the file locally is the only option.
8. Depending on the output type, you can optionally specify a **Header** and/or a **Footer** in the text box(es).

   Headers and footers are centered by default. Type a left angle bracket (<) before the header or footer text to left align the text. Click **Header** or **Footer** to enter a header or footer of more than one line. Alternatively, you can enter a semi-colon (;) as a line-break character in the header or footer text box. Left aligning multiple lines requires a left angle bracket at the beginning of each line.
9. Click the **More** tab.
10. Select the appropriate option in the **Scope** panel:
    - **All**
    - **First**
    - **Next**
    - **While**

Show me more

All	This option is selected by default. Leave it selected to specify that all records in the view are processed.
First	Select this option and enter a number in the text box to start processing at the first record in the view and include only the specified number of records.
Next	Select this option and enter a number in the text box to start processing at the currently selected record in the view and include only the specified number of records. The actual record number in the leftmost column must be selected, not data in the row.
While	Select this option to use a WHILE statement to limit the processing of records in the view based on a particular criterion or set of criteria. You can enter a condition in the **While** text box, or click **While** to create a WHILE statement using the **Expression Builder**. A WHILE statement allows records in the view to be processed only while the specified condition evaluates to true. As soon as the condition evaluates to false, the processing terminates, and no further records are considered. You can use the **While** option in conjunction with the **All**, **First**, or **Next** options. Record processing stops as soon as one limit is reached.

> **Note**
> The number of records specified in the **First** or **Next** options references either the physical or the indexed order of records in a table, and disregards any filtering or quick sorting applied to the view. However, results of analytical operations respect any filtering.
> If a view is quick sorted, **Next** behaves like **First**.

11. In the **Error Limit** field, specify the maximum number of non-sequential items to list, or keep the default of 10.

    If the limit is reached, Analytics stops processing and outputs the non-sequential items to that point.The **Error Limit** number applies to the combined number of errors across all fields being tested. It is not a per-field limit.

12. If you selected **File** as the output type, and want to append the output results to the end of an existing text file, select **Append To Existing File**.
13. Click **OK**.
14. If the overwrite prompt appears, select the appropriate option.

Testing for gaps

Gaps in sequentially ordered numeric or datetime fields could indicate a data file is incomplete. You can test for gaps in sequentially ordered values in a field, and identify one or more gaps or missing items, if they exist.

For results to be valid, the field being tested must be in sequential order prior to testing. You can sort a field in advance, or use the **Presort** option during the gaps test.

You can test numeric or datetime fields, or numbers in character fields. You can test only one field at a time.

List gap ranges or list missing items

You have two options when outputting the results of gaps testing:

- **List Gap Ranges**
- **List Missing Items**

List Gap Ranges

This option identifies the start and end points of a gap, and the total number of missing items in the gap.

The values that identify the start and end points of the gap are not missing items themselves. They are the sequential values that appear immediately before and immediately after the gap – labeled **Gap Start (Exclusive)** and **Gap End (Exclusive)**. For example, check numbers 12345 and 12350 would identify a gap with four missing items between those two numbers.

List Missing Items

This option lists the individual missing items in a gap. For example, check numbers 12346, 12347, 12348, and 12349.

These missing items are calculated values and do not actually appear in the data being tested. When using this option, the **Maximum Missing Items** field allows you to specify the maximum number of missing items to list individually for each gap, which is useful if a gap is large. If the maximum is exceeded, Analytics uses the range method of identifying gaps instead, with the one difference that the values identifying the start and end points of the gap are the first and last missing items, sequentially – labeled **Gap Start (Inclusive)** and **Gap End (Inclusive)**.

> **Note**
> The number you specify in **Maximum Missing Items** applies on a per-gap basis. It does not limit the total number of missing item results across a data set, whether listed individually or by range.

When using the missing items method, the results can contain a mix of individual missing items and ranges depending on the value in the **Maximum Missing Items** field and the size of the different gaps.

Testing numeric data for gaps

When you test numeric data for gaps, the number of decimal places in the data governs the allowable interval in the data:

- **numeric data contains only integers (no decimal portion)** – the allowable interval is 1

  An interval greater than 1 is a gap. For gaps reported as ranges, the number of missing items is the number of missing integers.

- **numeric data contains decimal places** – the allowable interval is equivalent to the smallest decimal interval

  For example, if a numeric field has two decimal places, the allowable interval is 0.01. An interval greater than the smallest decimal interval is a gap. For gaps reported as ranges, the number of missing items is the number of missing decimal intervals.

Examples of testing numeric data for gaps

In the first example the numeric data contains only integers. The allowable interval is 1.

Test values	Missing items	Number of missing items
-2 -1 0 1 4 5 15	2 3 6 (to) 14 (Inclusive)	1 (integer) 1 (integer) 9 (integers)

In the second example the numeric data contains two decimal places. The allowable interval is 0.01.

Test values	Missing items	Number of missing items
4.24 4.25 4.26 4.29 4.30 5.00	4.27 4.28 4.31 (to) 4.99 (Inclusive)	1 (0.01 interval) 1 (0.01 interval) 69 (0.01 intervals)

Testing datetime data for gaps

You can test date, datetime, or time data for gaps:

- **The allowable interval in date fields is one day**

  An interval greater than one day is a gap. For gaps reported as ranges, the number of missing items is the number of missing days.

- **The allowable interval in datetime or time fields is one second**

  An interval greater than one second is a gap. For gaps reported as ranges, the number of missing items is the number of missing seconds. So a gap of one hour would be reported as a range with 3,600 missing items, and a gap of one day would be reported as a range with 86,400 missing items.

Examples of testing dates and datetimes for gaps

In the first example the data contains only dates. The allowable interval is one day.

Test values	Missing items	Number of missing items
27 Dec 2014 28 Dec 2014 31 Dec 2014 01 Jan 2015 02 Jan 2015 12 Jan 2015 13 Jan 2015	29 Dec 2014 30 Dec 2014 03 Jan 2015 (to) 11 Jan 2015 (Inclusive)	1 (day) 1 (day) 9 (days)

In the second example the data contains datetimes. The allowable interval is one second.

Test values	Missing items	Number of missing items
31 Dec 2014 23:59:54 31 Dec 2014 23:59:55 31 Dec 2014 23:59:58 31 Dec 2014 23:59:59 01 Jan 2015 01:00:00 01 Jan 2015 01:00:01 02 Jan 2015 01:00:02	31 Dec 2014 23:59:56 31 Dec 2014 23:59:57 01 Jan 2015 00:00:00 (to) 01 Jan 2015 00:59:59 (Inclusive) 01 Jan 2015 01:00:02 (to) 02 Jan 2015 01:00:01 (Inclusive)	1 (second) 1 (second) 3,600 (seconds) 86,400 (seconds)

Testing numeric data in a character field for gaps

You can test for gaps in numeric data that appears in a character field – for example, check numbers, which are typically formatted as character data.

If letters and numbers appear together in a character field, only the numbers are tested and the letters are ignored.

Examples of testing numbers in a character field for gaps

Note how the alpha prefixes are ignored, and only the numbers are considered.

Test values	Missing items	Number of missing items
A123 C124		0 (character number)
A123 B125	124	1 (character number)

Sorting of character fields can affect gaps testing

Depending on the arrangement of letters and numbers in character field values, anomalies may exist among the results of gaps testing. For example, if some numbers are prefaced with a letter and some

are not, or in the non-Unicode edition of Analytics if some preceding letters are lowercase and some are uppercase, results may not be accurate.

The reason for the inaccuracy is that the inconsistent presence of alpha characters, or the inconsistent case of alpha characters, prevents the numbers being fully sequentially ordered by the **Presort** option. In the table below, 126 and 127, and 124, are not actually missing items, but because of the way the alphanumeric strings are sorted they are returned as missing items.

If you suspect an anomaly exists, perform a separate sort operation on the field in question to reveal the sequence of character field values being tested for gaps. If sequential numeric order is being disrupted by the presence of letters, you can ensure valid results by using an Analytics function such as INCLUDE( ) to strip out the letters before testing for gaps.

Examples of incorrect gaps results

Note how the inconsistent presence of alpha characters, or the inconsistent case of alpha characters, is causing items to be incorrectly reported as missing.

Test values	Missing items	Number of missing items
123 124 125 128 129 A-126 A-127	126 127	1 (character number) 1 (character number)
A-123 a-124 A-125 A-128 A-129 A-126 A-127	124	1 (character number)

Steps

You can test a single field at a time in the active table to detect whether sequentially ordered numbers or datetime values contain any gaps.

Show me how

1. Select **Analyze > Gaps**.
2. On the **Main** tab, do one of the following:
   - Select a field to test from the **Gaps On** list.
   - Click **Gaps On** to select the field, or to create an expression.

   You can test only one field at a time.
3. If you clicked **Gaps On**, you can optionally specify a descending sort order in the output results for the selected field by clicking the sort arrow  (the default is ascending).
4. If there are records in the current view that you want to exclude from processing, enter a condition in the **If** text box, or click **If** to create an IF statement using the **Expression Builder**.

   > **Note**
   > The **If** condition is evaluated against only the records remaining in a table after any scope options have been applied (**First**, **Next**, **While**).

   The IF statement considers all records in the view and filters out those that do not meet the specified condition.
5. If the field is already sorted based on a prior operation, you can optionally deselect **Presort** to save time when testing large tables for gaps.

   If data in the field is not sorted you must leave **Presort** selected to ensure that all gaps are found.

   > **Note**
   > If you deselect **Presort**, the field you select for gaps testing must have been previously sorted for results to be valid. The message **Warning: File out of sequence** accompanies results if you test an unsorted field. If you output results to an Analytics table, the warning message appears in the command log.

6. Do one of the following:
   - Select **List Gap Ranges** to identify the start and end points of gaps, and the total number of individual missing items in a gap.
   - Select **List Missing Items** to list the individual missing items in gaps. In the **Maximum Missing Items** field, specify the maximum number of missing items to list individually for each gap, or keep the default of 5.
7. Click the **Output** tab.
8. Select the appropriate output option in the **To** panel:
   - **Screen** – Select this option to display the results in the Analytics display area.

     > **Tip**
     > You can click any linked result value in the display area to drill down to the associated record or records in the source table.

If the output table contains a large number of records, it is faster and more useful to save the results to a file than to display the results on the screen.

- **Print** – Select this option to send the results to the default printer.
- **Graph** – Select this option to create a graph of the results and display it in the Analytics display area.
- **File** – Select this option to save or append the results to a text file. The file is saved outside Analytics.

> **Note**
> Output options that do not apply to a particular analytical operation are disabled.

9. If you selected **File** as the output type, specify the following information in the **As** panel:
   - **File Type** – **ASCII Text File** or **Unicode Text file** (depending on which edition of Analytics you are using) is the only option. Saves the results to a new text file, or appends the results to an existing text file.
   - **Name** – Enter a file name in the **Name** text box. Or click **Name** and enter the file name, or select an existing file in the **Save** or **Save File As** dialog box to overwrite or append to the file. If Analytics prefills a file name, you can accept the prefilled name, or change it.

     You can also specify an absolute or relative file path, or navigate to a different folder, to save or append the file in a location other than the project location. For example: `C:\Results\Output.txt` or `Results\Output.txt`.
   - **Local** – Disabled and selected. Saving the file locally is the only option.

   > **Note**
   > For output results produced from analysis or processing of AX Server tables, select **Local**. You cannot deselect the **Local** setting to import results tables to AX Server.

10. Depending on the output type, you can optionally specify a **Header** and/or a **Footer** in the text box(es).

    Headers and footers are centered by default. Type a left angle bracket (<) before the header or footer text to left align the text. Click **Header** or **Footer** to enter a header or footer of more than one line. Alternatively, you can enter a semi-colon (;) as a line-break character in the header or footer text box. Left aligning multiple lines requires a left angle bracket at the beginning of each line.
11. Click the **More** tab.
12. Select the appropriate option in the **Scope** panel:
    - **All**
    - **First**
    - **Next**
    - **While**

    Show me more

All	This option is selected by default. Leave it selected to specify that all records in the view are

	processed.
First	Select this option and enter a number in the text box to start processing at the first record in the view and include only the specified number of records.
Next	Select this option and enter a number in the text box to start processing at the currently selected record in the view and include only the specified number of records. The actual record number in the leftmost column must be selected, not data in the row.
While	Select this option to use a WHILE statement to limit the processing of records in the view based on a particular criterion or set of criteria. You can enter a condition in the **While** text box, or click **While** to create a WHILE statement using the **Expression Builder**. A WHILE statement allows records in the view to be processed only while the specified condition evaluates to true. As soon as the condition evaluates to false, the processing terminates, and no further records are considered. You can use the **While** option in conjunction with the **All**, **First**, or **Next** options. Record processing stops as soon as one limit is reached.

> **Note**
> The number of records specified in the **First** or **Next** options references either the physical or the indexed order of records in a table, and disregards any filtering or quick sorting applied to the view. However, results of analytical operations respect any filtering.
> If a view is quick sorted, **Next** behaves like **First**.

13. If you selected **File** as the output type, and want to append the output results to the end of an existing file, do one of the following:
    - Select **Append To Existing File** if you are appending to a text file, or to an Analytics table that you are certain is identical in structure to the output results.
    - Leave **Append To Existing File** deselected if you are appending to an Analytics table and you want Analytics to compare the record lengths of the output results and the existing table. If the record lengths are not identical, the data structure is not identical, and the append will not work correctly.

    > **Note**
    > Leaving **Append To Existing File** deselected is recommended if you are uncertain whether the output results and the existing table have an identical data structure. For more information about appending and data structure, see "Appending output results to an existing table" on page 200.

14. If you selected **File (Analytics Table)** as the output type, select **Use Output Table** if you want the output table to open automatically upon completion of the operation.
15. Click **OK**.

16. If the overwrite prompt appears, select the appropriate option.

    If you are expecting the **Append** option to appear and it does not, click **No** to cancel the operation and see "Appending output results to an existing table" on page 200.

Testing for duplicates

Duplicate values in one or more fields, or duplicate records, can be the result of data entry errors or fraudulent activity such as splitting credit card transactions to avoid scrutiny.

Requirement for unique values

Fields that should never contain duplicates are ones where the values uniquely identify records. For example, an employee table should never have duplicate employee numbers because each number should identify a unique employee.

Valid duplicates

Duplicate values may also be valid. For example, a transaction table could have duplicate customer numbers because of multiple transactions by the same customers.

Different types of duplicates testing

You can use Analytics to test for duplicates in the following ways:

Test scope	Use this test when:
One field	All values in a particular field should be unique, such as employee numbers, or check numbers.
Two or more fields in combination	Uniqueness is not a requirement of any field in isolation, but it is a requirement of certain fields in combination. **Example** In a payroll file covering a year, the employee number field and the pay date field are going to contain numerous duplicates. Employees get paid every two weeks, and multiple employees are paid on the same date. However, an individual employee should appear only once for a particular date. If a duplicate exists across the employee number and pay date fields in combination, an employee may have been paid twice for the same pay period.

Test scope	Use this test when:
All fields in a record	Checking for entire duplicate records, in which every field in a record is duplicated. Entire duplicate records could be the result of data entry error, or other transactional irregularities.

Sorting and duplicates

Generally, you should only test for duplicates using a sorted key field or fields. Duplicate values in a key field are only found if they are immediately adjacent.

If you test for duplicates using an unsorted key field, non-adjacent duplicate values are not reported as duplicates. If two or more clusters of the same duplicate value exist, they are reported as duplicates, but in separate groups.

Depending on your analysis goal, it may make sense to test for duplicates using an unsorted key field. For example, you may want to find only those duplicate values that are immediately adjacent in the source table, and ignore duplicate values that are non-adjacent.

Including the Group Number field in the output table

You have the option of including the **Group Number** field in the duplicates output table. The field assigns a sequentially incremented number to each unique group of duplicates. The ability to reference groups of duplicates by number can be useful when you analyze data in the output table.

Filter duplicates output table by group number

You use several key fields in combination to test an accounts payable table for duplicate records:

- vendor number
- invoice number
- invoice date
- invoice amount

You want to filter the resulting duplicates output table so that only some of the groups of duplicates are subject to additional processing.

To create a filter using the combination of key fields would be laborious. For example:

```
SET FILTER TO ((Vendor_No = "11475") AND (Invoice_No = "8752512") AND
(Invoice_Date = `20191021`) AND (Invoice_Amount = 7125.80)) OR ((Vendor_
No = "12130") AND (Invoice_No = "589134") AND (Invoice_Date =
`20191117`) AND (Invoice_Amount = 10531.71)) OR ((Vendor_No = "13440")
AND (Invoice_No = "5518912") AND (Invoice_Date = `20191015`) AND
(Invoice_Amount = 11068.20))
```

Instead, you achieve the same result by creating a filter based on group number:

```
SET FILTER TO MATCH(GROUP_NUM, 3 , 8, 11)
```

Steps

You can test one or more fields in the active table to detect whether duplicate values or entire duplicate records exist.

Show me how

You can test character, numeric, and datetime fields for duplicates. If letters and numbers appear together in a character field, all alphanumeric characters are tested.

> **Note**
>
> For results to be valid, the field(s) being tested must be in sequential order prior to testing. You can sort the field(s) in advance, or use the **Presort** option during the duplicates test.

Select the fields

1. Open the table that you want to test for duplicates.
2. Select **Analyze > Duplicates**.
3. **To detect duplicates in one or more fields:**
   a. In the **Main** tab, select the field(s) to test from the **Duplicates On** list, or click **Duplicates On** to select the field(s), or to create an expression.

      The order in which you select the fields is the order in which the columns appear in the results. If you are appending results to an existing Analytics table, the column selection and order must be identical to the column selection and order in the existing table.

   b. If you clicked **Duplicates On**, you can optionally specify a descending sort order in the output results for one or more selected fields by clicking the sort arrow  (the default is ascending).
   c. Select one or more **List Fields** to include any additional field(s) in the output results, or click **List Fields** to select the field(s), to **Add All** fields, or to create an expression.

Additional fields can provide useful context for the results. Fields selected for duplicates testing are displayed automatically at the beginning of any result records and do not need to be specifically selected under **List Fields**.

d. Optional. Select **Add Groups** if you want to include the **Group Number** field in the output table.

The **Group Number** field assigns a sequentially incremented number to each unique group of duplicates.

4. **To detect entire duplicate records:**
   a. In the **Main** tab, click **Duplicates On**.
   b. Click **Add All** to add all fields to **Selected Fields**.
   c. Optionally specify a descending sort order in the output results for one or more fields by clicking the sort arrow  (the default is ascending).
   d. Click **OK**.

You do not need to select any fields from the **List Fields** list because all fields in the table are displayed automatically in the result records.

e. Optional. Select **Add Groups** if you want to include the **Group Number** field in the output table.

The **Group Number** field assigns a sequentially incremented number to each unique group of duplicates.

Exclude records from processing (optional)

If there are records in the current view that you want to exclude from processing, enter a condition in the **If** text box, or click **If** to create an IF statement using the **Expression Builder**.

> **Note**
> The **If** condition is evaluated against only the records remaining in a table after any scope options have been applied (**First**, **Next**, **While**).

The IF statement considers all records in the view and filters out those that do not meet the specified condition.

Deselect Presort (optional)

If the field or fields you are testing are already sorted based on a prior operation, you can optionally deselect **Presort** to save time when testing large tables for duplicates.

> **Note**
>
> If you deselect **Presort**, the field or fields you select for duplicates testing must match the field or fields that were previously sorted for results to be valid.
>
> The message **Warning: File out of sequence** accompanies results if there is a mismatch between selected and sorted fields. If you output results to an Analytics table, the warning message appears in the command log.

If data in field(s) is not sorted you must leave **Presort** selected to ensure that all duplicates are found.

Configure the output

1. Click the **Output** tab.
2. Select the appropriate output option in the **To** panel:
   - **Screen** – Select this option to display the results in the Analytics display area.

     > **Tip**
     >
     > You can click any linked result value in the display area to drill down to the associated record or records in the source table.

     If the output table contains a large number of records, it is faster and more useful to save the results to a file than to display the results on the screen.
   - **Print** – Select this option to send the results to the default printer.
   - **Graph** – Select this option to create a graph of the results and display it in the Analytics display area.
   - **File** – Select this option to save or append the results to a text file. The file is saved outside Analytics.

     > **Note**
     >
     > Output options that do not apply to a particular analytical operation are disabled.

3. If you selected **File** as the output type, specify the following information in the **As** panel:
   - **File Type** – Select **Analytics Table** to save the results to a new Analytics table, or append the results to an existing Analytics table. Select **ASCII Text File** or **Unicode Text file** (depending on which edition of Analytics you are using) to save or append the results to a text file.
   - **Name** – Enter a file name in the **Name** text box. Or click **Name** and enter the file name, or select an existing file in the **Save** or **Save File As** dialog box to overwrite or append to the file. If Analytics prefills a file name, you can accept the prefilled name, or change it.

     You can also specify an absolute or relative file path, or navigate to a different folder, to save or append the file in a location other than the project location. For example: `C:\Results\Output.fil` or `Results\Output.fil`.
   - **Local** – Only enabled when connected to a server table and saving or appending the results to an Analytics table. Select **Local** to save the file to the same location as the project, or to

specify a path or navigate to a different local folder. Leave **Local** deselected to save the file to the Prefix folder on a server.

> **Note**
> For output results produced from analysis or processing of AX Server tables, select **Local**. You cannot deselect the **Local** setting to import results tables to AX Server.

4. Depending on the output type, you can optionally specify a **Header** and/or a **Footer** in the text box(es).

   Headers and footers are centered by default. Type a left angle bracket (<) before the header or footer text to left align the text. Click **Header** or **Footer** to enter a header or footer of more than one line. Alternatively, you can enter a semi-colon (;) as a line-break character in the header or footer text box. Left aligning multiple lines requires a left angle bracket at the beginning of each line.

Specify the scope of the operation

1. Click the **More** tab.
2. Select the appropriate option in the **Scope** panel:
   - **All**
   - **First**
   - **Next**
   - **While**

Show me more

All	This option is selected by default. Leave it selected to specify that all records in the view are processed.
First	Select this option and enter a number in the text box to start processing at the first record in the view and include only the specified number of records.
Next	Select this option and enter a number in the text box to start processing at the currently selected record in the view and include only the specified number of records. The actual record number in the leftmost column must be selected, not data in the row.
While	Select this option to use a WHILE statement to limit the processing of records in the view based on a particular criterion or set of criteria. You can enter a condition in the **While** text box, or click **While** to create a WHILE statement using the **Expression Builder**. A WHILE statement allows records in the view to be processed only while the specified condition evaluates to true. As soon as the condition evaluates to false, the processing terminates, and no further records are considered. You can use the **While** option in conjunction with the **All**, **First**, or **Next** options. Record processing stops as soon as one limit is reached.

> **Note**
> The number of records specified in the **First** or **Next** options references either the physical or the indexed order of records in a table, and disregards any filtering or quick sorting applied to the view. However, results of analytical operations respect any filtering.
> If a view is quick sorted, **Next** behaves like **First**.

Finalize the settings

1. If you selected **File** as the output type, and want to append the output results to the end of an existing file, do one of the following:
   - Select **Append To Existing File** if you are appending to a text file, or to an Analytics table that you are certain is identical in structure to the output results.
   - Leave **Append To Existing File** deselected if you are appending to an Analytics table and you want Analytics to compare the record lengths of the output results and the existing table. If the record lengths are not identical, the data structure is not identical, and the append will not work correctly.

   > **Note**
   > Leaving **Append To Existing File** deselected is recommended if you are uncertain whether the output results and the existing table have an identical data structure. For more information about appending and data structure, see "Appending output results to an existing table" on page 200.

2. If you selected **File (Analytics Table)** as the output type, select **Use Output Table** if you want the output table to open automatically upon completion of the operation.
3. Click **OK**.

   > **Note**
   > Only the duplicate values or records are displayed, and not the initial occurrence of a value or record, if you do both of the following:
   > - output the results to screen or a text file
   > - include only tested fields in the output results and do not select any additional fields
   >
   > If you output to screen, you can click any value to see the initial occurrence of a value or record along with the duplicates.

4. If the overwrite prompt appears, select the appropriate option.

   If you are expecting the **Append** option to appear and it does not, click **No** to cancel the operation and see "Appending output results to an existing table" on page 200.

Remove duplicates

You can use the summarize operation to remove duplicate values or records from a data set and save the remaining unique values or records to a new Analytics table.

Show me how

Select the fields

1. Open the table that you want to remove duplicates from.
2. Select **Analyze > Summarize**.
3. On the **Main** tab, do one of the following:
   - Select the field or fields that may contain duplicate values from the **Summarize On** list.
   - Click **Summarize On** to select the field or fields, or to create an expression.

   The order in which you select the fields is the order in which the columns appear in the results.

   > **Note**
   >
   > Select the appropriate fields to achieve your required degree of uniqueness.
   >
   > For example, if you want to remove duplicate employee records and you select only the last name field, you risk removing all records for employees who have the same last name, but a different first name. Select both the last name and the first name fields to increase the degree of uniqueness.
   >
   > To remove only perfectly duplicate records, click **Summarize On** and **Add All**.

4. Do not select any **Subtotal Fields**.
5. Optional. Do one of the following:
   - From the **Other Fields** list, select the other field(s) to include in the output results.
   - Click **Other Fields** to select the field(s), or to create an expression.

   > **Note**
   >
   > Select only fields that contain the same value for all records in each summarized group. For more information, see "The Other Fields option" on page 1272.

Deselect Presort (optional)

If the field that may contain duplicate values is already sorted, you can optionally deselect **Presort**. If the data in the field is not sorted, you must leave **Presort** selected to ensure valid results.

Exclude records from processing (optional)

If there are records in the current view that you want to exclude from processing, enter a condition in the **If** text box, or click **If** to create an IF statement using the **Expression Builder**.

> **Note**
> The **If** condition is evaluated against only the records remaining in a table after any scope options have been applied (**First**, **Next**, **While**).

The IF statement considers all records in the view and filters out those that do not meet the specified condition.

Configure the output

1. Click the **Output** tab.
2. In the **To** panel, select **File**.
3. Specify the following information in the **As** panel:
   - **File Type** – **Analytics Table** is the only option. Saves the results to a new Analytics table, or appends the results to an existing Analytics table.
   - **Name** – Enter a table name in the **Name** text box. Or click **Name** and enter the table name, or select an existing table in the **Save** or **Save File As** dialog box to overwrite or append to the table. If Analytics prefills a table name, you can accept the prefilled name, or change it.

     You can also specify an absolute or relative file path, or navigate to a different folder, to save or append the table in a location other than the project location. For example: `C:\Results\No_duplicates.fil` or `Results\No_duplicates.fil`.
   - **Local** – Only enabled when connected to a server table. Select **Local** to save the output table to the same location as the project, or to specify a path or navigate to a different local folder. Leave **Local** deselected to save the output table to the Prefix folder on the Analytics server.

     > **Note**
     > For output results produced from analysis or processing of Analytics Exchange server tables, select **Local**. You cannot use the **Local** setting to import results tables to AX Server.

Specify the scope of the operation

1. Click the **More** tab.
2. Select the appropriate option in the **Scope** panel:
   - **All**
   - **First**

- **Next**
- **While**

Show me more

All	This option is selected by default. Leave it selected to specify that all records in the view are processed.
First	Select this option and enter a number in the text box to start processing at the first record in the view and include only the specified number of records.
Next	Select this option and enter a number in the text box to start processing at the currently selected record in the view and include only the specified number of records. The actual record number in the leftmost column must be selected, not data in the row.
While	Select this option to use a WHILE statement to limit the processing of records in the view based on a particular criterion or set of criteria. You can enter a condition in the **While** text box, or click **While** to create a WHILE statement using the **Expression Builder**. A WHILE statement allows records in the view to be processed only while the specified condition evaluates to true. As soon as the condition evaluates to false, the processing terminates, and no further records are considered. You can use the **While** option in conjunction with the **All**, **First**, or **Next** options. Record processing stops as soon as one limit is reached.

> **Note**
> The number of records specified in the **First** or **Next** options references either the physical or the indexed order of records in a table, and disregards any filtering or quick sorting applied to the view. However, results of analytical operations respect any filtering.
> If a view is quick sorted, **Next** behaves like **First**.

Finalize the settings

1. Select **Use Output Table** if you want the output table to open automatically upon completion of the operation.
2. If you want to append the output results to the end of an existing Analytics table, do one of the following:
   - Select **Append To Existing File** if you are certain the output results and the existing table are identical in structure.
   - Leave **Append To Existing File** deselected if you want Analytics to compare the record lengths of the output results and the existing table. If the record lengths are not identical, the data structure is not identical, and the append will not work correctly.

> **Note**
>
> Leaving **Append To Existing File** deselected is recommended if you are uncertain whether the output results and the existing table have an identical data structure. For more information about appending and data structure, see "Appending output results to an existing table" on page 200.

3. Click **OK**.
4. If the overwrite prompt appears, select the appropriate option.

   If you are expecting the **Append** option to appear and it does not, click **No** to cancel the operation and see "Appending output results to an existing table" on page 200.

Fuzzy duplicates analysis

Fuzzy duplicates are nearly identical character values that may refer to the same real-world entity. For example, the following four values may all be the same company:

- Intercity Couriers
- Inter-city Couriers
- Intercity Couriers Inc.
- Intrecity Couriers

Common causes of fuzzy duplicates are data entry errors such as typos and misspellings, differing methods of formatting data, and differing data entry conventions. The intentional creation of nearly identical values may indicate fraud. Fuzzy duplicates impede data analysis, which relies upon data referencing real-world entities in a consistent manner.

Fuzzy duplicates versus fuzzy join

The fuzzy duplicates feature analyzes values in a single field in a single Analytics table. To use fuzzy matching to combine fields from two Analytics tables into a new, third table, see "Fuzzy join" on page 913.

How it works

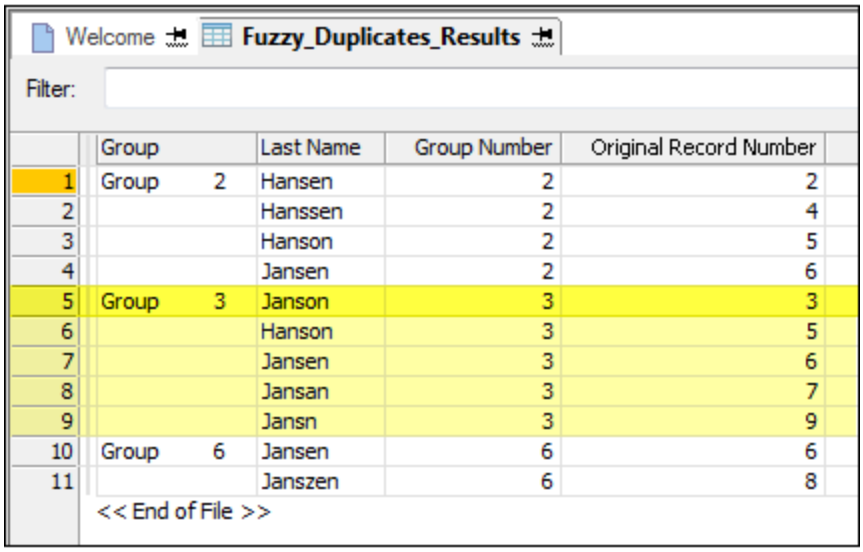
The fuzzy duplicates feature in Analytics lets you test a specific character field in a table to identify any fuzzy duplicates contained by the field. The output results group fuzzy duplicates based on a degree of difference that you specify. By adjusting the degree of difference, you can control the number and the size of output groups and the amount of difference between group members.

To confirm whether fuzzy duplicate group members do in fact reference the same real-world entity, you may need to perform additional analysis, such as a duplicates test of fields other than the test field.

> **Note**
>
> Testing for fuzzy duplicates is more involved than identifying exact duplicates. Understanding the settings that control the degree of difference between fuzzy duplicates, and how fuzzy duplicates are grouped in the output results, will help optimize your use of the feature.

Fuzzy duplicate output results

The example below shows the output results produced by testing for fuzzy duplicates in the **Last Name** field of a table.



Welcome | Fuzzy_Duplicates_Results

Filter:

	Group	Last Name	Group Number	Original Record Number
1	Group 2	Hansen	2	2
2		Hanssen	2	4
3		Hanson	2	5
4		Jansen	2	6
5	Group 3	Janson	3	3
6		Hanson	3	5
7		Jansen	3	6
8		Jansan	3	7
9		Jansn	3	9
10	Group 6	Jansen	6	6
11		Janszen	6	8

<< End of File >>

The output results are arranged in groups identified as **2**, **3**, and **6**. The **Original Record Number** of the first fuzzy duplicate in each group is used to identify the group. For example, “Janson” is the name in record number 3 in the original table, and because “Janson” is the first value in the group, based on record sequence in the original table, the group is identified as **Group 3**. For more information, see "How fuzzy duplicates are grouped" on page 1237.

The fuzzy duplicates feature uses character-based comparison

When comparing two values, the fuzzy duplicates feature performs a character-based comparison, not a word-based comparison. The feature treats blanks or spaces between words as characters, and does not differentiate between individual words. Regardless of the number of individual words in a value, the feature treats the value as a single, unbroken string of characters.

The implication of this approach is that some values that appear to be fuzzy duplicates to the human eye may not be included in the output results, based on the nature of the data and the difference settings you specify in the **Fuzzy Duplicates** dialog box.

Example

Consider these names:

- “JW Smith” and “John William Smith”
- “Diamond Tire” and “Diamond Tire & Auto”

The first example could be two versions of the same name, one using initials and the other using spelled-out first and middle names. The second example could be a short version and a longer version of a company name.

Neither of these pairs of names will be returned as fuzzy duplicates unless the difference settings are quite loose, which would have the adverse effect of also returning large numbers of false positives.

The fuzzy duplicates feature processes each pair of names as simply two strings of characters. In each case, because the two strings differ significantly in length, the strings are significantly different from each other when considered at the character level.

For more information, see "How the difference settings work" on page 1232.

Improving the effectiveness of fuzzy duplicate analysis

In addition to using the main fuzzy duplicates feature, you may need to limit the size of the test data set, use fuzzy duplicate helper functions, or concatenate test fields, to achieve your goals.

The table below summarizes the different techniques for improving the effectiveness of fuzzy duplicate analysis.

For more information about the helper functions, see "Fuzzy duplicate helper functions" on page 1223.

Technique	Analytics feature	Details
Limit the size of the test data set	Filters Extracting subsets of data	Reduce execution time by processing only records that are meaningful to your analysis
Sort individual elements in test field values	SORTWORDS( ) function	Reduce the size and increase the precision of results by minimizing the importance of the physical position of individual elements in test values **Note** Although the fuzzy duplicates feature uses character-based comparison, sorting words or elements in test values has the benefit of aligning characters more closely between strings being compared.

Technique	Analytics feature	Details
Remove generic elements from test field values	OMIT( ) function	Reduce the size and increase the precision of results by focusing on just the portion of test values where meaningful differences may occur
Concatenate fields to increase the uniqueness of test values	an Analytics expression using the Add operator (+)	Reduce the size and increase the precision of results by testing values of greater uniqueness, which are produced by concatenating two or more fields
Generate a single, exhaustive list of fuzzy duplicates for a specific value in the fuzzy duplicates output results	ISFUZZYDUP( ) function	Produce a convenient and exhaustive list of fuzzy duplicates for a output value of particular relevance to your analysis goal

Should I sort the test field?

Testing a field for fuzzy duplicates does not require that the field is sorted. Sorting a table by the test field in advance of testing does not increase the effectiveness of the fuzzy duplicates operation in any way. However, you may choose to sort a test field in advance because it can make the output results easier to scan, and the **Fuzzy Duplicates** dialog box does not include the **Presort** option.

> **Note**
> Although sorting test field values does not increase effectiveness, sorting individual elements in field values with multiple elements, such as addresses, can significantly increase effectiveness. For more information, see "Fuzzy duplicate helper functions" on page 1223.

Including exact duplicates

When testing for fuzzy duplicates, you can optionally include exact duplicates in the output results. If you are interested in finding only exact duplicates, use the duplicates feature instead. For more information, see "Testing for duplicates" on page 1204.

Testing for fuzzy duplicates

You can test a character field in the active table to detect whether nearly identical values exist (fuzzy duplicates). You can optionally include identical values (exact duplicates) in the output results as well as nearly identical values.

A message appears in the log if one or more fuzzy duplicate groups in the results reach the maximum size. For more information, see "Controlling the size of fuzzy duplicate results" on page 1228.

Improving the effectiveness of fuzzy duplicates testing

You can significantly improve the effectiveness of fuzzy duplicates testing by incorporating one or more of the following techniques:

- sorting individual elements in test field values
- removing generic elements from test field values
- concatenating test fields

For more information, see "Fuzzy duplicate helper functions" on page 1223, and "Concatenating fields" on page 218.

Reducing execution time and the size of the output results

The fuzzy duplicates feature is processor-intensive, because every value in a test field must be compared with every subsequent value in the field.

If your analysis allows it, use methods such as filtering or extracting subsets of records to limit the size of the data set you test. Smaller data sets reduce overall execution time, and also help control the size of the output results.

Steps

1. Select **Analyze > Fuzzy Duplicates**.
2. On the **Main** tab, do one of the following:
   - Select the field to test from the **Fuzzy Duplicates On** list.
   - Click **Fuzzy Duplicates On** to select the field, or to create an expression.

   > **Tip**
   > Creating an expression is how you concatenate test fields, remove generic elements from test field values, or sort individual elements in test field values. For more information, see "Fuzzy duplicate helper functions" on page 1223, and "Concatenating fields" on page 218.

3. Optional. Select one or more **List Fields** to include any additional field(s) in the results, or click **List Fields** to select the field(s), to **Add All** fields, or to create an expression.

   Additional fields can provide useful context for the results, and can help verify whether fuzzy duplicates reference the same real-world entity.

   > **Note**
   > The field selected for fuzzy duplicates testing is displayed automatically at the beginning of any result records and does not need to be specifically selected under **List Fields**.

4. Specify a **Difference Threshold** to control the amount of difference between fuzzy duplicates.

   The setting is explained below.

5. Do one of the following:
   - Specify a **Difference Percentage** to control the percentage of each fuzzy duplicate that can be different.
   - Deselect **Difference Percentage** to turn it off.

   The setting is explained below.

6. Do one of the following:
   - Specify a **Result Size (%)** to set the maximum size of the results relative to the size of the test field.
   - Deselect **Result Size (%)** to turn it off.

   The setting is explained below.

7. If you want to include exact duplicates as well as fuzzy duplicates in the results, select **Include Exact Duplicates**.

   For more information, see "How fuzzy duplicates are grouped" on page 1237.

8. If there are records in the current view that you want to exclude from processing, enter a condition in the **If** text box, or click **If** to create an IF statement using the **Expression Builder**.

The IF statement considers all records in the view and filters out those that do not meet the specified condition.

9. If you are connected to a server table, do one of the following:
   - Select **Local** to save the output table to the same location as the project, or to specify a path or navigate to a different local folder.
   - Leave **Local** deselected to save the output table to the Prefix folder on a server.

   > **Note**
   > For output results produced from analysis or processing of Analytics Exchange server tables, select **Local**. You cannot deselect the **Local** setting to import results tables to Analytics Exchange.

10. Do one of the following:
    - In the **To** text box, specify the name of the Analytics table that will contain the output results.
    - Click **To** and select an existing table in the **Save** or **Save File As** dialog box to overwrite or append to the table.

    You can also specify an absolute or relative file path, or navigate to a different folder, to save or append the table in a location other than the project location. For example: `C:\Results\Output.fil` or `Results\Output.fil`.

    Regardless of where you save or append the table, it is added to the open project if it is not already in the project.

    If Analytics prefills a table name, you can accept the prefilled name, or change it.

11. Select **Use Output Table** if you want the output table to open automatically upon completion of the operation.
12. Click **OK**.
13. If the overwrite prompt appears, select the appropriate option.

Fuzzy Duplicates dialog box options

The table below provides detailed information about options in the Fuzzy Duplicates dialog box.

Options - Fuzzy Duplicates dialog box	Description
Difference Threshold	The allowable amount of difference between fuzzy duplicates. Specify a number from 1 to 10. Increasing the **Difference Threshold** increases the number of characters that can differ between fuzzy duplicate pairs, which increases the size of the results. For more information, see "How the difference settings work" on page 1232.
Difference Percentage	The percentage of each fuzzy duplicate that can be different. Specify a percentage from 1 to 99. Increasing the **Difference Percentage** increases the percentage of a fuzzy duplicate that can be different, which increases the size of the results.

Options - Fuzzy Duplicates dialog box	Description
	If you turn off **Difference Percentage**, the results do not take into account the percentage of a fuzzy duplicate that is different. The results will be larger than when you use **Difference Percentage** with any setting. For more information, see "How the difference settings work" on page 1232.
Result Size (%)	The maximum size of the results relative to the size of the test field. Specify a percentage from 1 to 1000 (one thousand). This option allows you to automatically terminate the fuzzy duplicates operation if the size of the results grows beyond what you consider useful. For example, for a test field with 50,000 values, a **Result Size (%)** of 1 would terminate processing if the results exceeded 500 fuzzy duplicates. No output table is produced if processing is terminated. If you turn off **Result Size (%)**, Analytics does not impose any limit on the size of the results. **Caution** Turning off **Result Size (%)** can produce an unduly large set of results that takes a very long time to process, or can cause available memory to be exceeded, which terminates processing. Turn off this option only if you are confident that the results will be of a manageable size. For more information, see "Controlling the size of fuzzy duplicate results" on page 1228.

Fuzzy duplicate helper functions

Two Analytics functions help make the fuzzy duplicates feature more effective:

- SORTWORDS( )
- OMIT( )

You can use the two functions separately, or in combination.

A third function, ISFUZZYDUP( ), gives you the option to identify fuzzy duplicates for a specific value, rather than for an entire field.

SORTWORDS function

When using the fuzzy duplicates feature, use the SORTWORDS( ) function to create an expression or a computed field that sorts individual elements in test field values into a sequential order.

Sorting elements, such as the components of an address, reduces the importance of the physical position of elements in fuzzy duplicates comparisons. The resulting improvement in effectiveness allows you to use a much lower **Difference Threshold** and produce a smaller, more focused set of results containing fewer false positives.

For detailed information, see "SORTWORDS( ) function" on page 2378. For more information about the **Difference Threshold**, see "How the difference settings work" on page 1232.

For a video providing an overview of SORTWORDS( ), see Fuzzy Matching Using SORTWORDS() (English only).

Example

The following two values would require a **Difference Threshold** of at least 22 to be included in fuzzy duplicate output results:

- 125 SW 39TH ST, Suite 100
- Suite 100, 125 SW 39TH ST

The maximum allowable **Difference Threshold** is 10, so the fuzzy duplicates feature would never identify the two values as fuzzy duplicates of each other. Although, clearly, they are the same address.

By contrast, if you use SORTWORDS( ) to create an expression or a computed field that sorts the individual address elements, a **Difference Threshold** of only 2 would return the two addresses as fuzzy duplicates of each other:

- 100 125 39TH ST, SW Suite
- 100, 125 39TH ST SW Suite

OMIT function

When using the fuzzy duplicates feature, use the OMIT( ) function to create an expression or a computed field that removes generic elements from test field values.

Removal of elements such as hyphens, commas, and number signs, and words or abbreviations such as "Inc.", "Street", or "St.", focuses the fuzzy duplicates comparisons on just the portion of the test values where a meaningful difference may occur. The resulting improvement in effectiveness allows you to use a much lower **Difference Threshold** and produce a smaller, more focused set of results containing fewer false positives.

For detailed information, see "OMIT( ) function" on page 2268. For more information about the **Difference Threshold**, see "How the difference settings work" on page 1232.

Example

The following two values require a **Difference Threshold** of at least 8 to be included in fuzzy duplicate output results:

- Intercity Couriers Corporation
- Inter-city Couriers Corp.

A **Difference Threshold** of 8 might produce a large, unfocused set of results containing many false positives. However, a lower **Difference Threshold** would allow the two values to escape detection as fuzzy duplicates of each other.

By contrast, if you use OMIT( ) to create an expression or a computed field that removes generic elements such as "Corporation" and "Corp.", a **Difference Threshold** of only 1 would return the two names as fuzzy duplicates of each other:

- Intercity Couriers
- Inter-city Couriers

ISFUZZYDUP function

After using the fuzzy duplicates feature and reviewing the results, you can use the ISFUZZYDUP( ) function to output a single, exhaustive list of fuzzy duplicates for a specific value in the results. You can take this additional step for values that appear to be of particular relevance to your analysis goal.

Exhaustive means that all values within the specified degree of difference of the test value are returned, regardless of their position in the test field relative to the test value.

By design, the fuzzy duplicates feature organizes the output results in non-exhaustive groups. The results, in total, are exhaustive, but the individual groups may or may not be. This approach prevents the output results from becoming very large and unmanageable.

The non-exhaustive groups may be sufficient for the purposes of your analysis. If they are not, you can use ISFUZZYDUP( ) to produce exhaustive results for individual values.

For detailed information, see "ISFUZZYDUP( ) function" on page 2198. For more information about non-exhaustive groups, see "How fuzzy duplicates are grouped" on page 1237.

Working with fuzzy duplicate output results

If you want to see the actual difference threshold (Levenshtein distance) between each group owner and group member in a fuzzy duplicates output table, and the difference percentage that applies to each owner-member pair, you can add computed fields to display these values. Once you have added the computed fields, you can create a nested sort order to rank the output results by their degree of fuzziness.

You need to create three computed fields, and the fields must be created in this order:

- Group owner computed field
- Levenshtein distance computed field
- Difference percentage computed field

To add difference threshold and difference percentage fields:

1. Open the table containing the fuzzy duplicate output results and select **Edit > Table Layout**.
2. Create the group owner computed field by doing the following:
   a. On the **Edit Fields/Expressions** tab, click **Add a New Expression**  .
   b. In the **Name** field, type `Group_Owner`.
   c. In the **Default Value** field, type the physical name of the fuzzy duplicates test field in the output results (for example, vendor_name), or click **f(x)**  to select it in the **Expression Builder**.
   d. In the **If** field, type the following expression: `NOT ISBLANK(GROUP_FL)`.
   e. Select the **Static** field.
   f. Click **Accept Entry**  .
3. Create the Levenshtein distance computed field by doing the following:
   a. Click **Add a New Expression**  .
   b. In the **Name** field, type Lev_Dist.
   c. In the **Default Value** field, type the following expression, or click f(x)  to build the expression in the **Expression Builder**:

      LEVDIST(ALLTRIM(Group_Owner),ALLTRIM(*fuzzy_dup_test_field*),F)

      Replace *fuzzy_dup_test_field* with the actual name of the fuzzy duplicates test field.
   d. Click **Accept Entry**  .
4. Create the difference percentage computed field by doing the following:
   a. Click **Add a New Expression**  .
   b. In the **Name** field, type `Diff_Pct`.
   c. In the **Default Value** field, type the following expression, or click **f(x)**  to build the expression in the **Expression Builder**:

```
100*DEC(Lev_Dist,2)/MINIMUM(LENGTH(ALLTRIM(Group_Owner)), LENGTH
(ALLTRIM(fuzzy_dup_test_field))
```

Replace *fuzzy_dup_test_field* with the actual name of the fuzzy duplicates test field.

d. Click **Accept Entry** and click **Close** to exit the **Table Layout** dialog box.

5. Add the **Lev_Dist** and **Diff_Pct** computed fields to the view.

   The difference threshold (Levenshtein distance) between each group owner and group member, and the difference percentage that applies to each owner-member pair, is now displayed.

   For information about how to add fields to a view, see "Add columns to a view" on page 778.

6. If you want to rank the output results by their degree of fuzziness, do the following:

   a. Extract all fields except the Group field to a new table, and filter out records in which the Group field is not blank.

   The ACLScript syntax for the extract operation appears below.

```
EXTRACT FIELDS Lev_Dist Diff_Pct GROUP_NUM Group_Owner ORIG_REC_NUM
fuzzy_dup_test_field IF ISBLANK(GROUP_FL) TO "Ranked_Fuzzy_Dupes_1"
OPEN
```

Replace *fuzzy_dup_test_field* with the actual name of the fuzzy duplicates test field.

b. Perform a nested sort of the extracted table using **Lev_Dist** as the first sort field and **Diff_Pct** as the second sort field.

The ACLScript syntax for the sort operation appears below.

```
SORT ON Lev_Dist Diff_Pct TO "Ranked_Fuzzy_Dupes_2" OPEN
```

The fuzziness of the output results increases as you go down the table. The **Group Number** field is the original record number of the group owner in each fuzzy duplicate pair, and the **Original Record Number** field is the original record number of the group member in each pair.

For information about how to create a nested sort, see "Sorting on multiple key fields" on page 1128.

Controlling the size of fuzzy duplicate results

Fuzzy duplicate results have the potential to grow very large because the fuzzy duplicates feature uses an algorithm that performs a many-to-many comparison of values in the test field. The comparison, by design, also returns matches more easily than a comparison that requires exact matching.

Depending on the nature of the data and the difference settings you specify, the results can potentially be many times larger than the table being tested. If results get very large in relation to the test table, they may no longer be useful or meaningful, and the majority of the results could be false positives.

Methods for controlling the size of fuzzy duplicate results

You can use one or more of the following methods to control the size of fuzzy duplicate results and reduce the number of false positives that are returned:

- **Use more than one test field** – concatenate test fields to increase the degree of uniqueness of test values.
- **Sort elements in test field values** – use the SORTWORDS( ) function to sort individual elements in test field values into a sequential order, which allows you to use a smaller **Difference Threshold**.
- **Remove generic elements from test field values** – use the OMIT( ) function to remove generic elements from test field values, which allows you to use a smaller **Difference Threshold**.
- **Difference Threshold** – use a small **Difference Threshold** initially (for example, 3 or less), and only increase it if you feel the results are overly restrictive.
- **Difference Percentage** – use the default **Difference Percentage** initially (50), and only increase it if you feel the results are overly restrictive. Do not turn off **Difference Percentage** unless you have a specific reason for doing so.
- **Result Size (%)** – Based on the number of values in the test field, specify a **Result Size (%)** that prevents the results growing to an unmanageable size. **Result Size (%)** sets the maximum size of the results relative to the size of the test field. Do not turn off **Result Size (%)** unless you have a specific reason for doing so.

  > **Note**
  > This setting has no effect on the inclusion or exclusion of false positives.

- **Limit fuzzy duplicate group size** – use the SET command to specify a maximum fuzzy duplicate group size smaller than the default size of 20 – for example, `SET FUZZYGROUPSIZE TO`

10.

> **Note**
> This setting has no effect on the inclusion or exclusion of false positives.

> **Caution**
> Some of the methods itemized above, if set too restrictively, can exclude valid fuzzy duplicates. You may need to try different combinations of settings to find out what works best for a particular data set.
>
> The methods least likely to exclude valid fuzzy duplicates are concatenating test fields, using the SORTWORDS( ) function, and using the OMIT( ) function.

Specifying a maximum result size

Using the **Result Size (%)** option to specify a maximum result size allows you to automatically terminate the fuzzy duplicates operation if the size of the results grows beyond what you consider manageable. No output table is produced if the operation is terminated.

The **Result Size (%)** option is a safety mechanism to prevent extremely long processing times. It has no relation to the validity of the results that are returned. Specifying a large result size limit may simply increase the number of false positives in the results. Conversely, specifying a small result size may cause processing to terminate before all valid fuzzy duplicates are captured.

Choosing an appropriate limit

Choosing an appropriate limit for the result size is a matter of judgment and may require some experimentation. Start with a conservative limit. If the limit is exceeded and processing terminated, you can increase the limit. Once you have a limit that allows processing to complete, examine the results. If they include a large proportion of false positives, the best approach is to use one or more of the "Methods for controlling the size of fuzzy duplicate results" on the previous page.

An optimal result set includes all valid fuzzy duplicates in the test field (true positives) while also minimizing the number of false positives. Achieving an optimal result set typically requires balancing all the fuzzy duplicates settings and helper methods available to you.

Why you can specify a result size limit greater than one hundred percent

By default, the maximum size of the set of results is 10% of the size of the test field. You can specify a different percentage from 1 to 1000 (one thousand). The limit of 1000% is to accommodate the nature of many-to-many matching, and to prevent runaway processing. Many-to-many matching can produce results that are more numerous than the original test data set. However, results that exceed the size of the original test data set may be primarily false positives.

Rounding of the result size calculation

The result size calculation uses rounding to produce only positive integers, and rounds up any number less than 2 to 2, the minimum result size (1 group owner and 1 group member).

Turning off the result size limit

Generally, you should not turn off **Result Size (%)** unless you are confident that the results will be of a manageable size. Running the fuzzy duplicates operation without any limit on the number of results can cause the operation to run for a very long time, or exceed available memory, which terminates processing.

Setting a maximum fuzzy duplicate group size

Using the SET command to specify a maximum fuzzy duplicate group size can be a way of limiting the size of groups that would otherwise contain a large number of false positives. This feature is most useful if you find a setting that limits the size of only some of the groups in the output results. If all or most of the groups reach their maximum size, the setting may be too small, and you may be excluding valid fuzzy duplicates. The other possibility is that the difference settings are not restrictive enough, which is causing the size of the groups to grow larger.

The default maximum group size is 20, and does not include the group owner. You can specify a different maximum from 2 to 100. The specified maximum remains in effect for the duration of the Analytics session.

What happens if a group reaches the maximum size?

If a fuzzy duplicate group reaches the maximum size, any subsequent fuzzy duplicates for the group owner are not detected and do not appear in the group. These excluded fuzzy duplicates may or may not appear in a subsequent group, depending on whether they are part of a subsequent fuzzy duplicate match.

If producing an exhaustive list of fuzzy duplicates for an owner of a group that has reached its maximum size is important to your analysis, you can use the ISFUZZYDUP( ) function for this purpose. For more information, see "Fuzzy duplicate helper functions" on page 1223.

A message appears in the log if one or more groups reach the maximum size. If the number of groups that reach the maximum size is ten or fewer, the groups are individually identified by group number.

Exact duplicates are included in the group size calculation

Exact duplicates are included in the group size calculation even if you have chosen not to include exact duplicates in the results. For example, if a group is identified in the log as having reached the maximum group size of 20 (1 group owner and 20 group members), but only 18 group members appear in the results, at least two exact duplicates for the group owner exist in the test field.

Groups that are composed entirely of exact duplicates are also referenced in the log if they reach the maximum group size, but the groups do not appear in the results if you have chosen not to include exact duplicates.

For more information, see "SET command" on page 1960.

How the difference settings work

Fuzzy duplicates are selected based on the degree of difference you specify, and then grouped in the output results. The degree of difference is a combination of two settings in the **Fuzzy Duplicates** dialog box:

- **Difference Threshold** – controls how much two fuzzy duplicates can differ
- **Difference Percentage** – control what proportion of an individual value can be different

The two settings act as two separate thresholds. Values in the field you are testing must be within the bounds of **both** thresholds to be included in a group of fuzzy duplicates in the results. By adjusting the two settings you can maximize the precision and usefulness of the results.

You can turn off **Difference Percentage**, in which case values only need to be within the bounds of the **Difference Threshold**. You cannot turn off **Difference Threshold**.

Difference Threshold in detail

The **Difference Threshold** is the maximum allowable Levenshtein distance between two values for them to be identified as fuzzy duplicates.

What is Levenshtein distance?

Levenshtein distance is the minimum number of single character edits required to make one value identical to another. The number of required edits is calculated by a computing science algorithm.

Example of Levenshtein distance

The Levenshtein distance between “Smith” and “Smythe” is 2:

- **edit 1** – ‘y’ must be substituted for ‘i’
- **edit 2** – ‘e’ must be inserted

The greater the Levenshtein distance, the greater the difference between two values. A distance of 0 (zero) means two values are identical.

The table below provides examples of various Levenshtein distances. For more information about Levenshtein distance, see LEVDIST( ).

> **Note**
> The Levenshtein algorithm treats blanks or spaces between words as characters.

Value 1	Value 2	Levenshtein distance	Included in results if Difference Threshold set to 3
Smith	Smith	0	Yes (if **Include Exact Duplicates** is checked)
Smith	Smithe	1	Yes
Smith	Smythe	2	Yes
Hanssen	Jansn	3	Yes
Smith	Brown	5	No
Intercity Couriers	Intercity Couriers Inc.	5	No
Diamond Tire	Diamond Tire & Auto	7	No
JW Smith	John William Smith	10	No

Changing the Difference Threshold

Increasing the **Difference Threshold** increases the maximum allowable Levenshtein distance, which increases the size of the results by including values that are more different from one another. You can specify a **Difference Threshold** from 1 to 10.

The upper limit is imposed because increasing the maximum Levenshtein distance beyond a certain point creates a very large set of results that contains primarily false positives.

The lower limit is imposed because entering 0 (zero) would include only exact duplicates. If you are interested in finding only exact duplicates, use the duplicates feature instead.

Difference Percentage in detail

The **Difference Percentage** is the maximum allowable percentage of the shorter of two compared values that can be different for the two values to be identified as fuzzy duplicates.

How is the difference percentage calculated?

Using the Levenshtein distance between each pair of values it compares in the test field, Analytics performs the following internal calculation:

Levenshtein distance / number of characters in the shorter value × 100 = difference percentage

Example of difference percentage

The Levenshtein distance between “Smith” and “Smythe” is 2, and the shorter of the two values is 5 characters long, producing a difference percentage of 40 (2/5 x 100).

If the difference percentage is less than or equal to the specified **Difference Percentage**, the two values are eligible to be included in the results, assuming they are also within the maximum allowable Levenshtein distance of each other (the **Difference Threshold**).

The table below provides examples of various difference percentages.

Value 1 (length)	Value 2 (length)	Levenshtein distance, and difference percentage	Included in results if Difference Percentage set to 50
Smith (5)	Smith (5)	0, 0% (0/5)	Yes (if **Include Exact Duplicates** is checked)
Smith (5)	Smithe (6)	1, 20% (1/5)	Yes
Smith (5)	Smythe (6)	2, 40% (2/5)	Yes
Hanssen (7)	Jansn (5)	3, 60% (3/5)	No
Smith (5)	Brown (5)	5, 100% (5/5)	No
Intercity Couriers (18)	Intercity Couriers Inc. (23)	5, 27.77% (5/18)	Yes
Diamond Tire (12)	Diamond Tire & Auto (19)	7, 58.33% (7/12)	No
JW Smith (8)	John William Smith (18)	10, 125% (10/8)	No

Changing the Difference Percentage

Increasing the **Difference Percentage** increases the size of the results by including values that contain a greater percentage of difference. You can specify a **Difference Percentage** from 1 to 99.

The upper limit is imposed because allowing difference percentages of 100 or greater could include pairs of values that are completely different from each other in the same fuzzy duplicates group in the results. For example, “ABC” and “XYZ” have a Levenshtein distance of 3, and a shorter value length of 3, producing a difference percentage of 100.

The lower limit is imposed because entering 0 (zero) would include only exact duplicates. If you are interested in finding only exact duplicates, use the duplicates feature instead.

Turning off Difference Percentage

You can optionally turn off **Difference Percentage**. If you turn off **Difference Percentage** the results do not take into account the percentage of a value that is different. You may capture some additional valid fuzzy duplicates, such as “JW Smith” and “John William Smith”. However, fuzzy duplicate groups could also include values that are completely different from each other, such as “Smith” and “Brown”. The results will also be larger than when you use **Difference Percentage** with any setting.

How Difference Threshold and Difference Percentage work together

The table below shows how **Difference Threshold** and **Difference Percentage** work together. The compared values that appear in "Difference Threshold in detail" on page 1232 and "Difference Percentage in detail" on page 1233 must now be within the bounds of both thresholds to be included in the results.

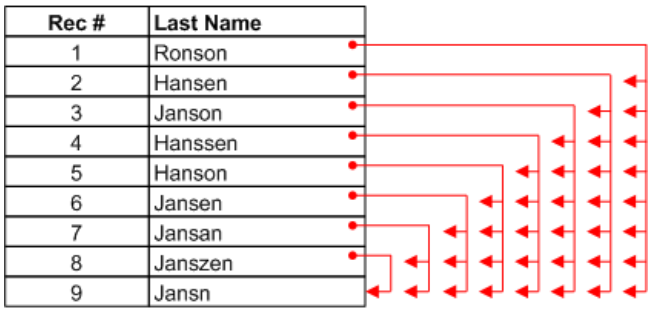
“Hanssen/Jansn” and “Intercity Couriers/Intercity Couriers Inc.” are included if **Difference Threshold** and **Difference Percentage** are considered individually. However, they are not included when the two settings are considered together because they do not fall within the bounds of both thresholds.

Value 1 (length)	Value 2 (length)	Levenshtein distance, and difference percentage	Included in results if Difference Threshold set to 3 and Difference Percentage set to 50
Smith (5)	Smith (5)	0, 0% (0/5)	Yes (if **Include Exact Duplicates** is checked)
Smith (5)	Smithe (6)	1, 20% (1/5)	Yes
Smith (5)	Smythe (6)	2, 40% (2/5)	Yes
Hanssen (7)	Jansn (5)	3, 60% (3/5)	No
Smith (5)	Brown (5)	5, 100% (5/5)	No
Intercity Couriers (18)	Intercity Couriers Inc. (23)	5, 27.77% (5/18)	No
Diamond Tire (12)	Diamond Tire & Auto (19)	7, 58.33% (7/12)	No

Value 1 (length)	Value 2 (length)	Levenshtein distance, and difference percentage	Included in results if Difference Threshold set to 3 and Difference Percentage set to 50
JW Smith (8)	John William Smith (18)	10, 125% (10/8)	No

How fuzzy duplicates are grouped

When processing data, the fuzzy duplicates operation moves sequentially down the test field. The operation compares the first value in the field with every subsequent value, then it compares the second value in the field with every subsequent value, and so on, looping through the field until all the values have been compared with every subsequent value. It does not compare values with previous values.



Rec #	Last Name
1	Ronson
2	Hansen
3	Janson
4	Hanssen
5	Hanson
6	Jansen
7	Jansan
8	Janszen
9	Jansn

With each comparison, the operation determines whether the two compared values are fuzzy duplicates based on the difference settings you specified. (For information about the difference settings, see "How the difference settings work" on page 1232.) If the two values are fuzzy duplicates, they are placed together in a group. Redundant matches are suppressed (explained later in this topic). The results of a fuzzy duplicates operation can contain multiple groups.

Group owner and group members

The first fuzzy duplicate in a group is the controlling value or 'the owner' of the group based solely on the fact that among the group members it appears first in the field you are testing. A test field containing the same data but sorted differently would produce different group owners, and differently constituted groups.

The group is identified using the record number of the group owner. The example below shows the results of testing a **Last Name** field. "Janson" forms a group (based on the difference settings), and "Janson" is record number 3 in the original table, so the group becomes **Group 3**.

Welcome | Fuzzy_Duplicates_Results

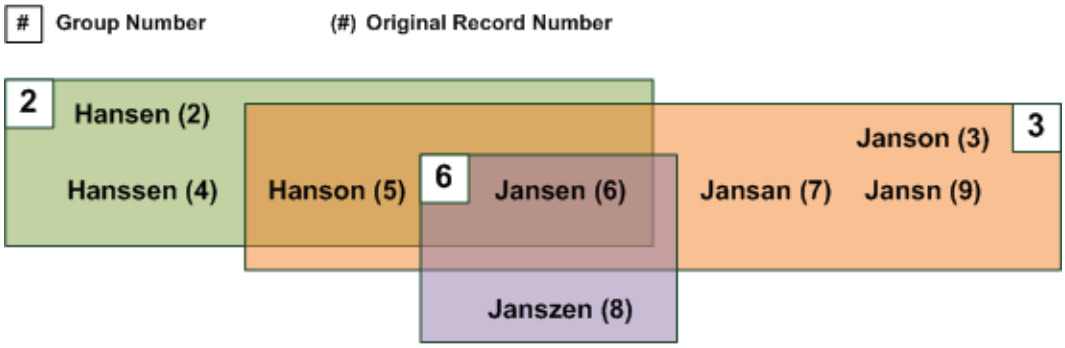
Filter:

	Group	Last Name	Group Number	Original Record Number
1	Group 2	Hansen	2	2
2		Hanssen	2	4
3		Hanson	2	5
4		Jansen	2	6
5	Group 3	Janson	3	3
6		Hanson	3	5
7		Jansen	3	6
8		Jansan	3	7
9		Jansn	3	9
10	Group 6	Jansen	6	6
11		Janszen	6	8
	<< End of File >>			

The group owner is not necessarily the correct value

The group owner is not necessarily the 'correct' or canonical value. It is simply the value from which the degree of difference you have specified is measured or calculated in the process of group formation. All the members of a group are within the specified degree of difference of the group owner. The members may or may not be within the specified degree of difference of one another.

The diagram below provides a visual representation of the results in the output table above. The **Difference Threshold** is 1, which means that group members can differ from the group owner by a maximum of one (1) character. Note that some of the fuzzy duplicates appear in more than one group.



Exhaustive versus non-exhaustive results

To prevent results from becoming unmanageably large, the fuzzy duplicates feature is designed to produce groups that are **non-exhaustive**. Non-exhaustive means that individual fuzzy duplicate

groups may not contain all the fuzzy duplicates in a test field that are within the specified degree of difference of the group owner. However, if a group owner is a fuzzy duplicate of another value in the test field, the two values will appear together in a group somewhere in the results, but not necessarily in the group associated with the group owner. So groups may be non-exhaustive, but the results, in total, are exhaustive.

If producing a single, exhaustive list of fuzzy duplicates for a specific value in the test field is important to your analysis, you can use the ISFUZZYDUP( ) function for this purpose. For more information, see "Fuzzy duplicate helper functions" on page 1223.

Group formation in detail

The fuzzy duplicates feature creates non-exhaustive groups by excluding values from a group if they have appeared with the group owner in a previous group. This approach to group formation reduces the number of redundant fuzzy duplicate pairings, and helps control the overall size of the results.

The rules governing group formation are explained below, with associated examples.

Rule	Explanation
The owner-member relation is non-reciprocal	Because the fuzzy duplicates operation moves sequentially down the test field, group owners are associated with only those fuzzy duplicates that appear below them in the field, not with any that appear above them. In many cases, a group owner is a member of one or more groups that appear above it. However, the reverse is not true. The owners of the groups above are not members of the subsequent group. Once a value becomes a group owner it never appears in a subsequent group. In the example above, the **Group 6** owner, “Jansen”, is a member of two previous groups, but the owners of those groups (“Hansen” and “Janson”), even though they are fuzzy duplicates of “Jansen”, are not members of **Group 6**.
If two values are members of a previous group, they will not be put together in a subsequent group if one of the values is the owner of the subsequent group	In the example above, “Jansen”, “Jansan”, and “Jansn” are all members of **Group 3**. When “Jansen” becomes the **Group 6** owner, “Jansan” and “Jansn” are not placed in the group, even though they are both fuzzy duplicates that appear below “Jansen” in the test field.
If two values are members of a previous group, they can appear together in a subsequent group if neither of the values is the owner of the subsequent group	In the example above, “Hanson” and “Jansen” appear together in both **Group 2** and **Group 3**. In this instance, appearance together in more than one group can occur because the degree of difference is being measured from the respective group owners, not from each other.

Rule	Explanation

> **Note**
> On occasion, there can be exceptions to the second and third rules. During execution, the fuzzy duplicates operation stores temporary values. If the space allotted to these temporary values is filled, the result can be some group owners with one or more group members that are redundant. (The owner and the member have appeared together in a previous group.) The smaller the specified maximum size for fuzzy duplicate groups, the more likely it is that this redundancy can occur.

Fuzzy Duplicates data processing and group formation

The table below shows the record-by-record processing of the example above.

Input data		Data processed in descending sequence	Output results		
Rec #	Last name	Fuzzy duplicates found	Group #	Group owner	Group members (exclude any that have appeared with the group owner in a previous group)
1	Ronson				
2	Hansen	Hanssen, Hanson, Jansen	2	Hansen	Hanssen, Hanson, Jansen
3	Janson	Hanson, Jansen, Jansan, Jansn	3	Janson	Hanson, Jansen, Jansan, Jansn
4	Hanssen				
5	Hanson				
6	Jansen	Jansan, Janszen, Jansn	6	Jansen	Janszen
7	Jansan	Jansn			
8	Janszen				
9	Jansn				
(Difference settings: Difference Threshold = 1, Difference Percentage = 99)					

Including exact duplicates in results

When processing data, the fuzzy duplicates operation always includes exact duplicates but it filters them out of the results unless you select **Include Exact Duplicates** in the **Fuzzy Duplicates** dialog box.

Exact duplicates are subject to the same group formation rules as fuzzy duplicates. They are excluded from a group if they have appeared with the group owner in a previous group. If the group owner and the excluded value are exact duplicates, it may appear that the excluded value should be in the owner's group. However, the exclusion is consistent with the group formation rules because the two values have been in a previous group together.

The table below shows the processing of exact duplicates.

- "Ronson (3)" does not form a group with "Ronson (4)" because the two values are already together in Group 1.
- "Jansen (9)" is excluded from the group formed by "Jansen (8)" because the two values are already together in Group 2 and Group 5.

Input data		Data processed in descending sequence	Output results		
Rec #	Last name	Fuzzy duplicates and exact duplicates found	Group #	Group owner	Group members (exclude any that have appeared with the group owner in a previous group)
1	**Ronson**	Ronson (3), Ronson (4)	1	Ronson	Ronson (3), Ronson (4)
2	Hansen	Hanssen, Hanson, Jansen (8), Jansen (9)	2	Hansen	Hanssen, Hanson, Jansen (8), Jansen (9)
3	**Ronson**	Ronson (4)			
4	**Ronson**				
5	Janson	Hanson, Jansen (8), Jansen (9), Jansan, Jansn	5	Janson	Hanson, Jansen (8), Jansen (9), Jansan, Jansn
6	Hanssen				
7	Hanson				
8	**Jansen**	Jansen (9), Jansan, Janszen, Jansn	8	Jansen	Janszen
9	**Jansen**	Jansan, Janszen,	9	Jansen	Janszen

Input data		Data processed in descending sequence	Output results		
Rec #	Last name	Fuzzy duplicates and exact duplicates found	Group #	Group owner	Group members (exclude any that have appeared with the group owner in a previous group)
		Jansn			
10	Jansan	Jansn			
11	Janszen				
12	Jansn				
(Difference settings: Difference Threshold = 1, Difference Percentage = 99, Include Exact Duplicates = Y)					

Grouping data

Grouping data creates an overview that can help identify patterns, trends, irregularities, or outliers. You group data based on values in a field, or on combinations of values in more than one field.

Grouping allows you to determine how many records, and how much value or quantity, is concentrated by the measures or identifiers that you choose.

Values you can use for grouping

One powerful aspect of grouping is that you can base groups on a wide variety of different types of values:

- Numeric or value range
- Specific number
- Similar numbers
- Time period
- Specific date
- Record identifier, such as:
  - Vendor or customer number
  - Transaction code
  - Product identifier
  - Location code

Grouping operations

Operation	Data type	Functionality	Output
Stratify	Numeric	◦ Groups numeric data. ◦ Groups records into numeric intervals within a range, and subtotals on specified numeric fields. You can specify the size of the range, and the number of intervals in the range.	◦ Screen ◦ Graph ◦ Print ◦ File (text or Analytics table)
Age	Datetime	◦ Groups datetime data. ◦ Groups records into date ranges or aging periods, and subtotals on specified numeric fields.	◦ Screen ◦ Graph ◦ Print ◦ File (text)
Classify	Character Numeric	◦ Groups character or numeric data. ◦ Groups records based on values in a character or numeric field, and subtotals on specified numeric fields.	◦ Screen ◦ Graph ◦ Print ◦ File (Analytics table)
Summarize	Character	◦ Groups character, numeric, or datetime data.	◦ Screen

Operation	Data type	Functionality	Output
	Numeric Datetime	◦ Groups records based on values in one or more character, numeric, or datetime fields, and subtotals on specified numeric fields. ◦ Summarize is similar to Classify, but Summarize allows you to group records by more than one field.	◦ Print ◦ File (Analytics table)
Cross-tabulate	Character Numeric	◦ Groups character or numeric data. ◦ Groups records based on values in two or more character or numeric fields, and displays the resulting groups in a grid of rows and columns. Also subtotals on specified numeric fields. ◦ Cross-tabulate is similar to Summarize using two character or numeric fields.	◦ Screen ◦ Graph ◦ Print ◦ File (text or Analytics table)
Histogram	Character Numeric	◦ Groups character or numeric data. ◦ Groups records based on values in a character or a numeric field, and displays the resulting groups in a bar chart. Does not support subtotaling numeric fields. ◦ Histogram using a character field is similar to Classify. Histogram using a numeric field is similar to Stratify.	◦ Screen ◦ Graph ◦ Print ◦ File (text)
Cluster	Numeric	◦ Groups numeric data. ◦ Groups records based on similar, or nearby, values in one or more numeric fields.	◦ File (Analytics table)

Stratifying data

Stratifying groups the records in a table into numeric intervals (value ranges) based on values in a numeric field, and counts the number of records in each interval.

For example, you could stratify an accounts receivable table on the invoice amount field to group records into $5000 intervals – invoices from $0 to $4,999.99, from $5,000 to $9,999.99, and so on – and to find the total number of transactions, and the total transaction amount, for each interval.

Subtotaling associated numeric fields

When stratifying, you can optionally subtotal one or more associated numeric fields. In the example above, you could subtotal the discount amount field to find the total discount amount for each interval.

> **Note**
> If you do not specify a subtotal field, the field you are stratifying on is automatically subtotaled.

How numeric intervals work

Numeric intervals are value ranges. You have two options when creating numeric intervals:

- equal-sized intervals
- custom-sized intervals

Equal-sized intervals

Analytics calculates equal-sized intervals by grouping the values in the key field into a specified number of intervals.

To create equal-sized intervals, you specify the minimum value of the first interval and the maximum value of the last interval, and the number of intervals you want.

> **Tip**
> If you use the actual minimum and maximum values in the field, the interval size is typically not a round amount. If you want the interval size to be a round amount, you can specify minimum and maximum values in round amounts – for example, 0 and 5000.

Custom-sized intervals

Analytics calculates custom-sized intervals by grouping the values in the key field into intervals with starting values that you specify.

To create custom-sized intervals, you specify the start value of each interval and the end value of the last interval. You can create equal-sized intervals, or intervals that vary in size.

Examples of equal-sized and custom-sized intervals

The table below shows examples of the types of intervals that you could create for a set of values that ranges from $48.19 to $4,792.83.

Equal-sized intervals (using actual min and max values)	Equal-sized intervals (min and max specified as round numbers)	Custom-sized intervals
48.19 - 997.11	0.00 - 999.99	0.00 - 99.99
997.12 - 1,946.04	1,000.00 - 1,999.99	100.00 - 999.99
1,946.05 - 2,894.97	2,000.00 - 2,999.99	1,000.00 - 5,000.00
2,894.98 - 3,843.90	3,000.00 - 3,999.99	
3,843.91 - 4,792.83	4,000.00 - 5,000.00	

Stratifying and sorting

You can stratify sorted or unsorted tables. When you stratify an unsorted table Analytics automatically sorts the output results as part of the stratify operation.

The Statistics option

The **Include Statistics for Subtotal Fields** option allows you to calculate average, minimum, and maximum values for each subtotaled numeric field. In the example above, using the statistics option would calculate the average, minimum, and maximum invoice amounts in each interval, and the average, minimum, and maximum discount amounts in each interval if you also subtotaled the discount amount field.

Stratifying in detail

Stratifying performs the following operations:

Operation	Location in "Stratify results" below
Groups the records into intervals based on a numeric field	**Trans Amount** field, first
Counts (subtotals) the number of records falling into each interval, and calculates the percentage of the total count represented by each subtotal	**Count** field **Percent of Count** field
Provides the minimum and maximum values in the numeric field being stratified	not shown
Optionally subtotals the values of one or more numeric fields for each interval, and for the first selected field calculates the percentage of the field total represented by each subtotal	**Trans Amount** field, second **Percent of Field** field
Optionally calculates average, minimum, and maximum values for each subtotaled numeric field	not shown
Provides totals for all numeric fields included in the output results	**Totals** row
Optionally breaks down the output results based on the values in a character field such as customer ID or transaction type (requires that the character field is sorted prior to stratifying)	not shown

Stratify results

Output results produced by:

- stratifying on transaction amount in an accounts receivable table

  (the **Ar** table in `ACL DATA\Sample Data Files\Sample Project.ACL`)
- using $1000 intervals
- outputting the results to screen

Trans Amount	Count	Percent of Count	Percent of Field	Trans Amount
-4,000.00 - -3,000.01	1	0.13%	-0.76%	-3,582.98
-3,000.00 - -2,000.01	3	0.39%	-1.36%	-6,371.13
-2,000.00 - -1,000.01	16	2.07%	-4.69%	-21,974.43
-1,000.00 - -0.01	141	18.26%	-5.65%	-26,468.32
0.00 - 999.99	421	54.53%	44.55%	208,898.46
1,000.00 - 1,999.99	151	19.56%	43.96%	206,110.67
2,000.00 - 2,999.99	24	3.11%	12.15%	56,980.77
3,000.00 - 3,999.99	12	1.55%	8.61%	40,357.68
4,000.00 - 4,999.99	2	0.26%	2%	9,380.78
5,000.00 - 6,000.00	1	0.13%	1.18%	5,549.19
Totals	772	100%	100%	468,880.69

Steps

You can stratify data by grouping the records in a table into equal-sized, or custom-sized, numeric intervals.

For each interval, you can optionally include the following calculations for associated numeric fields: subtotal, average value, minimum value, maximum value.

Show me how

1. Select **Analyze > Stratify**.
2. On the **Main** tab, do one of the following:
   - Select a field to stratify on from the **Stratify On** drop-down list.
   - Click **Stratify On** to select the field, or to create an expression.
3. Optional. Select one or more **Subtotal Fields**, or click **Subtotal Fields**, to select the subtotal field(s), or to create an expression.

   If you do not select a subtotal field, the field you are stratifying on is automatically subtotaled. You must explicitly select the stratify field if you want to subtotal it along with one or more other fields, or if you want to include statistics for the subtotaled stratify field.

   The order in which you select the subtotal fields is the order in which the columns appear in the results. If you are appending results to an existing Analytics table, the column selection and order must be identical to the column selection and order in the existing table.
4. In **Minimum**, enter the minimum value of the first interval.

   If you previously performed a profile or statistics operation on the stratify field, the lowest value in the stratify field is automatically entered by default. You can change the default, if required.
5. In **Maximum**, enter the maximum value of the last interval.

   If you previously performed a profile or statistics operation on the stratify field, the highest value in the stratify field is automatically entered by default. You can change the default, if required.

6. Do one of the following:
   - Select **Intervals**, and enter the number of equal-sized intervals that you want in the range specified by the **Minimum** and **Maximum** values. The default number of intervals is 10.

     > **Tip**
     > You can change the default number of intervals by selecting **Tools > Options** and updating the **Intervals** number on the **Command** tab.

   - Select **Free** to create custom-sized intervals, and enter the start value of each interval and the end value of the last interval. You must enter each value on a separate line.

     Specifying **Minimum** and **Maximum** values is optional when you use **Free**. If you do specify **Minimum** and **Maximum** values, those values are the start point of the first interval and the end point of the last interval, and the values you enter create additional intervals within the range. The values you enter must be greater than the value specified in **Minimum**, and equal to or less than the value specified in **Maximum**.
7. If there are records in the current view that you want to exclude from processing, enter a condition in the **If** text box, or click **If** to create an IF statement using the **Expression Builder**.

   > **Note**
   > The **If** condition is evaluated against only the records remaining in a table after any scope options have been applied (**First**, **Next**, **While**).

   The IF statement considers all records in the view and filters out those that do not meet the specified condition.
8. Optional. Select **Include Statistics for Subtotal Fields** if you want to calculate average, minimum, and maximum values for each subtotaled numeric field.

   You must select at least one subtotal field in order to include statistics.
9. Click the **Output** tab.
10. Select the appropriate output option in the **To** panel:
    - **Screen** – Select this option to display the results in the Analytics display area.

      > **Tip**
      > You can click any linked result value in the display area to drill down to the associated record or records in the source table.

      If the output table contains a large number of records, it is faster and more useful to save the results to a file than to display the results on the screen.
    - **Print** – Select this option to send the results to the default printer.
    - **Graph** – Select this option to create a graph of the results and display it in the Analytics display area.
    - **File** – Select this option to save or append the results to an Analytics table or a text file. If you save or append to an Analytics table, the table is added to the open project if it is not already in the project. If you save or append to a text file, the file is saved outside Analytics.

> **Note**
> Output options that do not apply to a particular analytical operation are disabled.

11. If you selected **File** as the output type, specify the following information in the **As** panel:
    - **File Type** – Select **Analytics Table** to save the results to a new Analytics table, or append the results to an existing Analytics table. Select **ASCII Text File** or **Unicode Text file** (depending on which edition of Analytics you are using) to save or append the results to a text file.
    - **Name** – Enter a file name in the **Name** text box. Or click **Name** and enter the file name, or select an existing file in the **Save** or **Save File As** dialog box to overwrite or append to the file. If Analytics prefills a file name, you can accept the prefilled name, or change it.

      You can also specify an absolute or relative file path, or navigate to a different folder, to save or append the file in a location other than the project location. For example: `C:\Results\Output.fil` or `Results\Output.fil`.
    - **Local** – Only enabled when connected to a server table and saving or appending the results to an Analytics table. Select **Local** to save the file to the same location as the project, or to specify a path or navigate to a different local folder. Leave **Local** deselected to save the file to the Prefix folder on a server.

      > **Note**
      > For output results produced from analysis or processing of AX Server tables, select **Local**. You cannot deselect the **Local** setting to import results tables to AX Server.

12. Depending on the output type, you can optionally specify a **Header** and/or a **Footer** in the text box(es).

    Headers and footers are centered by default. Type a left angle bracket (<) before the header or footer text to left align the text. Click **Header** or **Footer** to enter a header or footer of more than one line. Alternatively, you can enter a semi-colon (;) as a line-break character in the header or footer text box. Left aligning multiple lines requires a left angle bracket at the beginning of each line.
13. Click the **More** tab.
14. Select the appropriate option in the **Scope** panel:
    - **All**
    - **First**
    - **Next**
    - **While**

    Show me more

All	This option is selected by default. Leave it selected to specify that all records in the view are processed.
First	Select this option and enter a number in the text box to start processing at the first record in the view and include only the specified number of records.

Next	Select this option and enter a number in the text box to start processing at the currently selected record in the view and include only the specified number of records. The actual record number in the leftmost column must be selected, not data in the row.
While	Select this option to use a WHILE statement to limit the processing of records in the view based on a particular criterion or set of criteria. You can enter a condition in the **While** text box, or click **While** to create a WHILE statement using the **Expression Builder**. A WHILE statement allows records in the view to be processed only while the specified condition evaluates to true. As soon as the condition evaluates to false, the processing terminates, and no further records are considered. You can use the **While** option in conjunction with the **All**, **First**, or **Next** options. Record processing stops as soon as one limit is reached.

> **Note**
>
> The number of records specified in the **First** or **Next** options references either the physical or the indexed order of records in a table, and disregards any filtering or quick sorting applied to the view. However, results of analytical operations respect any filtering.
>
> If a view is quick sorted, **Next** behaves like **First**.

15. If you do not want to include values that exceed the specified **Minimum** and **Maximum** values, select **Suppress Others**.
16. If you want to break down the output results based on the values in a character field, enter the field name in the **Break** text box, or click **Break** to select the field, or to create an expression.

    For example, the results of stratifying an accounts receivable table by transaction amount could be further broken down by customer. **Break** can only be used with a single character field, so nested breakdowns are not supported.

    > **Note**
    >
    > For the **Break** option to yield meaningful results, the character field used for the breakdown must be sorted prior to stratifying.

17. If you selected **File** as the output type, and want to append the output results to the end of an existing file, do one of the following:
    - Select **Append To Existing File** if you are appending to a text file, or to an Analytics table that you are certain is identical in structure to the output results.
    - Leave **Append To Existing File** deselected if you are appending to an Analytics table and you want Analytics to compare the record lengths of the output results and the existing table. If the record lengths are not identical, the data structure is not identical, and the append will not work correctly.

> **Note**
>
> Leaving **Append To Existing File** deselected is recommended if you are uncertain whether the output results and the existing table have an identical data structure. For more information about appending and data structure, see "Appending output results to an existing table" on page 200.

18. If you selected **File (Analytics Table)** as the output type, select **Use Output Table** if you want the output table to open automatically upon completion of the operation.
19. Click **OK**.
20. If the overwrite prompt appears, select the appropriate option.

    If you are expecting the **Append** option to appear and it does not, click **No** to cancel the operation and see "Appending output results to an existing table" on page 200.

Aging data

Aging groups the records in a table into aging periods based on values in a date or datetime field, and counts the number of records in each period.

Common uses of aging include evaluating sales trends, looking at transaction volumes, and grouping invoices by the number of days outstanding.

For example, you could age an accounts receivable table on the invoice date field to group records into 30-day periods – invoices from the cutoff date to 29 days previous, from 30 days previous to 59 days previous, and so on – and to find the total number of outstanding invoices for each period.

> **Note**
> You can age on datetime values, however only the date portion of the values is considered. The time portion is ignored. You cannot age on time data alone.

Subtotaling numeric fields

When aging, you can optionally subtotal one or more numeric fields. In the example above, you could subtotal the invoice amount field to find the total outstanding invoice amount for each aging period.

How aging periods work

Aging periods are based on date intervals (that is, number of days) measured backward in time from either:

- The current system date
- A cutoff date you specify such as a fiscal period end date

Specifying a single date interval of 30 creates an aging period that includes any dates 30 days prior to the cutoff date, or earlier.

Specifying multiple date intervals creates multiple aging periods. You can specify date intervals such as 0, 90, and 120 days as starting points for aging periods, or you can accept the default settings of 0, 30, 60, 90, 120, and 10,000 days.

An interval of 10,000 days, or an appropriate final interval you specify, is used to isolate records with dates that are probably invalid.

The table below shows how the cutoff date and the date intervals combine to create five aging periods, and the dates that are included in each period.

Cutoff date	Date intervals				
31 Dec 2016	0	30	60	90	120
	includes: 31 Dec 2016 to 02 Dec 2016	includes: 01 Dec 2016 to 02 Nov 2016	includes: 01 Nov 2016 to 03 Oct 2016	includes: 02 Oct 2016 to 02 Sep 2016	includes: 01 Sep 2016 to earliest date

The Statistics option

The **Include Statistics for Subtotal Fields** option allows you to calculate average, minimum, and maximum values for each subtotaled numeric field. In the example above, using the statistics option would calculate the average, minimum, and maximum invoice amounts for each aging period.

Aging in detail

Aging performs the following operations:

Operation	Location in "Aging results" on the facing page
Groups the records into aging periods based on cutoff date and date intervals	**Days** field
Counts (subtotals) the number of records in each aging period, and calculates the percentage of the total count represented by each subtotal	**Count** field **Percent of Count** field
Provides the minimum and maximum ages of records (that is, the most recent and the oldest)	not shown
Optionally subtotals the values of one or more numeric fields for each aging period, and for the first selected field calculates the percentage of the total value represented by each subtotal	**Trans Amount** field **Percent of Field** field
Optionally calculates average, minimum, and maximum values for each subtotaled numeric field	not shown
Provides totals for all numeric fields included in the output results	**Totals** row
Optionally breaks down the output results based on the values in a character field such as customer ID or transaction type (requires that the character field is sorted prior to aging)	not shown

Aging results

Output results produced by:

- aging on invoice date in an accounts receivable table

  (the **Ar** table in `ACL DATA\Sample Data Files\Sample Project.ACL`)

- subtotaling transaction amount
- using 30-day aging periods
- outputting the results to screen

Days	Count	Percent of Count	Percent of Field	Trans Amount
0 - 29	212	27.46%	6.06%	28,422.47
30 - 59	240	31.09%	36.16%	169,527.02
60 - 89	179	23.19%	27.54%	129,133.34
90 - 119	107	13.86%	25.63%	120,153.91
120 - 10,000	34	4.4%	4.62%	21,643.95
Totals	772	100%	100%	468,880.69

> **Note**
>
> If you output the results to screen or graph, the graph displays the count subtotals for each aging period, or the numeric subtotals if you include one or more numeric subtotal fields in the aging operation.

Steps

You can age data by grouping the records in a table into aging periods.

For each period, you can optionally include the following calculations for associated numeric fields: subtotal, average value, minimum value, maximum value.

Show me how

1. Select **Analyze > Age**.
2. On the **Main** tab, do one of the following:
   - Select the field on which to base the aging from the **Age On** drop-down list.
   - Click **Age On** to select the field, or to create an expression.
3. In the **Cutoff Date** field, leave the default current date, or do one of the following to specify a different cutoff date:
   - Edit the date directly in the **Cutoff Date** field.
   - Click the down arrow to select a date from the calendar. You can use the left or right arrows to move backward or forward one month at a time, or click the month and year, year, or decade at the top center of the calendar to move backward or forward in larger intervals of time.

Specifying a different cutoff date allows you to align the beginning of the first aging period with a date such as a fiscal period end date. If you leave the default date, the first aging period begins on the current date, which may or may not be appropriate for your analysis.

4. Enter the aging periods to use in the **Aging Periods** text box, or keep the default values.

   The aging period values must be entered in days. Each value must be listed on a separate line from lowest to highest (most recent to oldest). A value of '0' specifies that the first aging period begins on the specified cutoff date. The final value specifies the end of the oldest aging period.

   > **Note**
   > You can change the values used for the default aging periods by selecting **Tools > Options** and updating the **Aging Periods** on the **Date and Time** tab.

5. Optional. Select one or more **Subtotal Fields**, or click **Subtotal Fields**, to select the subtotal field(s), or to create an expression.

   The order in which you select the subtotal fields is the order in which the columns appear in the results. If you are appending results to an existing Analytics table, the column selection and order must be identical to the column selection and order in the existing table.

6. If there are records in the current view that you want to exclude from processing, enter a condition in the **If** text box, or click **If** to create an IF statement using the **Expression Builder**.

   > **Note**
   > The **If** condition is evaluated against only the records remaining in a table after any scope options have been applied (**First**, **Next**, **While**).

   The IF statement considers all records in the view and filters out those that do not meet the specified condition.

7. Optional. Select **Include Statistics for Subtotal Fields** if you want to calculate average, minimum, and maximum values for each subtotaled numeric field.

   You must select at least one subtotal field in order to include statistics.

8. Click the **Output** tab.
9. Select the appropriate output option in the **To** panel:
   - **Screen** – Select this option to display the results in the Analytics display area.

     > **Tip**
     > You can click any linked result value in the display area to drill down to the associated record or records in the source table.

     If the output table contains a large number of records, it is faster and more useful to save the results to a file than to display the results on the screen.

   - **Print** – Select this option to send the results to the default printer.
   - **Graph** – Select this option to create a graph of the results and display it in the Analytics display area.

- **File** – Select this option to save or append the results to a text file. The file is saved outside Analytics.

> **Note**
> Output options that do not apply to a particular analytical operation are disabled.

10. If you selected **File** as the output type, specify the following information in the **As** panel:
    - **File Type** – **ASCII Text File** or **Unicode Text file** (depending on which edition of Analytics you are using) is the only option. Saves the results to a new text file, or appends the results to an existing text file.
    - **Name** – Enter a file name in the **Name** text box. Or click **Name** and enter the file name, or select an existing file in the **Save** or **Save File As** dialog box to overwrite or append to the file. If Analytics prefills a file name, you can accept the prefilled name, or change it.

      You can also specify an absolute or relative file path, or navigate to a different folder, to save or append the file in a location other than the project location. For example: `C:\Results\Output.txt` or `Results\Output.txt`.
    - **Local** – Disabled and selected. Saving the file locally is the only option.
11. Depending on the output type, you can optionally specify a **Header** and/or a **Footer** in the text box(es).

    Headers and footers are centered by default. Type a left angle bracket (<) before the header or footer text to left align the text. Click **Header** or **Footer** to enter a header or footer of more than one line. Alternatively, you can enter a semi-colon (;) as a line-break character in the header or footer text box. Left aligning multiple lines requires a left angle bracket at the beginning of each line.
12. Click the **More** tab.
13. Select the appropriate option in the **Scope** panel:
    - **All**
    - **First**
    - **Next**
    - **While**

Show me more

All	This option is selected by default. Leave it selected to specify that all records in the view are processed.
First	Select this option and enter a number in the text box to start processing at the first record in the view and include only the specified number of records.
Next	Select this option and enter a number in the text box to start processing at the currently selected record in the view and include only the specified number of records. The actual record number in the leftmost column must be selected, not data in the row.
While	Select this option to use a WHILE statement to limit the processing of records in the view based on a particular criterion or set of criteria. You can enter a condition in the **While** text box, or click **While** to create a WHILE statement using the **Expression Builder**. A WHILE statement allows records in the view to be processed only while the specified

	condition evaluates to true. As soon as the condition evaluates to false, the processing terminates, and no further records are considered. You can use the **While** option in conjunction with the **All**, **First**, or **Next** options. Record processing stops as soon as one limit is reached.

> **Note**
> The number of records specified in the **First** or **Next** options references either the physical or the indexed order of records in a table, and disregards any filtering or quick sorting applied to the view. However, results of analytical operations respect any filtering.
> If a view is quick sorted, **Next** behaves like **First**.

14. If you want to exclude from the output results any values that fall outside the specified aging periods, select **Suppress Others**.
15. If you want to break down the output results based on the values in a character field, enter the field name in the **Break** text box, or click **Break** to select the field, or to create an expression.

    For example, an aged summary of an accounts receivable table could be broken down by customer. **Break** can only be used with a single character field, so nested breakdowns are not supported.

    > **Note**
    > For the **Break** option to yield meaningful results, the character field must be sorted prior to aging.

16. If you selected **File** as the output type, and want to append the output results to the end of an existing text file, select **Append To Existing File**.
17. Click **OK**.

    If you output the results to screen or graph, you can switch between the two output types using the **Text** and **Graph** buttons at the bottom of the display area.
18. If the overwrite prompt appears, select the appropriate option.

Classifying versus summarizing

Classifying and summarizing are similar methods of grouping data, but they have different options and process data in different ways. You can classify or summarize sorted or unsorted tables.

You must summarize instead of classify if you want to do any of the following:

- work with a datetime key field
- use multiple key fields
- include non-key fields in the output table

Requirement	Classifying	Summarizing
Calculates and reports the number of times a key field value appears in a table	Yes	Yes
Computes and displays subtotals for selected numeric fields	Yes	Yes
Computes and displays average, minimum, and maximum values for subtotaled numeric fields	Yes	Yes
Computes and displays percentages	Yes	Yes
Computes and displays additional statistical values for subtotaled numeric fields (standard deviation, median, mode, first and third quartile)	No	Yes
Key field can be character	Yes	Yes
Key field can be numeric	Yes	Yes
Key field can be datetime	No	Yes
Multiple key fields allowed	No	Yes
Include non-key fields in output	No	Yes
Primary processing location	RAM	Hard disk
Key field length restriction	Yes (maximum 2048 characters)	No
Sorts output results	Yes	Yes (**Presort** selected) No (**Presort** not selected)

Requirement	Classifying	Summarizing
Output to table, screen, or print	Yes	Yes
Output to graph	Yes	No

Classifying data

Classifying groups the records in a table based on identical key field values, and counts the number of records in each group. Key fields can be character or numeric.

For example, you could classify a transactions table on the customer number field to find the total number of transactions for each customer.

In the example below, there are ten values in the Customer Number field in the input table. Some values are unique, and some values are identical. After summarizing, the values are grouped into four unique groups. The Count tells you how many records, or transactions, are in each customer number group.

Input table	Output results	
Key field: Customer Number	Classified group	Count
795401	230575	1
518008	518008	5
518008	795401	3
925007	925007	1
518008		
795401		
518008		
230575		
795401		
518008		

Subtotaling associated numeric fields

When classifying, you can optionally subtotal one or more associated numeric fields. In the example above, you could subtotal the transaction amount field to find the total transaction amount for each customer.

Classifying and sorting

You can classify sorted or unsorted tables. When you classify an unsorted table Analytics automatically sorts the output results as part of the classify operation.

Classifying unsorted tables requires Analytics to create a variable for each set of identical values in the key field and store these variables in memory until the entire table is read. If you are working with a large table, storing the required variables requires a lot of RAM and can be slow.

The Statistics option

The **Include Statistics for Subtotal Fields** option allows you to calculate average, minimum, and maximum values for any subtotal field you specify. The results of the calculations are broken down by group in the classified output table.

In the example above, the statistics option would calculate the average, minimum, and maximum transaction amounts for each customer.

Classifying in detail

Classifying performs the following operations:

Operation	Location in "Classify results" on the facing page
Groups the records based on identical values in a character or numeric field	**Product Class** field
Counts (subtotals) the number of records for each group, and calculates the percentage of the total count represented by each subtotal	**Count** field **Percent of Count** field
Optionally subtotals the values of one or more numeric fields for each group, and for the first selected numeric field calculates the percentage of the total value represented by each subtotal	**Inventory Value at Cost** field **Percent of Field** field
Optionally calculates average, minimum, and maximum values for each subtotaled numeric field	not shown
Provides totals for all numeric fields included in the output results	**Totals** row
Optionally breaks down the output results based on the values in a character field such as customer ID or transaction type (requires that the character field is sorted prior to classifying)	not shown

Classify results

Output results produced by:

- classifying on product class in an inventory table

  (the **Inventory** table in `ACL DATA\Sample Data Files\Sample Project.ACL`)

- subtotaling inventory value
- outputting the results to screen

The results show that the inventory value is concentrated in four product classes: 03, 04, 08, 09.

Product Class	Count	Percent of Count	Percent of Field	Inventory Value at Cost
01	17	11.18%	5.14%	34,954.68
02	19	12.5%	3.02%	20,544.20
03	20	13.16%	15.09%	102,702.76
04	17	11.18%	13.08%	89,018.95
05	13	8.55%	6.24%	42,479.36
06	17	11.18%	8.59%	58,479.60
07	7	4.61%	7%	47,609.10
08	19	12.5%	27.66%	188,230.86
09	21	13.82%	11.85%	80,646.05
13	1	0.66%	1.67%	11,352.48
18	1	0.66%	0.66%	4,461.90
Totals	152	100%	100%	680,479.94

Steps

You can classify data by grouping the records in a table based on identical values in a character or numeric field.

For each group, you can optionally include the following calculations for associated numeric fields: subtotal, average value, minimum value, maximum value.

Show me how

> **Note**
>
> Classifying supports a maximum key field length of 2048 characters.
>
> If you want to classify a table using a key field longer than 2048 characters, you can summarize, which does not have a length restriction. For more information, see "Summarizing data" on page 1267.
>
> If you classify a larger data set and output the results to screen or graph, you may exceed available memory. You can reduce memory usage when outputting to screen by selecting **Suppress XML Output for Command Results** (**Tools > Options > Command**).

1. Select **Analyze > Classify**.
2. On the **Main** tab, do one of the following:
   - Select the field to classify from the **Classify On** drop-down list.
   - Click **Classify On** to select the field, or to create an expression.
3. Optional. Select one or more **Subtotal Fields**, or click **Subtotal Fields**, to select the subtotal field(s), or to create an expression.

   The order in which you select the subtotal fields is the order in which the columns appear in the results. If you are appending results to an existing Analytics table, the column selection and order must be identical to the column selection and order in the existing table.
4. If there are records in the current view that you want to exclude from processing, enter a condition in the **If** text box, or click **If** to create an IF statement using the **Expression Builder**.

   > **Note**
   > The **If** condition is evaluated against only the records remaining in a table after any scope options have been applied (**First**, **Next**, **While**).

   The IF statement considers all records in the view and filters out those that do not meet the specified condition.
5. Optional. Select **Include Statistics for Subtotal Fields** if you want to calculate average, minimum, and maximum values for each subtotaled numeric field.

   You must select at least one subtotal field in order to include statistics.
6. Click the **Output** tab.
7. Select the appropriate output option in the **To** panel:
   - **Screen** – Select this option to display the results in the Analytics display area.

     > **Tip**
     > You can click any linked result value in the display area to drill down to the associated record or records in the source table.

     If the output table contains a large number of records, it is faster and more useful to save the results to a file than to display the results on the screen.
   - **Print** – Select this option to send the results to the default printer.
   - **Graph** – Select this option to create a graph of the results and display it in the Analytics display area.
   - **File** – Select this option to save or append the results to an Analytics table. The table is added to the open project if it is not already in the project.

     > **Note**
     > Output options that do not apply to a particular analytical operation are disabled.
8. If you selected **File** as the output type, specify the following information in the **As** panel:
   - **File Type** – **Analytics Table** is the only option. Saves the results to a new Analytics table, or appends the results to an existing Analytics table.

- **Name** - Enter a table name in the **Name** text box. Or click **Name** and enter the table name, or select an existing table in the **Save** or **Save File As** dialog box to overwrite or append to the table. If Analytics prefills a table name, you can accept the prefilled name, or change it.

  You can also specify an absolute or relative file path, or navigate to a different folder, to save or append the table in a location other than the project location. For example: `C:\Results\Output.fil` or `Results\Output.fil`.

- **Local** - Only enabled when connected to a server table. Select **Local** to save the output table to the same location as the project, or to specify a path or navigate to a different local folder. Leave **Local** deselected to save the output table to the Prefix folder on a server.

> **Note**
>
> For output results produced from analysis or processing of AX Server tables, select **Local**. You cannot deselect the **Local** setting to import results tables to AX Server.

9. Depending on the output type, you can optionally specify a **Header** and/or a **Footer** in the text box(es).

   Headers and footers are centered by default. Type a left angle bracket (<) before the header or footer text to left align the text. Click **Header** or **Footer** to enter a header or footer of more than one line. Alternatively, you can enter a semi-colon (;) as a line-break character in the header or footer text box. Left aligning multiple lines requires a left angle bracket at the beginning of each line.

10. Click the **More** tab.

11. Select the appropriate option in the **Scope** panel:

    - **All**
    - **First**
    - **Next**
    - **While**

    Show me more

All	This option is selected by default. Leave it selected to specify that all records in the view are processed.
First	Select this option and enter a number in the text box to start processing at the first record in the view and include only the specified number of records.
Next	Select this option and enter a number in the text box to start processing at the currently selected record in the view and include only the specified number of records. The actual record number in the leftmost column must be selected, not data in the row.
While	Select this option to use a WHILE statement to limit the processing of records in the view based on a particular criterion or set of criteria. You can enter a condition in the **While** text box, or click **While** to create a WHILE statement using the **Expression Builder**. A WHILE statement allows records in the view to be processed only while the specified condition evaluates to true. As soon as the condition evaluates to false, the processing terminates, and no further records are considered. You can use the **While** option in

	conjunction with the **All**, **First**, or **Next** options. Record processing stops as soon as one limit is reached.

> **Note**
> The number of records specified in the **First** or **Next** options references either the physical or the indexed order of records in a table, and disregards any filtering or quick sorting applied to the view. However, results of analytical operations respect any filtering.
> If a view is quick sorted, **Next** behaves like **First**.

12. If you want to break down the output results based on the values in a character field, enter the field name in the **Break** text box, or click **Break** to select the field, or to create an expression.

    For example, the results of classifying an accounts receivable table by transaction type could be further broken down by customer. **Break** can only be used with a single character field, so nested breakdowns are not supported.

    > **Note**
    > For the **Break** option to yield meaningful results, the character field used for the breakdown must be sorted prior to classifying.

13. If you selected **File (Analytics Table)** as the output type, select **Use Output Table** if you want the output table to open automatically upon completion of the operation.
14. If you selected **File** as the output type, and want to append the output results to the end of an existing Analytics table, do one of the following:
    - Select **Append To Existing File** if you are certain the output results and the existing table are identical in structure.
    - Leave **Append To Existing File** deselected if you want Analytics to compare the record lengths of the output results and the existing table. If the record lengths are not identical, the data structure is not identical, and the append will not work correctly.

    > **Note**
    > Leaving **Append To Existing File** deselected is recommended if you are uncertain whether the output results and the existing table have an identical data structure. For more information about appending and data structure, see "Appending output results to an existing table" on page 200.

15. Click **OK**.
16. If the overwrite prompt appears, select the appropriate option.

    If you are expecting the **Append** option to appear and it does not, click **No** to cancel the operation and see "Appending output results to an existing table" on page 200.

Summarizing data

Summarizing groups the records in a table based on identical values in one or more key fields, and counts the number of records in each group. You also have the option of performing various statistical calculations for each group.

If you summarize by more than one key field (nested summarizing), groups are based on identical combinations of values across the key fields.

Key fields can be character, numeric, or datetime.

Summarizing by one key field

Summarizing by one key field is the simplest form of summarizing.

For example, you could summarize a transactions table on the customer number field to find the total number of transactions for each customer.

In the example below, there are ten values in the Customer Number field in the input table. Some values are unique, and some values are identical. After summarizing, the values are grouped into four unique groups. The Count tells you how many records, or transactions, are in each customer number group.

Input table	Output results	
Key field: Customer Number	Summarized group	Count
795401	230575	1
518008	518008	5
518008	795401	3
925007	925007	1
518008		
795401		
518008		
230575		
795401		

Input table	Output results	
Key field: Customer Number	Summarized group	Count
518008		

Summarizing by multiple key fields

Summarizing by multiple key fields, or nested summarizing, lets you perform more detailed analysis of data.

For example, you could summarize a transactions table on the customer number and the transaction date fields to find the total number of transactions for each customer for each date that the customer had transactions.

In the example below, there are ten values in the Customer Number field in the input table, with accompanying dates in the Invoice Date field. Some combinations of customer number and date are unique, and some are identical. After summarizing, the customer number-date combinations are grouped into seven unique groups. The Count tells you how many records, or transactions, are in each group.

Input table		Output results		
Key field 1: Customer Number	Key field 2: Invoice Date	Nested summarized group		Count
795401	08/20/2016	230575	06/13/2016	1
518008	10/15/2016	518008	04/30/2016	1
518008	07/17/2016	518008	07/17/2016	3
925007	05/21/2016	518008	10/15/2016	1
518008	04/30/2016	795401	06/30/2016	1
795401	08/20/2016	795401	08/20/2016	2
518008	07/17/2016	925007	05/21/2016	1
230575	06/13/2016			
795401	06/30/2016			
518008	07/17/2016			

Nested summarizing in detail

If you summarize by more than one key field, you create nested summarized groups in the output results.

Nesting hierarchy

The order in which you select the key fields dictates the nesting hierarchy. The records are summarized by the first field you select, and within each of these primary groupings the records are then summarized by the second field you select, and so on.

> **Note**
> Reversing the order in which you select two summarize key fields may give quite different results.

Field order in the output results

The order in which you select the key fields is also the order in which the columns appear in the output results. If you are appending results to an existing Analytics table, the column selection and order must be identical to the column selection and order in the existing table.

Summarizing and sorting

Summarizing can process either sorted or unsorted data. The **Presort** option allows you to include initial sorting of the data with summarizing.

If you use Presort

If you use **Presort**, the output results are sorted and contain a single, unique group for each set of identical values, or identical combination of values, in the key field or fields.

> **Tip**
> If the input table is already sorted, you can save processing time by deselecting the **Presort** option.

If you do not use Presort

If you do not use **Presort**, the output results use the sort order of the input table.

If the key field or fields contain non-sequential identical values, the output results contain more than one group for each set of identical values, or identical combination of values.

> **Note**
> Depending on the context, more than one group for each set of identical values, or identical combination of values, can defeat the purpose of summarizing.

Subtotaling numeric fields

When summarizing, you can optionally subtotal one or more numeric fields. In the examples above, you could subtotal the transaction amount field to calculate:

- The total transaction amount for each customer
- The total transaction amount for each customer for each date that the customer had transactions

The statistical options

You have the option of performing statistical calculations on any subtotal field you specify. The statistical calculations are broken down by group in the output results.

In the examples above, if you subtotal the transaction amount field, you could also use one of the statistical options to calculate:

- the average, minimum, and maximum transaction amounts for each customer
- the average, minimum, and maximum transaction amounts for each customer for each date that the customer had transactions

Subtotal and statistical options in detail

The table below provides details about the subtotal and statistical options and calculations.

Show me more

Option	Alternate column title (display name) in output table	Field name in output table	Calculation performed on subtotaled field
Subtotal Fields	**Total** + *subtotaled alternate column title*	*subtotaled field name*	Subtotaled values for each group
Avg, min, max	**Average** + *subtotaled alternate column title*	**a**_*subtotaled field name*	The average value for each group
	Minimum + *subtotaled alternate column title*	**m**_*subtotaled field name*	The minimum value for each group

Option	Alternate column title (display name) in output table	Field name in output table	Calculation performed on subtotaled field
	Maximum + *subtotaled alternate column title*	**x**_*subtotaled field name*	The maximum value for each group
Std deviation, % of field	**STDDEV** + *subtotaled alternate column title*	**d**_*subtotaled field name*	The standard deviation for each group
	% Field + *subtotaled alternate column title*	**f**_*subtotaled field name*	Each group's subtotal expressed as a percentage of the field total
Median, Mode, Q25, Q75	**Median** + *subtotaled alternate column title*	**c**_*subtotaled field name*	The median value for each group ◦ Odd-numbered sets of values: the middle value ◦ Even-numbered sets of values: the average of the two values at the middle
	Mode + *subtotaled alternate column title*	**o**_*subtotaled field name*	The most frequently occurring value for each group ◦ Displays "N/A" if no value occurs more than once ◦ In the event of a tie, displays the lowest value
	Q25 + *subtotaled alternate column title*	**q**_*subtotaled field name*	The first quartile value for each group (lower quartile value) ◦ The result is an interpolated value based on an Analytics algorithm ◦ Produces the same result as the QUARTILE and QUARTILE.INC functions in Microsoft Excel
	Q75 + *subtotaled alternate column title*	**p**_*subtotaled field name*	The third quartile value for each group (upper quartile value) ◦ The result is an

Option	Alternate column title (display name) in output table	Field name in output table	Calculation performed on subtotaled field
			◦ interpolated value based on an Analytics algorithm ◦ Produces the same result as the QUARTILE and QUARTILE.INC functions in Microsoft Excel
% of Count	**Percent of Count**	**COUNT_PERCENTAGE**	The percentage of source table records belonging to each group **Note** Does not require a subtotal field

The Other Fields option

The **Other Fields** option allows you to select additional character, numeric, or datetime fields to include in the output. This option can provide useful information if the fields you select contain the same value for all records in each summarized group.

For example, if you summarize a table on customer number, an appropriate “other field” could be customer name. The customer name should be identical for all records with the same customer number.

If you specify an “other field” that contains values that are different for a summarized group, only the value for the first record in the group is displayed, which is not meaningful.

For example, if you summarize a vendor table by state, and select city as an “other field”, only the first city listed for each state appears in the output. In this instance, the better approach is to summarize using both state and city as key fields, in that order.

Summarize results

The example below shows the results of summarizing an accounts receivable table on customer number and transaction type. The transaction amount is subtotaled, with some associated statistics. The results are output to screen.

The example uses a subset of customer numbers from the **Ar** table in `ACL DATA\Sample Data Files\Sample Project.ACL`.

Cust Number	Trans Type	Total Trans Amount	Average Trans Amount	Minimum Trans Amount	Maximum Trans Amount	Percent of Count	Count	Name
051593	CN	-73.40	-73.40	-73.40	-73.40	0.80	1	CONNECTICUT CORP.
051593	IN	1,189.11	1,189.11	1,189.11	1,189.11	0.80	1	CONNECTICUT CORP.
056016	IN	1,807.66	903.83	736.74	1,070.92	1.60	2	CITIZENS INTERNATIONAL
056016	PM	-1,807.66	-903.83	-1,070.92	-736.74	1.60	2	CITIZENS INTERNATIONAL
065003	CN	-685.59	-52.74	-146.83	-9.17	10.40	13	UNIVERSITY ELECTRONICS
065003	IN	105,020.57	1,207.13	73.40	4,954.64	69.60	87	UNIVERSITY ELECTRONICS
065003	PM	-8,443.97	-562.93	-1,954.88	116.72	12.00	15	UNIVERSITY ELECTRONICS
081559	IN	1,779.07	1,779.07	1,779.07	1,779.07	0.80	1	KIDDER ENTERPRISES
090398	IN	634.38	317.19	55.02	579.36	1.60	2	AMSER SYSTEMS
097627	IN	1,301.83	1,301.83	1,301.83	1,301.83	0.80	1	STEPPING ELECTRONICS
Totals		100,722.00	5,105.26	1,889.14	10,172.34	100.00	125	

Summarizing in detail

Summarizing performs the following operations:

Operation	Location in "Summarize results" on the previous page above
Groups the records based on identical values, or identical combinations of values, in one or more character, numeric, or datetime key fields	**Cust Number** field **Trans Type** field
Optionally subtotals the values of one or more numeric fields for each group	**Total Trans Amount** field
Optionally performs statistical calculations on each subtotaled numeric field	**Average, Minimum,** and **Maximum Trans Amount** fields **Note** Additional statistical calculations not shown
Optionally calculates the percentage of source table records belonging to each group	**Percent of Count** field
Counts (subtotals) the number of records for each group	**Count** field

Operation	Location in "Summarize results" on the previous page above
Optionally displays additional character, numeric, or datetime fields with complementary information	**Name** field
Provides totals for all numeric fields included in the output results **Note** The **Totals** row is only provided when you output the results to screen.	**Totals** row

Steps

You can summarize data by grouping the records in a table based on identical values, or identical combinations of values, in one or more character, numeric, or datetime fields.

You can optionally subtotal associated numeric fields. You can also perform statistical calculations on any subtotal field you specify. The results of the statistical calculations are broken down by group in the summarized output table.

Show me how

1. Select **Analyze > Summarize**.
2. On the **Main** tab, do one of the following:
   - Select the field(s) to summarize from the **Summarize On** list.
   - Click **Summarize On** to select the field(s), or to create an expression.

   > **Note**
   > If you select more than one field you create nested summarized groups in the output results. For more information, see "Nested summarizing in detail" on page 1269.

3. Optional. Select one or more **Subtotal Fields**, or click **Subtotal Fields**, to select the subtotal field(s), or to create an expression.

   The order in which you select the subtotal fields is the order in which the columns appear in the results. If you are appending results to an existing Analytics table, the column selection and order must be identical to the column selection and order in the existing table.
4. Optional. Do one of the following:
   - From the **Other Fields** list, select the other field(s) to include in the output results.
   - Click **Other Fields** to select the field(s), or to create an expression.

> **Note**
> Select only fields that contain the same value for all records in each summarized group. For more information, see "The Other Fields option" on page 1272.

5. If the field(s) you are summarizing are already sorted, you can optionally deselect **Presort**.

> **Note**
> You can summarize unsorted fields, but the results may contain more than one summarized group for the same value, which may defeat the purpose of summarizing.
>
> Depending on the nature of your analysis, summarizing unsorted fields may be appropriate.

6. If there are records in the current view that you want to exclude from processing, enter a condition in the **If** text box, or click **If** to create an IF statement using the **Expression Builder**.

> **Note**
> The **If** condition is evaluated against only the records remaining in a table after any scope options have been applied (**First**, **Next**, **While**).

   The IF statement considers all records in the view and filters out those that do not meet the specified condition.

7. Optional. Select one or more of the statistical options to perform statistical calculations on subtotal fields:
   - **Avg, min, max**
   - **Std deviation, % of field**
   - **Median, Mode, Q25, Q75**
   - **% of Count**

   For more information, see "The statistical options" on page 1270.

> **Note**
> You must select at least one subtotal field in order to include statistics.
>
> **% of Count** does not require a subtotal field.
>
> Calculating these statistics requires additional computer memory. You may exceed your computer's memory and get an error message if you calculate the statistics for very large data sets.

8. Click the **Output** tab.

9. Select the appropriate output option in the **To** panel:
   - **Screen** - Select this option to display the results in the Analytics display area.

     > **Tip**
     > You can click any linked result value in the display area to drill down to the associated record or records in the source table.

     If the output table contains a large number of records, it is faster and more useful to save the results to a file than to display the results on the screen.
   - **Print** - Select this option to send the results to the default printer.
   - **Graph** - Select this option to create a graph of the results and display it in the Analytics display area.
   - **File** - Select this option to save or append the results to an Analytics table. The table is added to the open project if it is not already in the project.

     > **Note**
     > Output options that do not apply to a particular analytical operation are disabled.

10. If you selected **File** as the output type, specify the following information in the **As** panel:
    - **File Type** - **Analytics Table** is the only option. Saves the results to a new Analytics table, or appends the results to an existing Analytics table.
    - **Name** - Enter a table name in the **Name** text box. Or click **Name** and enter the table name, or select an existing table in the **Save** or **Save File As** dialog box to overwrite or append to the table. If Analytics prefills a table name, you can accept the prefilled name, or change it.

      You can also specify an absolute or relative file path, or navigate to a different folder, to save or append the table in a location other than the project location. For example: `C:\Results\Output.fil` or `Results\Output.fil`.
    - **Local** - Only enabled when connected to a server table. Select **Local** to save the output table to the same location as the project, or to specify a path or navigate to a different local folder. Leave **Local** deselected to save the output table to the Prefix folder on a server.

      > **Note**
      > For output results produced from analysis or processing of AX Server tables, select **Local**. You cannot deselect the **Local** setting to import results tables to AX Server.

11. Depending on the output type, you can optionally specify a **Header** and/or a **Footer** in the text box(es).

    Headers and footers are centered by default. Type a left angle bracket (<) before the header or footer text to left align the text. Click **Header** or **Footer** to enter a header or footer of more than one line. Alternatively, you can enter a semi-colon (;) as a line-break character in the header or footer text box. Left aligning multiple lines requires a left angle bracket at the beginning of each line.
12. Click the **More** tab.

13. Select the appropriate option in the **Scope** panel:
    - **All**
    - **First**
    - **Next**
    - **While**

    Show me more

All	This option is selected by default. Leave it selected to specify that all records in the view are processed.
First	Select this option and enter a number in the text box to start processing at the first record in the view and include only the specified number of records.
Next	Select this option and enter a number in the text box to start processing at the currently selected record in the view and include only the specified number of records. The actual record number in the leftmost column must be selected, not data in the row.
While	Select this option to use a WHILE statement to limit the processing of records in the view based on a particular criterion or set of criteria. You can enter a condition in the **While** text box, or click **While** to create a WHILE statement using the **Expression Builder**. A WHILE statement allows records in the view to be processed only while the specified condition evaluates to true. As soon as the condition evaluates to false, the processing terminates, and no further records are considered. You can use the **While** option in conjunction with the **All**, **First**, or **Next** options. Record processing stops as soon as one limit is reached.

> **Note**
> The number of records specified in the **First** or **Next** options references either the physical or the indexed order of records in a table, and disregards any filtering or quick sorting applied to the view. However, results of analytical operations respect any filtering.
> If a view is quick sorted, **Next** behaves like **First**.

14. If you selected **File (Analytics Table)** as the output type, select **Use Output Table** if you want the output table to open automatically upon completion of the operation.
15. If you selected **File** as the output type, and want to append the output results to the end of an existing Analytics table, do one of the following:
    - Select **Append To Existing File** if you are certain the output results and the existing table are identical in structure.
    - Leave **Append To Existing File** deselected if you want Analytics to compare the record lengths of the output results and the existing table. If the record lengths are not identical, the data structure is not identical, and the append will not work correctly.

> **Note**
>
> Leaving **Append To Existing File** deselected is recommended if you are uncertain whether the output results and the existing table have an identical data structure. For more information about appending and data structure, see "Appending output results to an existing table" on page 200.

16. Click **OK**.
17. If the overwrite prompt appears, select the appropriate option.

    If you are expecting the **Append** option to appear and it does not, click **No** to cancel the operation and see "Appending output results to an existing table" on page 200.

Cross-tabulating data

Cross-tabulating groups the records in a table based on identical combinations of values in two or more key fields, and counts the number of records in each group. Key fields can be character or numeric.

The resulting groups are displayed in a grid of rows and columns, similar to a pivot table, that allows you to visualize relations and patterns in the data.

For example, you could cross-tabulate an inventory table on the **Product Location** and **Product Class** fields to find the number of records in each class at each location.

Key field 1 (Product Location)	Key field 2 (Product Class)	Cross-tabulated group (Product Location and Product Class)		Record count
A-01	17	A-01	16	1
F-19	22	A-01	17	3
F-19	08	B-03	17	2
A-01	16	F-19	22	2
B-03	17	F-19	08	1
Q-28	03	Q-28	03	1
A-01	17			
F-19	22			
A-01	17			
B-03	17			

Subtotaling numeric fields

When cross-tabulating, you can optionally subtotal one or more numeric fields. In the example above, you could subtotal the inventory value field to find the total inventory value for each product class at each location.

Cross-tabulating is similar to summarizing

Cross-tabulating is similar to summarizing using two fields. In both operations the counts and subtotals in the output results are the same, but the information is arranged differently.

Cross-tabulating also displays counts and subtotals of zero, which summarizing does not. Depending on the type of analysis you are doing, displaying counts and subtotals of zero can be useful.

Cross-tabulating and sorting

You can cross-tabulate sorted or unsorted tables. When you cross-tabulate an unsorted table Analytics automatically sorts the output results as part of the cross-tabulate operation.

Cross-tabulating in detail

Cross-tabulating performs the following operations:

Operation	Location in "Cross-tabulate results" below
Groups the records based on identical combinations of values in two or more character or numeric fields, and displays the groups in a grid of rows and columns	intersections of **Cust Number** field (rows) and **Type** field (columns)
Optionally subtotals the values of one or more numeric fields for each group	**Amount** field
Optionally counts (subtotals) the number of records for each group **Note** Counts are automatically included if you do not select any subtotal fields.	**Count** field
Provides totals for all columns included in the output results	**Totals** row

Cross-tabulate results

Output results produced by:

- cross-tabulating customer number and transaction type in an accounts receivable table (the **Ar** table in `ACL DATA\Sample Data Files\Sample Project.ACL`)
- subtotaling transaction amount
- outputting the results to screen

Cust Number	Amount Type CN	Count CN	Amount Type IN	Count IN	Amount Type PM	Count PM
051593	-73.40	1	1,189.11	1	0.00	0
056016	0.00	0	1,807.66	2	-1,807.66	2
065003	-685.59	13	105,020.57	87	-8,443.97	15
081559	0.00	0	1,779.07	1	0.00	0
090398	0.00	0	634.38	2	0.00	0
097627	0.00	0	1,301.83	1	0.00	0
113236	0.00	0	681.93	1	0.00	0
176437	-241.49	3	14,825.62	13	-1,779.01	2
202028	-26.60	1	1,767.74	3	0.00	0
207275	0.00	0	3,678.68	3	0.00	0
Totals	-1,027.08	18	132,686.59	114	-12,030.64	19

Steps

You can cross-tabulate data by grouping the records in a table based on identical combinations of values in two or more character or numeric fields.

The resulting groups are displayed in a grid of rows and columns, similar to a pivot table, that allows you to visualize relations and patterns in the data.

Show me how

1. Select **Analyze > Cross-tab**.
2. On the **Main** tab, do one of the following:
   - Select the field(s) to display as rows from the **Rows** list.
   - Click **Rows** to select the field(s), or to create an expression.

   If you select more than one field you create an additional level of nesting in the output results. (Cross-tabulating using one row and one column is already a form of nesting.) The order in which you select the fields dictates the nesting hierarchy. The records are cross-tabulated by the first field you select, and within each of these primary groupings the records are then cross-tabulated by the second field you select, and so on. Reversing the order in which you select two fields gives quite different results.

   The order in which you select the fields is also the order in which they appear in the results. If you are appending results to an existing Analytics table, the column selection and order must be identical to the column selection and order in the existing table.
3. Do one of the following:
   - Select the field to display as columns from the **Columns** drop-down list.
   - Click **Columns** to select the field, or to create an expression.

4. Optional. Select one or more **Subtotal Fields**, or click **Subtotal Fields**, to select the subtotal field(s), or to create an expression.

   The order in which you select the subtotal fields is the order in which the columns appear in the results. If you are appending results to an existing Analytics table, the column selection and order must be identical to the column selection and order in the existing table.

5. If there are records in the current view that you want to exclude from processing, enter a condition in the **If** text box, or click **If** to create an IF statement using the **Expression Builder**.

   > **Note**
   > The **If** condition is evaluated against only the records remaining in a table after any scope options have been applied (**First**, **Next**, **While**).

   The IF statement considers all records in the view and filters out those that do not meet the specified condition.

6. If you want to include a count of the number of records for each row-column intersection, select **Include Count**.

   A count is automatically included if you do not select any subtotal fields.

7. Click the **Output** tab.
8. Select the appropriate output option in the **To** panel:
   - **Screen** – Select this option to display the results in the Analytics display area.

     > **Tip**
     > You can click any linked result value in the display area to drill down to the associated record or records in the source table.

     If the output table contains a large number of records, it is faster and more useful to save the results to a file than to display the results on the screen.

   - **Print** – Select this option to send the results to the default printer.
   - **Graph** – Select this option to create a graph of the results and display it in the Analytics display area.
   - **File** – Select this option to save or append the results to an Analytics table or a text file. If you save or append to an Analytics table, the table is added to the open project if it is not already in the project. If you save or append to a text file, the file is saved outside Analytics.

     > **Note**
     > Output options that do not apply to a particular analytical operation are disabled.

9. If you selected **File** as the output type, specify the following information in the **As** panel:
   - **File Type** – Select **Analytics Table** to save the results to a new Analytics table, or append the results to an existing Analytics table. Select **ASCII Text File** or **Unicode Text file** (depending on which edition of Analytics you are using) to save or append the results to a text file.
   - **Name** – Enter a file name in the **Name** text box. Or click **Name** and enter the file name, or select an existing file in the **Save** or **Save File As** dialog box to overwrite or append to the

file. If Analytics prefills a file name, you can accept the prefilled name, or change it.

You can also specify an absolute or relative file path, or navigate to a different folder, to save or append the file in a location other than the project location. For example: `C:\Results\Output.fil` or `Results\Output.fil`.

- **Local** – Only enabled when connected to a server table and saving or appending the results to an Analytics table. Select **Local** to save the file to the same location as the project, or to specify a path or navigate to a different local folder. Leave **Local** deselected to save the file to the Prefix folder on a server.

> **Note**
> For output results produced from analysis or processing of AX Server tables, select **Local**. You cannot deselect the **Local** setting to import results tables to AX Server.

10. Depending on the output type, you can optionally specify a **Header** and/or a **Footer** in the text box(es).

    Headers and footers are centered by default. Type a left angle bracket (<) before the header or footer text to left align the text. Click **Header** or **Footer** to enter a header or footer of more than one line. Alternatively, you can enter a semi-colon (;) as a line-break character in the header or footer text box. Left aligning multiple lines requires a left angle bracket at the beginning of each line.

11. Click the **More** tab.

12. Select the appropriate option in the **Scope** panel:

    - **All**
    - **First**
    - **Next**
    - **While**

    Show me more

All	This option is selected by default. Leave it selected to specify that all records in the view are processed.
First	Select this option and enter a number in the text box to start processing at the first record in the view and include only the specified number of records.
Next	Select this option and enter a number in the text box to start processing at the currently selected record in the view and include only the specified number of records. The actual record number in the leftmost column must be selected, not data in the row.
While	Select this option to use a WHILE statement to limit the processing of records in the view based on a particular criterion or set of criteria. You can enter a condition in the **While** text box, or click **While** to create a WHILE statement using the **Expression Builder**. A WHILE statement allows records in the view to be processed only while the specified condition evaluates to true. As soon as the condition evaluates to false, the processing terminates, and no further records are considered. You can use the **While** option in conjunction with the **All**, **First**, or **Next** options. Record processing stops as soon as one limit is reached.

> **Note**
> The number of records specified in the **First** or **Next** options references either the physical or the indexed order of records in a table, and disregards any filtering or quick sorting applied to the view. However, results of analytical operations respect any filtering.
> If a view is quick sorted, **Next** behaves like **First**.

13. If you selected **File (Analytics Table)** as the output type, select **Use Output Table** if you want the output table to open automatically upon completion of the operation.
14. If you selected **File** as the output type, and want to append the output results to the end of an existing file, do one of the following:
    - Select **Append To Existing File** if you are appending to a text file, or to an Analytics table that you are certain is identical in structure to the output results.
    - Leave **Append To Existing File** deselected if you are appending to an Analytics table and you want Analytics to compare the record lengths of the output results and the existing table. If the record lengths are not identical, the data structure is not identical, and the append will not work correctly.

    > **Note**
    > Leaving **Append To Existing File** deselected is recommended if you are uncertain whether the output results and the existing table have an identical data structure. For more information about appending and data structure, see "Appending output results to an existing table" on page 200.

15. Click **OK**.
16. If the overwrite prompt appears, select the appropriate option.

    If you are expecting the **Append** option to appear and it does not, click **No** to cancel the operation and see "Appending output results to an existing table" on page 200.

Creating histograms

Creating a histogram groups the records in a table, counts the number of records in each group, and displays the groups and counts in a vertical bar chart.

You can group the records:

- Based on identical values in a character field (similar to classifying)
- Into equal-sized, or custom-sized, numeric intervals (similar to stratifying)

Break field available for some output formats

In addition to a bar chart, you can also output the results to screen, to a text file, or to print. When outputting results in these formats, you can optionally break down the results based on the values in a character field, such as customer ID or transaction type. For this option to yield meaningful results, the character field you use for the breakdown must be sorted prior to creating the histogram.

Subtotaling numeric fields

Unlike the other grouping operations in Analytics, histograms do not support subtotaling numeric fields.

Histograms and sorting

You can create a histogram using sorted or unsorted tables. When you use an unsorted table Analytics automatically sorts the output results in ascending order as part of creating the histogram.

Steps

You can create a histogram that groups the records in a table and displays the groups in a bar chart.

You can group the records:

- Based on identical values in a character field
- Into equal-sized, or custom-sized, numeric intervals

Show me how

1. Select **Analyze > Histogram**.
2. On the **Main** tab, do one of the following:
   - Select the field on which to base the histogram from the **Histogram On** drop-down list.
   - Click **Histogram On** to select the field, or to create an expression.
3. If you selected a numeric field or expression in **Histogram On**, do the following:
   a. In **Minimum**, enter the minimum value of the first interval.

      If you previously performed a profile or statistics operation on the numeric field, the lowest value in the field is automatically entered by default. You can change the default, if required.

   b. In **Maximum**, enter the maximum value of the last interval.

      If you previously performed a profile or statistics operation on the numeric field, the highest value in the field is automatically entered by default. You can change the default, if required.
4. If you selected a numeric field or expression in **Histogram On**, do one of the following:
   - Select **Intervals**, and enter the number of equal-sized intervals that you want in the range specified by the **Minimum** and **Maximum** values. The default number of intervals is 10.

     > **Tip**
     > You can change the default number of intervals by selecting **Tools > Options** and updating the **Intervals** number on the **Command** tab.

   - Select **Free** to create custom-sized intervals, and enter the start value of each interval and the end value of the last interval. You must enter each value on a separate line.

     Specifying **Minimum** and **Maximum** values is optional when you use **Free**. If you do specify **Minimum** and **Maximum** values, those values are the start point of the first interval and the end point of the last interval, and the values you enter create additional intervals within the range. The values you enter must be greater than the value specified in **Minimum**, and equal to or less than the value specified in **Maximum**.
5. If there are records in the current view that you want to exclude from processing, enter a condition in the **If** text box, or click **If** to create an IF statement using the **Expression Builder**.

   > **Note**
   > The **If** condition is evaluated against only the records remaining in a table after any scope options have been applied (**First**, **Next**, **While**).

   The IF statement considers all records in the view and filters out those that do not meet the specified condition.
6. Click the **Output** tab.

7. Select the appropriate output option in the **To** panel:
   - **Screen** – Select this option to display the results in the Analytics display area.

     > **Tip**
     > You can click any linked result value in the display area to drill down to the associated record or records in the source table.

     If the output table contains a large number of records, it is faster and more useful to save the results to a file than to display the results on the screen.
   - **Print** – Select this option to send the results to the default printer.
   - **Graph** – Select this option to create a graph of the results and display it in the Analytics display area.
   - **File** – Select this option to save or append the results to a text file. The file is saved outside Analytics.

     > **Note**
     > Output options that do not apply to a particular analytical operation are disabled.

8. If you selected **File** as the output type, specify the following information in the **As** panel:
   - **File Type** – **ASCII Text File** or **Unicode Text file** (depending on which edition of Analytics you are using) is the only option. Saves the results to a new text file, or appends the results to an existing text file.
   - **Name** – Enter a file name in the **Name** text box. Or click **Name** and enter the file name, or select an existing file in the **Save** or **Save File As** dialog box to overwrite or append to the file. If Analytics prefills a file name, you can accept the prefilled name, or change it.

     You can also specify an absolute or relative file path, or navigate to a different folder, to save or append the file in a location other than the project location. For example: `C:\Results\Output.txt` or `Results\Output.txt`.
   - **Local** – Disabled and selected. Saving the file locally is the only option.
9. Depending on the output type, you can optionally specify a **Header** and/or a **Footer** in the text box(es).

   Headers and footers are centered by default. Type a left angle bracket (<) before the header or footer text to left align the text. Click **Header** or **Footer** to enter a header or footer of more than one line. Alternatively, you can enter a semi-colon (;) as a line-break character in the header or footer text box. Left aligning multiple lines requires a left angle bracket at the beginning of each line.
10. Click the **More** tab.
11. Select the appropriate option in the **Scope** panel:
    - **All**
    - **First**
    - **Next**
    - **While**

Show me more

All	This option is selected by default. Leave it selected to specify that all records in the view are processed.
First	Select this option and enter a number in the text box to start processing at the first record in the view and include only the specified number of records.
Next	Select this option and enter a number in the text box to start processing at the currently selected record in the view and include only the specified number of records. The actual record number in the leftmost column must be selected, not data in the row.
While	Select this option to use a WHILE statement to limit the processing of records in the view based on a particular criterion or set of criteria. You can enter a condition in the **While** text box, or click **While** to create a WHILE statement using the **Expression Builder**. A WHILE statement allows records in the view to be processed only while the specified condition evaluates to true. As soon as the condition evaluates to false, the processing terminates, and no further records are considered. You can use the **While** option in conjunction with the **All**, **First**, or **Next** options. Record processing stops as soon as one limit is reached.

> **Note**
> The number of records specified in the **First** or **Next** options references either the physical or the indexed order of records in a table, and disregards any filtering or quick sorting applied to the view. However, results of analytical operations respect any filtering.
> If a view is quick sorted, **Next** behaves like **First**.

12. If you do not want to include values that exceed the specified **Minimum** and **Maximum** values, select **Suppress Others**.
13. Optional. If you are outputting histogram results to a text file, specify the length of the x-axis in the textual representation of the bar chart by entering a number in **Columns**.

    The number you enter specifies the number of character spaces (text columns) to use for the x-axis (and the y-axis labels). In most cases, you can leave **Columns** blank to use the default of 78 character spaces.
14. Optional. If you are outputting histogram results to screen, a file, or printer, enter the name of a break field in the **Break** text box, or click **Break** to select the field, or to create an expression.

    For example, a histogram of an accounts receivable table could be broken down by customer. **Break** can only be used with a single character field, so nested breakdowns are not supported.

    > **Note**
    > For the **Break** option to yield meaningful results, the character field must be sorted prior to creating a histogram.

15. If you selected **File** as the output type, and want to append the output results to the end of an existing text file, select **Append To Existing File**.
16. Click **OK**.

17. If the overwrite prompt appears, select the appropriate option.

Machine learning analysis

Machine learning in Analytics is automated (AutoML). Complex computational work such as data preprocessing, algorithm selection, hyperparameter tuning, and model validation is performed for you by Analytics. This automation allows you to put machine learning to work on company data with relatively little effort, and without requiring that you have specialized data science capabilities.

Supervised and unsupervised machine learning

Analytics supports both supervised and unsupervised machine learning.

Supervised machine learning uses existing data labeled with categories or numeric values as the basis for predicting categories or numeric values in similar, unlabeled data.

Unsupervised machine learning discovers categories in uncategorized or unlabeled data.

Machine learning not supported on 32-bit computers

If you install Analytics on a 32-bit computer, the machine learning operations are not supported, and the **Machine Learning** menu does not appear. The computation required by machine learning is processor-intensive and better suited to 64-bit computers.

Machine learning operations

Operation	ML type	Supported data types	Functionality	Output
Train	Supervised	Character Numeric Datetime Logical	◦ Uses automated machine learning to create a predictive model.	◦ Predictive model file (`*.model`) ◦ Model evaluation file (Analytics table)
Predict	Supervised	Character	◦ Applies a predictive model to an	◦ Results file

Operation	ML type	Supported data types	Functionality	Output
		Numeric Datetime Logical	unlabeled data set to predict classes or numeric values.	(Analytics table)
Cluster	Unsupervised	Numeric	◦ Groups numeric data. ◦ Groups records based on similar, or nearby, values in one or more numeric fields.	◦ Results file (Analytics table)

Predicting classes and numeric values

Use automated machine learning in Analytics to predict classes or numeric values associated with unlabeled data. Data is unlabeled if the classes or numeric values you are interested in do not exist in the data. For example, you could use machine learning to predict loan defaults, or future house prices:

Prediction problem	Prediction type	Description
Loan defaults	Classification	Based on applicant information such as age, job category, credit score, and so on, predict which applicants will default if given a loan. Put another way, will applicants fall into the class of Default = Yes, or Default = No?
Future house prices	Regression	Based on features such as age, square footage, ZIP code, number of bedrooms and bathrooms, and so on, predict the future sale price of houses.

Automated machine learning

Machine learning in Analytics is "automated" because two related commands – Train and Predict – perform all the computational work associated with training and evaluating a predictive model, and applying the predictive model to an unlabeled data set. The automation provided by Analytics allows you to put machine learning to work on company data without requiring that you have specialized data science capabilities.

The train and predict workflow

The train and predict workflow is composed of two related processes, and two related data sets:

- **Training process** – uses a training data set (labeled)
- **Prediction process** – uses a new data set (unlabeled)

Training process

The training process is performed first, using a training data set that includes a **labeled field** (also called a **target field**).

The labeled field contains the known class, or the known numeric value, associated with each record in the training data set. For example, whether a borrower defaulted on a loan (Y/N), or the sale price of a house.

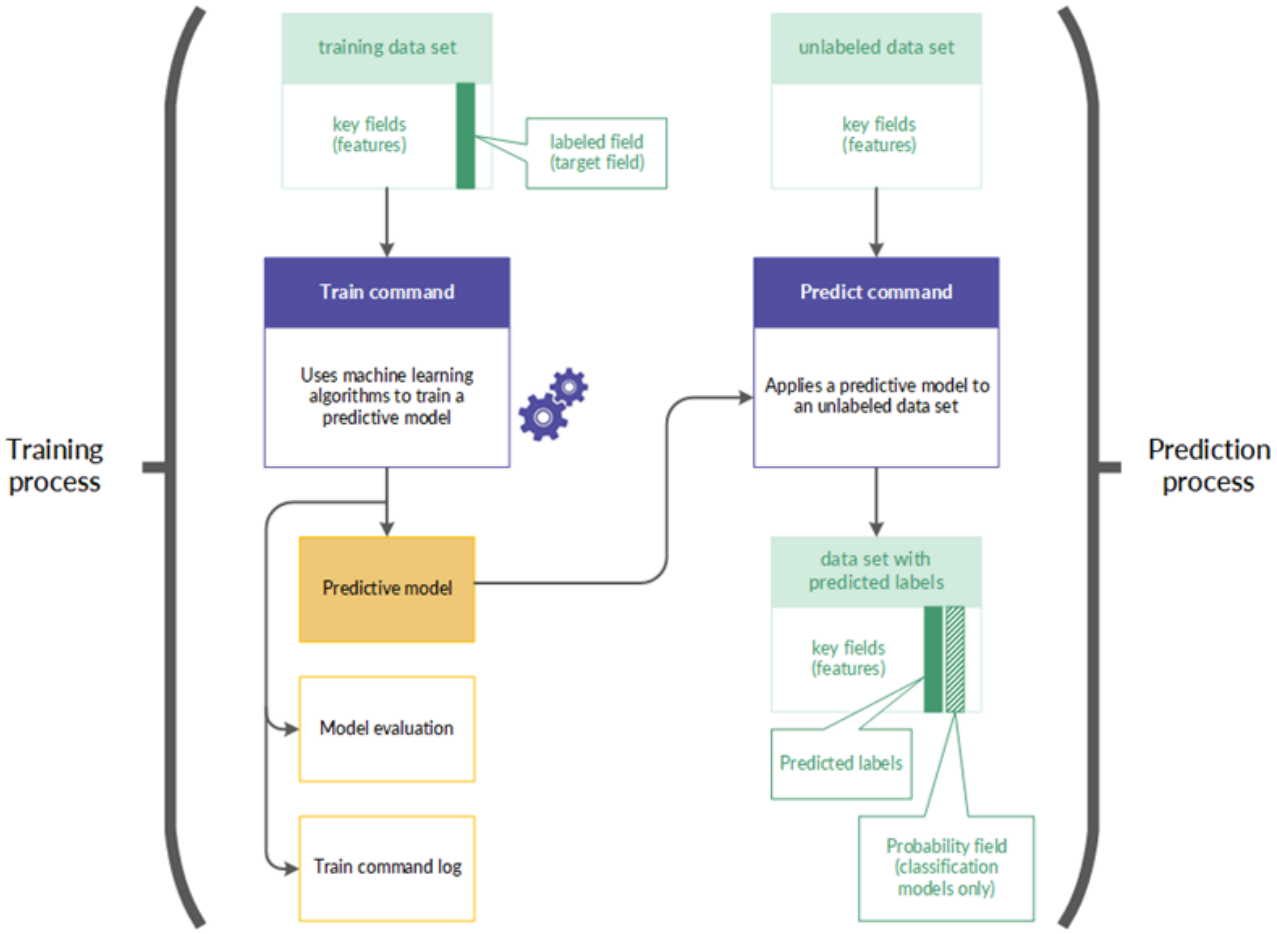
Using machine learning algorithms, the training process generates a **predictive model**. The training process generates a number of different model permutations in order to discover the model that is best suited to the predictive task you are performing.

Prediction process

The prediction process is performed second. It applies the predictive model generated by the training process to a new, unlabeled data set that contains data similar to the data in the training data set.

Label values such as loan default information or house sale price do not exist in the new data set because these are future events.

Using the predictive model, the prediction process predicts a class or a numeric value associated with each unlabeled record in the new data set.



The train and predict workflow in more detail

	Process	Description	Data set examples
1	**Training** (Train command)	◦ **Train command** - You run the train command against a training data set to train a predictive model. The command uses several different machine learning algorithms to generate numerous models before selecting a single model best suited to the predictive task ("the winning model"). ◦ **Training data set** - The data set includes key fields (features) and a labeled field (target field). ◦ **Learning** - The training process builds a mathematical model that represents the relations between the key fields and the labeled field. ◦ **Example** - For example, with all other features equal, the training process might find that a fourth bedroom increased the selling price of a house by $35,000.	◦ **Loan data** - historical loan data, including loan default information (Y/N) "Default" is the labeled field. ◦ **House data** - recent house sales data, including the sale price "Sale price" is the labeled field.

		"Number of bedrooms" is a key field, and "sale price" is the labeled field. ◦ **Predictive model** - The training process stores the predictive model in an output file.	
2	**Prediction** (Predict command)	◦ **Predict command** - You use the predict command to apply the predictive model produced by the train command. ◦ **New data** - You apply the model to a new data set with the same key fields (features) as the training data set, but without the labeled field. ◦ **Predictions** - The prediction process uses the mathematical relations stored in the predictive model to predict label values for similar key field relations in the new data set. ◦ **Example** - For example, with all other features equal, the prediction process predicts a sale price of $400,000 for a three-bedroom house, and $435,000 for a four-bedroom house. ◦ **Probability** - (classification only) For each predicted value, the prediction output includes the probability or confidence that the prediction is accurate.	◦ **Loan data** - current loan applicant data Loan default information does not yet exist because loans are only at the application stage. ◦ **House data** - house price evaluation data Recent sale price data does not exist because the houses are not yet on the market.

Processing time

The computation required by machine learning is time consuming and processor-intensive. Training a predictive model using a large data set with numerous fields can take hours, and is typically a task you might run overnight.

Including datetime key fields in the training process is particularly processor-intensive because each datetime field is used to automatically derive 10 synthetic features. Datetime synthetic features can significantly expand the scope of the predictive data, but you should only include datetime fields if you think that they might be relevant.

> **Tip**
> If you are just familiarizing yourself with machine learning in Analytics, use small data sets so that you can keep processing times manageable, and see results relatively quickly.

Strategies for reducing the size of the training data set

You can use different strategies to reduce the size of the training data set, and the associated processing time, without significantly affecting the accuracy of the resulting predictive model.

- Exclude fields from the training process that do not contribute to predictive accuracy. Exclude irrelevant fields, and redundant fields.

- Exclude datetime fields from the training process if they do not contribute to predictive accuracy. Although, be careful about assuming that datetime fields are not relevant. For more information, see "Datetime key fields" below.
- Sample the training data set and use the sampled data as input for the training process. Possible sampling approaches include:
  - balancing the size of data classes by sampling majority classes to approximate average minority class size
  - random sampling of the entire training data set
  - stratified sampling based on features
  - stratified sampling based on clustering

Datetime key fields

You can use one or more datetime fields as key fields when training a predictive model. Typically, too many unique values exist in a datetime field for the field to be a suitable source of categories or identifiable features for the training process. Raw datetime data may also appear unrelated to your target field of interest.

However, once categorized, datetime data may have relevance. For example, events that you are examining may have a pattern of occurring on certain days of the week, or during certain times of the day.

The training process automatically derives a number of **synthetic features** from each datetime field by categorizing the raw datetime data. These synthetic features are then included in the algorithm that generates a **predictive model**.

Synthetic features derived from datetime fields

The synthetic features automatically derived from date, time, or datetime fields are listed below.

Description of synthetic feature	Feature type	Synthetic feature name
Day of the week	Numeric (1 to 7)	*fieldname*_DOW
Month of the year	Numeric (1 to 12)	*fieldname*_MONTH
Quarter	Numeric (1 to 4)	*fieldname*_QTR
Number of days since the start of the month	Numeric (1 to 31)	*fieldname*_DAY
Number of days since the start of the year	Numeric (1 to 366)	*fieldname*_DOY
Seconds	Numeric (00 to 59)	*fieldname*_SECOND
Hour of the day	Numeric (1 to 24)	*fieldname*_HOUR

Description of synthetic feature	Feature type	Synthetic feature name
Number of seconds since the start of the day	Numeric (1 to 86400)	*fieldname*_SOD
Quartile of the day	Categorical: ◦ 00:00-06:00 ◦ 06:00-12:00 ◦ 12:00-18:00 ◦ 18:00-24:00	*fieldname*_QOD
Octile of the day	Categorical: ◦ 00:00-03:00 ◦ 03:00-06:00 ◦ 06:00-09:00 ◦ 09:00-12:00 ◦ 12:00-15:00 ◦ 15:00-18:00 ◦ 18:00-21:00 ◦ 21:00-24:00	*fieldname*_OOD

Training a predictive model

> **Note**
> The maximum supported size of the data set used with the training process is 1 GB.
> If the machine learning menu options are disabled, the Python engine is probably not installed. For more information, see "Install ACL for Windows" on page 2583.

Steps

Specify basic settings for the training process

1. Open the Analytics table with the training data set.
2. From the Analytics main menu, select **Machine Learning > Train**.
3. Specify the time allotted to the training process:

Time to search for an optimal model	The total time in minutes to spend generating and testing predictive models, and selecting a winning model. Specify a search time that is at least 10x the maximum evaluation time per model.
Maximum time per model evaluation	Maximum runtime in minutes per model evaluation. Allot 45 minutes for every 100 MB of training data.

> **Note**
> The total runtime of the training process is the search time plus up to twice the maximum model evaluation time.
>
> The suggested time allotments strike a reasonable balance between processing time and allowing a variety of model types to be evaluated.

4. Specify the prediction type to use:
   - **Classification** – use classification algorithms to train a model

     Use classification if you want to predict which class or category records in an unlabeled data set belong to.
   - **Regression** – use regression algorithms to train a model

     Use regression if you want to predict numeric values associated with records in an unlabeled data set.

   For information about the specific algorithms used with classification and regression, see "Training algorithms" on page 1302.
5. In the **Model scorer** drop-down list, select the metric to use when scoring the models generated during the training process.

   The generated model with the best value for this metric is kept, and the rest of the models are discarded.

   A different subset of metrics is available depending on the prediction type you are using:

Classification	Log loss | AUC | Accuracy | F1 | Precision | Recall
Regression	Mean squared error | Mean absolute error | R2

> **Note**
> The classification metric **AUC** is only valid when used with a target field that contains binary data – that is, two classes, such as Yes/No, or True/False.

Select fields

1. From the **Train On** list, select one or more key fields to use as input when training the model.

   Key fields are the features that form the basis for predicting target field values in an unlabeled data set. Key fields can be character, numeric, datetime, or logical. Synthetic features are automatically derived from datetime key fields.

> **Note**
> Character fields must be "categorical". That is, they must identify categories or classes, and they cannot exceed a maximum number of unique values.
>
> The maximum is specified by the **Maximum Categories** option (**Tools > Options > Command**).

> **Tip**
> You can **Ctrl+click** to select multiple non-adjacent fields, and **Shift+click** to select multiple adjacent fields.

2. From the **Target Field** list, select the target field.

   The target field is the field that the model is being trained to predict based on the input key fields.

   Classification and regression work with different target field data types:

   - **classification** – a character or logical target field
   - **regression** – a numeric target field

Name the model file and the output ACL table

1. In the **Model Name** text box, specify the name of the model file output by the training process.

   The model file contains the model best fitted to the training data set. You will input the model file to the prediction process to generate predictions about a new, unseen data set.

2. In the **To** text box, specify the name of the model evaluation table output by the training process.

   The model evaluation table contains two distinct types of information:

   - **Scorer/Metric** – for the classification or regression metrics, quantitative estimates of the predictive performance of the model file output by the training process
   - **Importance/Coefficient** – in descending order, values indicating how much each feature (predictor) contributes to the predictions made by the model

3. If there are records in the current view that you want to exclude from processing, enter a condition in the **If** text box, or click **If** to create an IF statement using the **Expression Builder**.

   > **Note**
   > The **If** condition is evaluated against only the records remaining in a table after any scope options have been applied (**First**, **Next**, **While**).

   The IF statement considers all records in the view and filters out those that do not meet the specified condition.

Specify that only a subset of the training data set is used (optional)

On the **More** tab, select one of the options in the **Scope** panel:

All (default)	All records in the table are processed.
First	Select this option and enter a number in the text box to start processing at the first record in the table and include only the specified number of records.
Next	Select this option and enter a number in the text box to start processing at the currently selected record in the table view and include only the specified number of records.

	The actual record number in the leftmost column must be selected, not data in the row.
While	Select this option to use a WHILE statement to limit the processing of records in the table based on criteria.

Specify advanced settings for the training process

1. On the **More** tab, specify the **Number of cross-validation folds**.

   Leave the default number of 5, or specify a different number. Valid numbers are from 2 to 10.

   Folds are subdivisions of the training data set, and are used in a cross-validation process during model evaluation and optimization.

   Typically, using from 5 to 10 folds yields good results when training a model.

   > **Tip**
   > Increasing the number of folds can produce a better estimate of the predictive performance of a model, but it also increases overall runtime.

2. Optional. Select **Seed**, and enter a number.

   The seed value is used to initialize the random number generator in Analytics.

   If you do not select **Seed**, Analytics randomly selects the seed value.

   Explicitly specify a seed value, and record it, if you want to replicate the training process with the same data set in the future.

3. Optional. If you want to train and score only linear models, select **Only evaluate linear models**.

   If you leave this option unselected, all model types relevant to classification or regression are evaluated.

   > **Note**
   > With larger data sets, the training process typically completes more quickly if you include only linear models.
   > Including only linear models guarantees coefficients in the output.

4. Optional. Select **Disable feature selection and preprocessing** if you want to exclude these subprocesses from the training process.

   Feature selection is the automated selection of the fields in the training data set that are the most useful in optimizing the predictive model. Automated selection can improve predictive performance, and reduce the amount of data involved in model optimization.

   Data preprocessing performs transformations such as scaling and standardizing on the training data set to make it better suited for the training algorithms.

> **Caution**
> You should only disable feature selection and data preprocessing if you have a reason for doing so.

5. Click **OK**.

   The training process launches, and a dialog box appears that shows the input settings you specified, and elapsed processing time.

Applying a predictive model to an unlabeled data set

> **Note**
> If the machine learning menu options are disabled, the Python engine is probably not installed. For more information, see "Install ACL for Windows" on page 2583.

Steps

1. Open the Analytics table with the unlabeled data set.
2. From the Analytics main menu, select **Machine Learning > Predict**.
3. Click **Model**, in the **Select File** dialog box select a model file output by a previous training process, and click **Open**.

   Model files have a `*.model` file extension.

   > **Note**
   > The model file must have been trained on a data set with the same fields as the unlabeled data set – or substantially the same fields.
   >
   > You cannot use a model file trained in version 14.1 of Analytics. Version 14.1 model files are not compatible with subsequent versions of Analytics. Train a new predictive model to use with the prediction process.

4. In the **To** text box, specify the name of the Analytics table output by the prediction process.

   The output table contains the key fields you specified during the training process, and either one or two fields generated by the prediction process:

   - **Predicted** – the predicted classes or numeric values associated with each record in the unlabeled data set
   - **Probability** – (classification only) the probability that a predicted class is accurate

5. If there are records in the current view that you want to exclude from processing, enter a condition in the **If** text box, or click **If** to create an IF statement using the **Expression Builder**.

> **Note**
> The **If** condition is evaluated against only the records remaining in a table after any scope options have been applied (**First**, **Next**, **While**).

The IF statement considers all records in the view and filters out those that do not meet the specified condition.

6. Optional. To process only a subset of the unlabeled data set, on the **More** tab select one of the options in the **Scope** panel.
7. Click **OK**.

Training algorithms

Three train command options dictate which machine learning algorithms are used for training a predictive model:

Option	Train dialog box tab
Classification or **Regression**	**Main** tab
Only evaluate linear models	**More** tab
Disable feature selection and preprocessing	**More** tab

The sections that follow summarize how the options control which algorithms are used.

The names of the algorithms do not appear in the Analytics user interface. The name of the algorithm used for generating the model ultimately selected by the train command appears in the log.

Classification algorithms

Show me more

✅ **Algorithm used** ⊗ **Algorithm not used**

Algorithm name	Always included	Only evaluate linear models		Disable feature selection and preprocessing	
		Option not selected (default)	Option selected	Option not selected (default)	Option selected
Algorithm type: Classifier					
Logistic Regression	✅				
Linear Support Vector Machine	✅				

Algorithm name	Always included	Only evaluate linear models		Disable feature selection and preprocessing	
		Option not selected (default)	Option selected	Option not selected (default)	Option selected
Random Forest		✓	✗		
Extremely Randomized Trees		✓	✗		
Gradient Boosting Machine		✓	✗		
Algorithm type: Feature preprocessor					
One Hot Encoding - of categorical features	✓				
Fast Independant Component Analysis				✓	✗
Feature Agglomeration				✓	✗
Principal Component Analysis - Singular Value Decomposition				✓	✗
Second Degree Polynomial Features				✓	✗
Binarizer				✓	✗
Robust Scaler				✓	✗
Standard Scaler				✓	✗
Maximum Absolute Scaler				✓	✗
Min Max Scaler				✓	✗
Normalizer				✓	✗
Nystroem Kernel Approxim-ation				✓	✗
RBF Kernel Approximation				✓	✗
Zero Counter				✓	✗

Algorithm name	Always included	Only evaluate linear models		Disable feature selection and preprocessing	
		Option not selected (default)	Option selected	Option not selected (default)	Option selected
Algorithm type: Feature selector					
Family-wise Error Rate				✓	✗
Percentile of Highest Scores				✓	✗
Variance Threshold				✓	✗
Recursive Feature Elimination				✓	✗
Importance Weights				✓	✗

Regression algorithms

Show me more

✓ **Algorithm used** ✗ **Algorithm not used**

Algorithm name	Always included	Only evaluate linear models		Disable feature selection and preprocessing	
		Option not selected (default)	Option selected	Option not selected (default)	Option selected
Algorithm type: Regressor					
Elastic Net	✓				
Lasso	✓				
Ridge	✓				
Linear Support Vector Machine	✓				
Random Forest		✓	✗		
Extremely Randomized Trees		✓	✗		

Algorithm name	Always included	Only evaluate linear models		Disable feature selection and preprocessing	
		Option not selected (default)	Option selected	Option not selected (default)	Option selected
Gradient Boosting Machine		✓	✗		
Algorithm type: Feature preprocessor					
One Hot Encoding - of categorical features	✓				
Fast Independant Component Analysis				✓	✗
Feature Agglomeration				✓	✗
Principal Component Analysis - Singular Value Decomposition				✓	✗
Second Degree Polynomial Features				✓	✗
Binarizer				✓	✗
Robust Scaler				✓	✗
Standard Scaler				✓	✗
Maximum Absolute Scaler				✓	✗
Min Max Scaler				✓	✗
Normalizer				✓	✗
Nystroem Kernel Approxim-ation				✓	✗
RBF Kernel Approximation				✓	✗
Zero Counter				✓	✗
Algorithm type: Feature selector					
Family-wise Error Rate				✓	✗

Algorithm name	Always included	Only evaluate linear models		Disable feature selection and preprocessing	
		Option not selected (default)	Option selected	Option not selected (default)	Option selected
Percentile of Highest Scores				✓	✗
Variance Threshold				✓	✗
Importance Weights				✓	✗

Clustering data

Clustering groups the records in a table based on similar values in one or more numeric key fields. Similar values are values that are nearby or close to one another in the context of the entire data set. These similar values represent clusters that, once identified, reveal patterns in the data.

> **Note**
> If you want to make clustering a regular part of your analysis program, we recommend taking the Galvanize Academy course Finding data groups using the CLUSTER command in Analytics (ACL 361) (customer log-in required).

How clustering differs from other Analytics grouping commands

Clustering differs from other Analytics grouping commands:

- Clustering does not require grouping on pre-existing data categories such as transaction type or merchant category code, or on predefined strata with hard numeric boundaries. Instead, clustering groups data based on similar numeric values within the data itself – that is, values that are close or nearby to one another.
- Clustering based on more than one field outputs results that are not nested (non-hierarchical).

Choosing the fields to cluster on

Clustering data allows you to discover organic groupings in the data that you otherwise might not know exist. In particular, clusters based on multiple numeric fields (multi-dimensional clusters) would be hard to identify without the assistance of machine learning. In this sense, clustering is exploratory, and an example of unsupervised machine learning.

However, for the output clusters to be meaningful, a meaningful relation must exist between the fields that you select for clustering.

Cluster on a single field

Clustering on a single numeric field is relatively straightforward. You focus on a single set of values, and clustering groups the values based on closeness between values, or proximity. For example, you can cluster an amount field to find out where the amounts are concentrated over the range of values.

The benefit of clustering over a traditional approach like stratifying is that you do not have to make any assumptions, in advance, about where the concentrations may exist, or create arbitrary numeric boundaries. Clustering discovers where the boundaries lie for any given number of clusters.

Show example

Example of clustering on a single numeric field

You cluster the Ap_Trans table on the **Invoice Amount** field to find out where amounts are concentrated over the range of values. Your expectation is that most of the amounts will be clustered at the lower end of the range. Clustering will confirm whether your expected pattern is in fact the case.

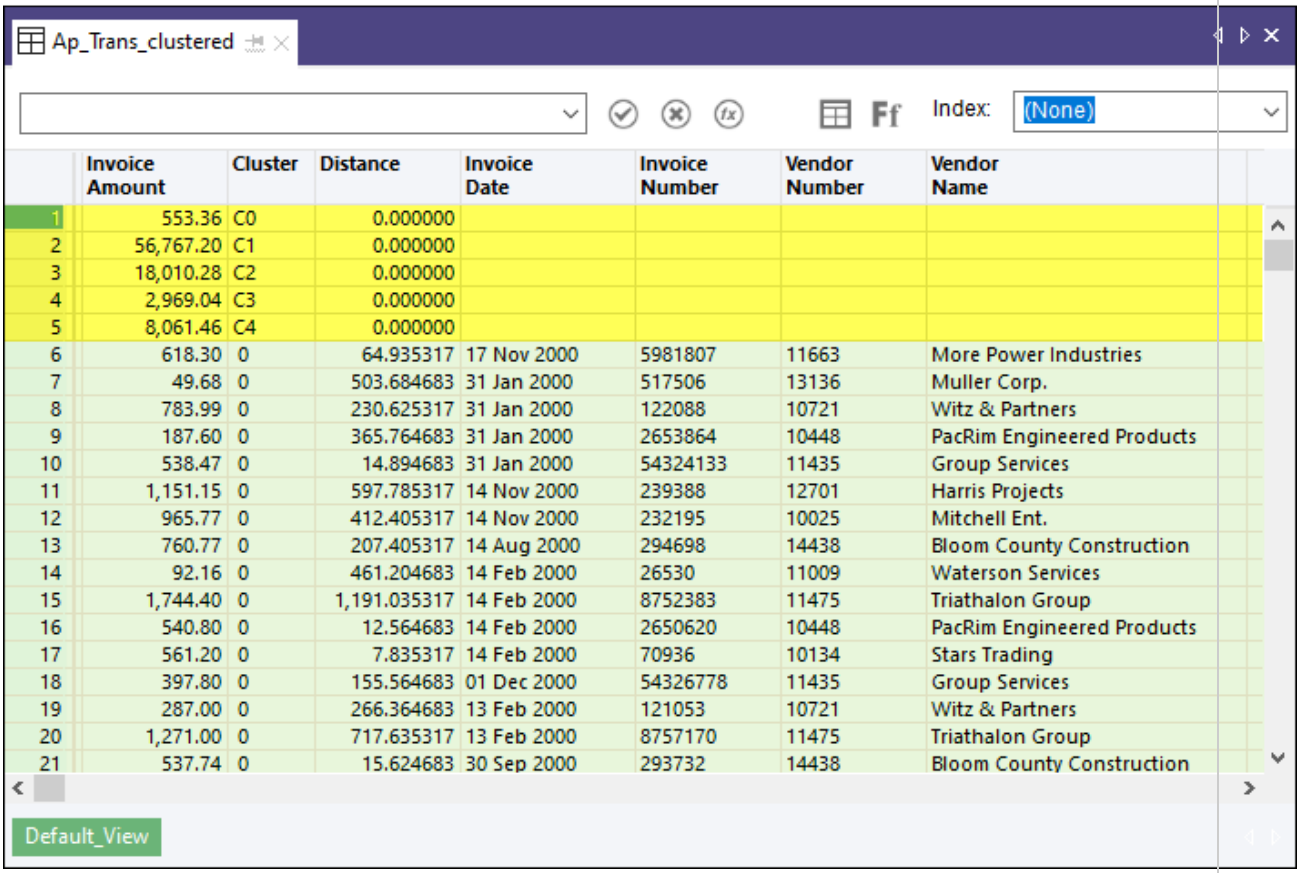
You decide to group the **Invoice Amount** field into five clusters, and then summarize the clusters to discover how many records are in each cluster.

The output results

In the output results shown below, the first five records are system-generated and equate to the desired number of clusters that you specified. In the **Invoice Amount** field, the five records show the **centroid**, or center point, that the clustering algorithm calculates for each of the five clusters of invoice amounts. For example, the centroid for cluster 3 (`C3`) is 2,969.04. For more information, see "How the clustering algorithm works" on page 1311.

Beneath the system-generated fields are the source data fields grouped into clusters, starting with cluster `0`. The value in the **Distance** field is the distance from the actual invoice amount to the calculated centroid value for that cluster. So, for example, in record 6, the invoice amount of 618.30 minus the distance of 64.935317 equals the centroid value of 553.36.

> **Note**
> You subtract or add the distance value, depending on whether the actual value is greater than or less than the centroid value.



	Invoice Amount	Cluster	Distance	Invoice Date	Invoice Number	Vendor Number	Vendor Name
1	553.36	C0	0.000000				
2	56,767.20	C1	0.000000				
3	18,010.28	C2	0.000000				
4	2,969.04	C3	0.000000				
5	8,061.46	C4	0.000000				
6	618.30	0	64.935317	17 Nov 2000	5981807	11663	More Power Industries
7	49.68	0	503.684683	31 Jan 2000	517506	13136	Muller Corp.
8	783.99	0	230.625317	31 Jan 2000	122088	10721	Witz & Partners
9	187.60	0	365.764683	31 Jan 2000	2653864	10448	PacRim Engineered Products
10	538.47	0	14.894683	31 Jan 2000	54324133	11435	Group Services
11	1,151.15	0	597.785317	14 Nov 2000	239388	12701	Harris Projects
12	965.77	0	412.405317	14 Nov 2000	232195	10025	Mitchell Ent.
13	760.77	0	207.405317	14 Aug 2000	294698	14438	Bloom County Construction
14	92.16	0	461.204683	14 Feb 2000	26530	11009	Waterson Services
15	1,744.40	0	1,191.035317	14 Feb 2000	8752383	11475	Triathalon Group
16	540.80	0	12.564683	14 Feb 2000	2650620	10448	PacRim Engineered Products
17	561.20	0	7.835317	14 Feb 2000	70936	10134	Stars Trading
18	397.80	0	155.564683	01 Dec 2000	54326778	11435	Group Services
19	287.00	0	266.364683	13 Feb 2000	121053	10721	Witz & Partners
20	1,271.00	0	717.635317	13 Feb 2000	8757170	11475	Triathalon Group
21	537.74	0	15.624683	30 Sep 2000	293732	14438	Bloom County Construction

Summarizing the clusters

If you summarize the **Cluster** field, and sort the summarized output by count, you get the following results, which confirm that the distribution of values is what you expected. Overall, invoice amounts are heavily skewed to lower values. (Centroid values added to the table for ease of comparison.)

The single large value in a cluster by itself appears to be an outlier and should probably be investigated.

Cluster	Count	Centroid value
0	73	553.36
3	16	2,969.04
4	8	8,061.46
2	4	18,010.28

1	1	56,767.20

Cluster on multiple fields

When you cluster on two or more fields, you need to ask yourself how the fields might relate. You could use clustering to test a hypothesis. For example, a company might be concerned about the rate of employee turnover, which management thinks is concentrated among younger, lower-paid employees.

You could use clustering to discover if there is a strong relation between:

- length of employee retention and employee age (two-dimensional clustering)
- length of employee retention, employee age, and salary (three-dimensional clustering)

> **Note**
> For this analysis, you need to avoid including any fields that do not clearly relate to the hypothesis, such as number of sick days taken.

Assessing the output clusters

The clustering algorithm will always output a table with the specified number of clusters. Every record in the output table will be in a cluster.

At this point, you need to assess whether the clusters have analytical significance or meaning. Just because the algorithm groups records in a cluster does not necessarily mean the grouping is significant. You need to ask yourself if the clusters form a significant pattern. Do they tell a story?

> **Tip**
> Graphing the cluster output table as a scatter plot in a reporting tool, with each cluster assigned a different color, is the easiest way to quickly assess the overall nature of the output clusters.

The following characteristics can help you assess the meaningfulness of the output clusters:

- **Cluster coherence** – Are the individual values in a cluster all relatively close to the centroid, or is the cluster somewhat diffuse? The more coherent a cluster, the stronger the relation between the values comprising the cluster.
- **Cluster size** – Are the majority of values contained in one or two large clusters? If so, the data set is significantly skewed, versus a data set in which values are relatively evenly distributed between a number of clusters.
- **Outliers** – Consider the values that resist inclusion in any of the significant clusters. These outliers can represent items that warrant additional scrutiny. Also consider "internal outliers" – that is, values that are included in a significant cluster, but at the outer extremity of the cluster.

> **Note**
> The characteristics above are all human or subjective methods of cluster assessment. Various mathematical methods of cluster evaluation exist but they are beyond the scope of the Analytics Help.

How the clustering algorithm works

Clustering in Analytics uses the K-means clustering algorithm, which is a popular machine learning algorithm. You can find detailed descriptions of K-means clustering on the Internet.

A summary of the algorithm appears below.

Show me more

The K-means clustering algorithm uses an iterative process to optimize clusters:

1	Specify the number of clusters	◦ Decide how many clusters, or groups, to use for grouping a data set. "K" represents the number of clusters you specify. ◦ The data points in the data set can be values in a single numeric field, or composite values that the algorithm computes based on multiple numeric fields.
2	Initialize cluster centroids	◦ Generate a set of random data points to use as the initial centroids, or center points, in the cluster calculation. ◦ The number of centroids generated is equivalent to the number of clusters you specified.
3	Assign each data point to the nearest centroid	◦ Find the shortest distance from each data point to a centroid. Distance comparisons use squared Euclidean distance. ◦ Assign each data point to the nearest centroid. All the data points assigned to a particular centroid become a cluster.
4	Recalculate the centroids	◦ Calculate the average, or mean, of all the data points in a cluster. The mean becomes the new centroid for that cluster.
5	Iterate	◦ Repeat steps 3 and 4: • Recalculate the shortest distance from each data point to a centroid. • Assign each data point to the nearest centroid, which results in some data points being reassigned to different clusters. • Recalculate the centroids. ◦ Continue iterating until no data points are reassigned, or until a specified maximum number of iterations is reached. ◦ With each iteration, the makeup of the clusters becomes more coherent. That is, the data points in a cluster are closer together.

Choosing the number of clusters (K value)

Determining the optimal number of clusters to use when clustering data can require some testing and experimentation. For any given data set, there is not an exact answer.

Show me more

Guidelines for determining the optimal number of clusters:

- **Familiarize with the data** – Familiarize with the data set beforehand, to get a general idea of the profile of the data and any obvious concentrations of values.
- **Go high initially** – Choose a relatively high number of clusters initially – 8 to 10.
- **Try a different number of clusters** – Perform the clustering several times, specifying a different K value each time. A review of the output results can help you judge whether you need more or fewer clusters.
- **Elbow method** – Use the elbow method to programmatically identify the optimal number of clusters. The optimal number is the point at which you achieve the best cluster coherence while avoiding the diminishing returns of additional clusters that only marginally improve coherence at the expense of splitting already coherent clusters.

  You can plot the results of the elbow method in a line chart to visually identify "the elbow", or the inflection point, where increasing the number of clusters does not significantly improve their coherence.

  An elbow method script that you can use in Analytics is available to download from ScriptHub: Elbow method - Sum of Squared Errors (SSE) for K (customer log-in required).

Can I cluster on character or datetime fields?

Generally, you cannot cluster on character or datetime fields. The clustering algorithm accepts only numbers, and it performs calculations with the numbers (Euclidean distance, mean).

Show me more

Categorical character data

You might have categorical character data, such as location IDs, in the form of numbers. Or you could use a computed field to map character categories to a set of numeric codes that you create. You could convert this data to the numeric data type and use it for clustering. However, the resulting clusters would not be valid because you would be performing mathematical calculations on numbers that are representative of something non-numeric.

For example, calculating a centroid position based on the average of a list of location IDs results in a meaningless number. The calculation is based on the invalid assumption that the mathematical distance between location numbers equates to some real-world, measurable distance.

If we consider physical distance, to say that the distance between location 1 and location 9 is twice as far as the distance between location 1 and location 5 makes no sense. Locations 1 and 9 might be beside each other, and location 5 could be miles away.

For a cluster analysis involving location and physical distance, the valid data to use would be geographic coordinates.

Categorical data that represents a scale

You could cluster on categorical data that represents a scale – for example, a rating scale from Poor to Excellent, with corresponding numeric codes from 1 to 5. In this case, an average of the numeric codes has meaning.

Datetime data

You can use Analytics functions to convert datetime data to numeric data. However, the resulting numeric data is not continuous, which presents problems for cluster analysis, which assumes continuous sets of numbers.

For example, the following three numbers, as dates, are all one day apart. However, as numbers, there is a considerable gap, or distance, between the first and second numbers.

- 20181130
- 20181201
- 20181202

You could use serial date values in cluster analysis. Serial dates are a continuous set of integers representing the number of days that have elapsed since 01 January 1900.

Steps

> **Note**
> If the machine learning menu options are disabled, the Python engine is probably not installed. For more information, see "Install ACL for Windows" on page 2583.

Specify settings for the clustering algorithm

1. Open the table with the data that you want to cluster.
2. From the Analytics main menu, select **Machine Learning > Cluster**.
3. In **Number of clusters (K Value)**, specify the number of clusters to use for grouping the data.
4. In **Maximum number of iterations**, specify an upper limit for the number of iterations performed by the clustering algorithm.
5. In **Number of initializations**, specify the number of times to generate an initial set of random centroids.
6. Optional. Select **Seed**, and enter a number.

Specify a data preprocessing method

If you cluster by more than one key field you should use the **Preprocessing** feature to standardize the scale of the fields before using them for clustering.

The scale and units of different numeric fields often vary. For example, a salary field containing dollars per year could range from 20,000 to 100,000, whereas an age field containing years could range from 18 to 70. If you cluster using the salary and age fields, without scaling, the output clusters will be essentially salary clusters, skewed by the size of the salary numbers in comparison to the age numbers, rather than salary/age clusters.

Preprocessing provides the methods explained below to scale all values in all cluster key fields so that they are equally weighted during the clustering process.

Preprocessing option	Description
Standardize	Key field values are centered on a mean of zero (0) and scaled, a process that converts the values to their z-score equivalent (standard score). The z-score is a measure of the number of standard deviations that separate a raw value from the raw mean for each field. In the scaled field, the mean is represented by zero (0), and the z-scores are positive or negative depending on whether the raw values they represent are greater than or less than the raw mean for the field. **Note** Use this option if the key fields contain mostly non-zero values ("dense matrices"). Show example **Example of calculating a z-score** In a scaled age field, the raw age value of 55 is represented by the z-score of 1.038189. • Age field raw mean: 42.04054054 • Age field standard deviation: 12.48276021 • Center the raw value by subtracting the raw mean: 55 - 42.04054054 = 12.95945946 • Scale the centered raw value by dividing by the standard deviation: 12.95945946/12.48276021 = 1.038189 • 55 is 1.038189 standard deviations away from the raw mean.
Scale to unit variance	Key field values are scaled by being divided by their standard deviation, but they are not centered on a mean of zero (0). **Note** Use this option if one or more key fields contain a large number of zero (0) values ("sparse matrices"). Show example **Example of scaling without centering**

Preprocessing option	Description
	In a scaled age field, the raw age value of 55 is represented by the scaled value of 4.406077. • Age field standard deviation: 12.48276021 • Scale the raw value by dividing by the standard deviation: 55/12.48276021 = 4.406077
None	Key field values are not centered or scaled. Clustering uses the raw values, uncentered and unscaled, when calculating the clusters.

Select fields

1. From the **Cluster On** list, select one or more key fields to use for clustering the records in the table.

   Key fields must be numeric.
2. Optional. From the **Other Fields** list, select one or more additional fields to include in the output table.

> **Tip**
> You can **Ctrl+click** to select multiple non-adjacent fields, and **Shift+click** to select multiple adjacent fields.

Finalize command inputs

1. If there are records in the current view that you want to exclude from processing, enter a condition in the **If** text box, or click **If** to create an IF statement using the **Expression Builder**.

   > **Note**
   > The **If** condition is evaluated against only the records remaining in a table after any scope options have been applied (**First**, **Next**, **While**).

   The IF statement considers all records in the view and filters out those that do not meet the specified condition.
2. In the **To** text box, specify the name of the output table.
3. Optional. On the **More** tab:
   a. To specify that only a subset of records are processed, select one of the options in the **Scope** panel.
   b. Select **Use Output Table** if you want the output table to open automatically.
4. Click **OK**.

Performing Benford analysis

Benford analysis counts the number of times each leading digit (1-9) or leading digit combination occurs in a field, and compares the actual count to the expected count.

The expected count, calculated using the Benford formula, provides the Benford distribution. In a naturally occurring set of numbers, the frequency distribution of the actual count of leading digits should approximate the Benford distribution.

If one or more leading digits or digit combinations in the data being tested deviate significantly from the Benford distribution, it may indicate that the numbers have been manipulated. Deviations may also have simple and reasonable explanations and are not necessarily indicative of manipulation.

What data can I test using Benford analysis?

You should only use Benford analysis for testing numeric data composed of "naturally occurring numbers", such as accounting amounts, transaction amounts, expenses, or address numbers. Benford analysis is not suitable for numeric data that is constrained in any way.

Follow these guidelines for identifying numeric data that is suitable for Benford analysis:

- **Size of the data set** - The data set must be large enough to support a valid distribution. Benford analysis may not give reliable results for fewer than 500 records.
- **Leading digit requirement** - All numbers from 1 to 9 must have the possibility of occurring as the leading digit.
- **Leading digit combination requirement** - All numbers from 0 to 9 must have the possibility of occurring as the second leading digit, and as any additional digits being analyzed.
- **Constrained data** - Numeric data that is assigned or generated according to a pre-ordained pattern is not suitable for Benford analysis. For example, do not use Benford to analyze:
  - sequential check or invoice numbers
  - social security numbers or telephone numbers that map to a specific pattern
  - any numbering scheme with a range that prevents certain numbers from appearing
- **Random numbers** - Numbers generated by a random number generator are not suitable for Benford analysis.

Usage details

The table below provides details about using the Benford analysis feature in Analytics.

Number of leading digits	You can analyze up to six leading digits. When analyzing four or more leading

	digits, Benford analysis output must be sent to a file instead of displayed on screen or sent to a printer.
Processing time	Depending on the number of records you are working with, analyzing five or more leading digits may take several minutes. Regardless of how many digits you are analyzing, you can press **Esc** to terminate the command at any time.
Size of data set	Effective Benford analysis requires large data sets. Analytics displays a warning in the results output when a data set may be too small for the specified number of digits.
Positive and negative values	Anomalous data is more apparent when you analyze positive and negative values separately. You can use a filter to separate the two before beginning your analysis.
Zeros and non-numeric characters	Records with values of zero are ignored, but the number of zero-value records bypassed is reported. Leading zeros, numeric formatting such as decimals and dollar signs, other non-numeric digits, and records that fail to meet test criteria are also ignored. If the resulting number of digits is less than specified, Analytics adds zeros to the right of the result.

Benford analysis output results

Benford analysis produces the following output results:

Leading Digits	Displays the leading digits that were tested. For example, if you specify one leading digit, the numbers 1 to 9 are displayed. If you specify two leading digits, the numbers 10 to 99 are displayed.
Actual Count	Displays the actual count of each leading digit or leading digit combination in the field.
Expected Count	Displays the expected count of each leading digit or leading digit combination calculated by the Benford formula.
Zstat Ratio	Displays the Z-Stat ratio for each digit or digit combination, which is a measurement in standard deviations of the distance between the actual count and the expected count. For example, a Z-statistic of 0.500 represents one-half of a standard deviation.
Lower Bound Upper Bound (optional)	Displays the computed lower and upper bound values for the count of each leading digit or digit combination. If the actual count of more than one digit or digit combination in the output results exceeds either of the bounds, the data may have been manipulated and should be investigated.

> **Note**
> The **Lower Bound** and **Upper Bound** values are included only if the **Include Upper and Lower Bounds** checkbox is selected in the **Benford** dialog box.

Steps

Perform Benford analysis on a field to discover if one or more leading digits or digit combinations deviate significantly from the Benford distribution.

Show me how

1. Open the table containing the field you want to analyze.
2. Select **Analyze > Benford**.
3. On the **Main** tab, do one of the following:
   - Select the field to analyze from the **Benford On** drop-down list.
   - Click **Benford On** to select the field, or to create an expression.

   > **Note**
   > Select a field that contains "naturally occurring numbers", such as transaction amounts. Benford analysis is not suitable for numeric data that is constrained in any way. For more information, see "What data can I test using Benford analysis?" on page 1316

4. Enter the **Number of Leading Digits**, from 1 to 6, that you want to analyze.

   > **Note**
   > If you are analyzing four or more leading digits, results output must be sent to a file. Results of analyzing four or more digits cannot be displayed on the screen, sent to the printer, or displayed in a graph.

5. If there are records in the current view that you want to exclude from processing, enter a condition in the **If** text box, or click **If** to create an IF statement using the **Expression Builder**.

   > **Note**
   > The **If** condition is evaluated against only the records remaining in a table after any scope options have been applied (**First**, **Next**, **While**).

   The IF statement considers all records in the view and filters out those that do not meet the specified condition.
6. (Optional) Select **Include Upper and Lower Bounds** if you want to include computed boundary values in the output results for each digit or digit combination.
7. Click the **Output** tab.

8. Select the appropriate output option in the **To** panel:
   - **Screen** – Select this option to display the results in the Analytics display area.

     > **Tip**
     > You can click any linked result value in the display area to drill down to the associated record or records in the source table.

     If the output table contains a large number of records, it is faster and more useful to save the results to a file than to display the results on the screen.
   - **Print** – Select this option to send the results to the default printer.
   - **Graph** – Select this option to create a graph of the results and display it in the Analytics display area.
   - **File** – Select this option to save or append the results to a text file. The file is saved outside Analytics.

     > **Note**
     > Output options that do not apply to a particular analytical operation are disabled.

9. If you selected **File** as the output type, specify the following information in the **As** panel:
   - **File Type** – **ASCII Text File** or **Unicode Text file** (depending on which edition of Analytics you are using) is the only option. Saves the results to a new text file, or appends the results to an existing text file.
   - **Name** – Enter a file name in the **Name** text box. Or click **Name** and enter the file name, or select an existing file in the **Save** or **Save File As** dialog box to overwrite or append to the file. If Analytics prefills a file name, you can accept the prefilled name, or change it.

     You can also specify an absolute or relative file path, or navigate to a different folder, to save or append the file in a location other than the project location. For example: `C:\Results\Output.txt` or `Results\Output.txt`.
   - **Local** – Disabled and selected. Saving the file locally is the only option.
10. Depending on the output type, you can optionally specify a **Header** and/or a **Footer** in the text box(es).

    Headers and footers are centered by default. Type a left angle bracket (<) before the header or footer text to left align the text. Click **Header** or **Footer** to enter a header or footer of more than one line. Alternatively, you can enter a semi-colon (;) as a line-break character in the header or footer text box. Left aligning multiple lines requires a left angle bracket at the beginning of each line.
11. Click the **More** tab.
12. Select the appropriate option in the **Scope** panel:
    - **All**
    - **First**
    - **Next**
    - **While**

    Show me more

All	This option is selected by default. Leave it selected to specify that all records in the view are processed.
First	Select this option and enter a number in the text box to start processing at the first record in the view and include only the specified number of records.
Next	Select this option and enter a number in the text box to start processing at the currently selected record in the view and include only the specified number of records. The actual record number in the leftmost column must be selected, not data in the row.
While	Select this option to use a WHILE statement to limit the processing of records in the view based on a particular criterion or set of criteria. You can enter a condition in the **While** text box, or click **While** to create a WHILE statement using the **Expression Builder**. A WHILE statement allows records in the view to be processed only while the specified condition evaluates to true. As soon as the condition evaluates to false, the processing terminates, and no further records are considered. You can use the **While** option in conjunction with the **All**, **First**, or **Next** options. Record processing stops as soon as one limit is reached.

> **Note**
> The number of records specified in the **First** or **Next** options references either the physical or the indexed order of records in a table, and disregards any filtering or quick sorting applied to the view. However, results of analytical operations respect any filtering.
> If a view is quick sorted, **Next** behaves like **First**.

13. If you selected **File** as the output type, and want to append the output results to the end of an existing text file, select **Append To Existing File**.
14. If you selected **File** as the output type, and want to append the output results to the end of an existing Analytics table, do one of the following:
    - Select **Append To Existing File** if you are certain the output results and the existing table are identical in structure.
    - Leave **Append To Existing File** deselected if you want Analytics to compare the record lengths of the output results and the existing table. If the record lengths are not identical, the data structure is not identical, and the append will not work correctly.

    > **Note**
    > Leaving **Append To Existing File** deselected is recommended if you are uncertain whether the output results and the existing table have an identical data structure. For more information about appending and data structure, see "Appending output results to an existing table" on page 200.

15. Click **OK**.

16. If the overwrite prompt appears, select the appropriate option.

    If you are expecting the **Append** option to appear and it does not, click **No** to cancel the operation and see "Appending output results to an existing table" on page 200.

Running R scripts

Analyze an Analytics table in an external R script and then return data from R to create a new table in the Analytics project. Source data is passed to R as a **data frame** that you can reference using a provided function.

Working with Analytics data in R

If you are preparing the R script to run from Analytics, familiarize yourself with the ways data is passed back and forth between Analytics and R. You must use the R functions provided by Analytics in your R script to successfully run the RCOMMAND.

Show me more

Referencing Analytics data in the R script

The Analytics table is passed to the script as an R **data frame**. Data frames are tabular data objects that may contain columns of different modes, or types, of data.

To work with the data frame created by Analytics in an R script, invoke the `acl.readData()` function and store the returned data frame in a variable:

```
# stores the Analytics table in a data frame called myTable that can be ref-
erenced throughout the script
myTable<-acl.readData()
```

To retrieve data from a cell in the data frame, you can use one of the following approaches:

- Using row and column coordinates:

  ```
  # Retrieves the value in the first row and second column of the data
  frame
  myTable[1,2]
  ```

  > **Note**
  > Coordinates are based on the order of fields specified in the command, not the table layout or view that is currently open.

- Using row and column names:

```
# Retrieves the value in the first row and "myColumnTitle" column of the
data frame
myTable["1","myColumnTitle"]
```

You must specify the **KEEPTITLE** option of the command to use column names.

Rows are named "1", "2", "3", and increment accordingly. You may also use a combination of names and coordinates.

Passing data back to Analytics

To return a data frame or matrix back to Analytics and create a new table, use the following syntax:

```
# Passes myNewTable data frame back to Analytics to create a new table
acl.output<-myNewTable
```

> **Note**
> You must return a data frame or a matrix to Analytics when the R script terminates. Ensure the columns in the data frame or matrix contain only atomic values and not lists, matrices, arrays, or non-atomic objects. If the values cannot be translated into Analytics data types, the command fails.

R data mapping

Analytics data types are translated into R data types using a translation process between the Analytics project and the R script:

Analytics data type	R data type(s)
Logical	Logical
Numeric	Numeric
Character	Character
Datetime	Date, POSIXct, POSIXlt

Performance and file size limits

The time it takes to run your R script and process the data that is returned increases for input data exceeding 1 GB. R does not support input files of 2 GB or higher.

The number of records sent to R also affects performance. For two tables with the same file size but a differing record count, processing the table with fewer records is faster.

Handling multi-byte character data

If you are sending data to R in a multi-byte character set, such as Chinese, you must set the system locale appropriately in your R script. To successfully send a table of multi-byte data to R, the first line of the R script must contain the following function:

```
# Example that sets locale to Chinese
Sys.setlocale("LC_ALL","Chinese")
```

For more information about Sys.setlocale( ), see the R documentation.

Hello world example

Analytics command

```
RCOMMAND FIELDS "Hello", ", world!" TO "r_result" RSCRIPT "C:\scripts\r_
scripts\analysis.r"
```

R script (analysis.r)

```
srcTable<-acl.readData()

# create table to send back to ACL
output<-data.frame(
  c(srcTable[1,1]),
  c(srcTable[1,2])
)

# add column names and send table back to ACL
colnames(output) <- c("Greeting","Subject")
acl.output<-output
```

Run an R script

1. From the menu, select **Analyze > R**.

   The **RCOMMAND** dialog box opens.
2. Next to the **R Script** field, click **Browse** and navigate to the R script on your computer that you want to run.

3. Click **Select Fields** and add one or more fields to include in the data frame that Analytics makes available in the R script.

> **Tip**
> You can also include expressions as fields in the data frame. To create an expression, click **Expr** and use the functions, fields, and operators available to you in the dialog box. For more information, see "Expression Builder overview" on page 799.

4. Optional. In the **RCommand Options** section, define how you want to send the Analytics data to the R script.

   For more information, see "RCommand options" below.

5. Optional. To filter the records that are sent to the R script, click **If** and use the **Expression Builder** dialog box to create a conditional expression to use as the filter.

   For more information about creating expressions using the Expression Builder, see "Creating expressions using the Expression Builder" on page 801.

6. To specify the output table, click **To** and in the **File name** field, enter a name for the table and associated `.FIL` file.

   Use the folder explorer to navigate to the folder you want to use to store the source data file.

7. Optional. On the **More** tab of the dialog box, specify any scope options for the command.

   For more information, see "More tab" on the next page.

8. To run the command, click **OK**.

RCOMMAND dialog box options

RCommand options

Option	Description
Export with field names	Use the column titles of the source Analytics table as header values for the R data frame. This option sets **KEEPTITLE** option on the command and is required if you want to retrieve data using column names in the R script.
Column Separator	The character to use as the separator between fields when sending data to R.
Text Qualifier	The character to use as the text qualifier to identify field values when sending data to R.

More tab

Option	Description
All	Processes all records in the view (default selection).
First	Processes from the first record in the table and includes only the specified number of records.
Next	Processes from the currently selected record in the table and includes only the specified number of records. **Note** The number of records specified in the **First** or **Next** options references either the physical or the indexed order of records in a table, and disregards any filtering or quick sorting applied to the view. However, results of analytical operations respect any filtering. If a view is quick sorted, **Next** behaves like **First**. **Caution** There is a known issue in the current version with Next when running the RCOMMAND. Avoid using this option as the record reference may reset to the first record regardless of which record is selected.
While	Uses a WHILE statement to limit the processing of records in the primary table based on criteria. Records in the view are processed only while the specified condition evaluates to true. As soon as the condition evaluates to false, the processing terminates, and no further records are considered. For more information, see "Creating expressions using the Expression Builder" on page 801.

Reporting your findings

When you have completed your analysis in Analytics you often need to report your findings. You can use the native reporting features in Analytics, or you can import Analytics data into a third-party reporting application such as Tableau.

The type of report you create depends on how you want to present the results of your analysis and the complexity of the report layout.

Using Analytics to report your findings

Three types of reports are available in Analytics for presenting your findings and analysis:

- Analytics reports
- Analytics graphs
- Analysis app interpretations and visualizations

Analytics reports

You can create text reports based on views you define in Analytics. A number of configurable options let you determine the content and layout of the report.

For more information, see "Formatting and generating Analytics reports" on page 1330.

Analytics graphs

You can generate graphs as the output of some Analytics operations. You can also graph data you select in views. The graph toolbar provides a number of options for formatting the graphs.

For more information, see "Working with Analytics graphs" on page 1336.

Analysis app interpretations and visualizations

You can create customized tables, charts, and metrics in the Analysis App window. You can use interpretations and visualizations with the results output by an analytic, or with any table in an Analytics project.

For more information, see "Working with analysis apps" on page 2642.

Using a third-party application to report your findings

You can use any ODBC-compliant reporting application to connect to Analytics data and report your findings. The applications you can use include:

- Tableau
- Microsoft Power BI Desktop
- Excel
- Crystal Reports
- Qlik
- MicroStrategy

For more information, see "Connecting to Analytics from a third-party reporting application" on page 1348.

Formatting and generating Analytics reports

You can create traditional tabular reports in Analytics based on the format of data and columns in views. You configure various settings in a view to control how data is displayed in the report. These configuration settings are saved with the view.

> **Tip**
> You can create a separate view for configuring and saving report settings, which differs from the view you use for displaying data.

When you generate a report for the first time, you can specify a number of additional properties, such as line spacing, header and footer text, and the report output type. These properties are saved with the Analytics project when the project is saved.

Configure a view to format a report

When you configure a view to format a report you can use the same view that you use to view data on screen, or you can create a new view specifically for the report. If required, you can create multiple views configured differently for different reports based on the same data set.

You can configure a view in a number of ways to create more readable and meaningful reports:

Filter data	Create a filter to remove irrelevant records from the view. Excluded records are not included in the report. For example, you could filter a table of sales data to include only the stores that interest you.
Index records	Create an index to sort the records in the view by one or more columns.
Select specific columns	Add or remove columns so that you display only relevant data. You can include any physical data fields or computed fields in the table layout.
Arrange columns	Reorder columns in the view to present the sequence of information you want.
Create sub-sections	Specify break columns to divide the report into sections with subtotals. For example, specifying Customer_Name as a break column in an invoice table groups and subtotals the invoices by customer. You also have the option of inserting a page break after each group in the report.
Tailor the data	Suppress duplicate identifier values such as repeated names, suppress numeric totals that are not meaningful, and display zeros as blanks.

Control report width	Adjust the rows in a view to span multiple rows.

> **Note**
> For detailed information about configuring views, see "Customizing columns in views" on page 778.

Specify the report font

You can specify the font for an individual report output to print, or to an HTML file.

Show me how

1. Open the view configured for the report.
2. Above the display area, select **Change Font** Ff.
3. In the **Select View Fonts** dialog box, click the button or buttons for the portions of the report that you want to format.
4. In the **Font** dialog box, specify the font information and click **OK**.
5. When you are finished specifying fonts, in the **Select View Fonts** dialog box, click **OK**.

   The specified fonts are used in the report generated from the view.

Generate a report

Once you have configured a view to use as the basis for a report, you are ready to generate the report.

Show me how

Steps

1. Open the view configured for the report.
2. Select **Data > Report**.
3. Specify options in the **Report** dialog box, using the tables below as a guide, then click **OK**.

   The report is generated.
4. (Optional) If you want to permanently save the report options you specified, select **File > Save Project**.

   The report options are saved with the view, and they are preselected the next time you generate a report from the same view.

> **Tip**
> To quickly generate subsequent reports from the same view using the same report options, select **File > Print**.

Report dialog box options

The tables below provide detailed information about the options in the **Report** dialog box.

Main tab

Options - Report dialog box	Description
Header Footer optional	Creates a report header and/or footer. Headers and footers can be more than one line, and are centered by default. Type a left angle bracket (<) before the header or footer text to left align the text. Left aligning multiple lines requires a left angle bracket at the beginning of each line. You also have the option of specifying a standard header and footer to apply to all reports produced during an Analytics session. For more information, see "Specify a standard header and footer" on page 1335.
If optional	Creates a condition that specifies which records appear in the report. You can enter a condition in the **If** text box, or click **If** to create an IF statement using the **Expression Builder**. The IF condition considers all records in the view and filters out those that do not meet the specified condition.
Presort optional	Sorts the report by: ◦ **break column or columns** - any column in the view with **Break Column** selected in the column properties ◦ **sort column or columns** - any column in the view with **Sort Key Column** selected in the column properties Nested sorting uses the order of the break columns or sort columns in the view, beginning with the leftmost column. **Note** The **Presort** option is available only when at least one column in the view has **Break Column** or **Sort Key Column** selected. For more information, see "Modify column properties" on page 783.
Summarize optional	Generates a report that includes only subtotals and totals of break fields, and excludes detail lines.
Suppress blank detail lines optional	Automatically removes blank detail lines from the report.
Single Spaced Double Spaced Triple Spaced	Specifies the spacing of the report.

Options - Report dialog box	Description
Setup	Specifies various print options.
Preview optional	Displays a preview of the report.
Fit to page optional	Scales the report to include all columns from the view. If **Fit to page** is not selected, only columns that appear to the left of the page width indicator in the view (vertical dotted line) are displayed in printed reports. **Note** Reports saved as files or displayed on the screen automatically include all columns from the view.

Output tab

Options - Report dialog box	Description
To panel	◦ **Screen** - displays the report in the Analytics display area. If the report contains a large number of records, it is faster and more useful to save the report to a file than to display the report on the screen. ◦ **Print** - sends the report to the default printer. ◦ **File** - saves or appends the report to a text file or an HTML file. The file is saved outside Analytics.
As panel	File output only. ◦ **File Type** - specifies output to a text file or an HTML file. ◦ **Name** - specifies a name for the output file. Do one of the following: • enter a file name in the **Name** text box • click **Name** and enter the file name • click **Name** and select an existing file in the **Save** or **Save File As** dialog box to overwrite or append to the file You can also specify an absolute or relative file path, or navigate to a different folder, to save or append the file in a location other than the project location. For example: `C:\Results\Report.txt` or `Results\Report.htm`. ◦ **Local** - disabled and selected. Saving the file locally is the only option.
Header Footer optional	Replicates any header or footer text from the **Main** tab. If required, you can add header or footer text in the **Output** tab, or update existing header or footer text. Changes you make to existing text are automatically updated on the **Main** tab. Type a semi-colon (;) in the **Header** or **Footer** text boxes to create a line break.
OK	Generates the report.

Options - Report dialog box	Description
	If you are saving the report to a file, and the overwrite prompt appears, select the appropriate option.

Global report options

You can specify several report options globally. Most of the global options apply only to reports output to print.

Append additional information

You can append additional information, such as Analytics table name and field definition information, to the end of a printed report. You specify options for additional information in the **Options** dialog box.

Show me how

1. Select **Tools > Options > Print**.
2. Specify any of the options shown below and click **OK**.

   The specified information appears at the end of every print report you generate.

Option	Description
Include Report History with Reports	Adds the following information to the end of print reports: ◦ the Analytics project, table, and data file names ◦ the REPORT command used to generate the report ◦ any table history information ◦ any table layout notes
Include Field Definitions in Table History	Adds the following information to the end of print reports: ◦ the field definitions for each physical data field and computed field in the table layout ◦ any field notes This option requires that the **Include Report History with Reports** option is also selected.
Include View Note in Report History	Adds the following information to the end of print reports: ◦ the view name ◦ any view notes This option requires that the **Include Report History with Reports** option is also selected.

> **Note**
> If you want to include record notes in a report, you must first add the **RecordNote** column to the view. Record notes are not affected by any of the global report options.

Set margins

You can specify the margins used for printed reports.

Show me how

1. Select **Tools > Options > Print**.
2. Use the **Margin** text boxes to specify margins for printed reports.

   The specified margins are used for every print report you generate.

Specify a standard header and footer

You can specify a standard header and footer to apply to all reports produced during an Analytics session. The header and footer text that you specify is stored in the HEADER and FOOTER system variables.

Show me how

1. In the Analytics **Command Line**, type `HEADER = "header text"`, or `FOOTER = "footer text"`.

   To create a multi-line header or footer, use a semi-colon between the lines. For example:

   `HEADER = "header line 1;header line 2"`

2. Press Enter.

   The values you specify are automatically used as headers or footers in all Analytics reports, unless you override the values with a header or a footer specified while performing an Analytics operation.

   The HEADER and FOOTER values remain in effect until the variables are updated or deleted, or until the end of the current Analytics session.

Working with Analytics graphs

> **Tip**
> The charts in the Analysis App window are more modern than the legacy graphs in Analytics and they offer a visually attractive alternative. If the charts in the Analysis App window meet your reporting needs, you should use them rather than the legacy graphs.
> For more information, see "Working with analysis apps" on page 2642.

You can select from a variety of graph types to display Analytics data. There are two ways to generate or create graphs from table data:

- **Generate a graph from command results** – The following commands in the **Analyze** menu, and the equivalent ACLScript commands, provide the option to output the results as a graph:
  - Age
  - Classify
  - Cross-tabulate
  - Histogram
  - Benford Analysis
  - Stratify

  Scripts that contain appropriate commands can display the results of those commands in a graph. This allows you to automate graph-based reports for analyzing data that changes over time.

  When you generate a graph from command results, you can us the Drill-down option to analyze a subset of the graphed data. When you drill down in a graph, Analytics creates a filter and displays the selected subset of records in an Analytics view.
- **Generate a graph from selected data in a view** – When working with an Analytics view, you can select data in the view and create a graph. The Drill-down option is not available in graphs generated from Analytics views because the subset of data is already selected in the view.

Changing graph formatting

The Graph toolbar includes a variety of commands for modifying particular aspects of the graph layout. You can also modify the graph directly in the **Graph** tab or **Graph** window by performing the following common Windows operations:

- Clicking a graph, or a graph legend, to display its handles and then dragging a handle to change the item's size
- Clicking an item to select it, then dragging the item to move it
- Double-clicking graphs and graph legends to display **Properties** dialog boxes.

The graph display area

When you generate a graph using the menu options in the **Analyze** menu, the graph appears in a new tab of the Analytics display area. The graph header contains the name of the ACLScript command executed to generate the graph, while the footer contains the complete syntax of the command.

When you generate a graph directly from a view, the graph appears in a new window. The graph header contains the phrase "Graph Column(s) from View."

The display area for graphs generated from both commands and views includes the same functionality, with the exception that the Drill-down option is disabled in the toolbar for graphs generated from views.

The graph toolbar

The graph toolbar, located above the graph header, contains buttons that allow you to do the following:

- Configure settings for various aspects of the graph display
- Save, copy, or print the graph
- Drill-down to analyze specific subsets of the graph data in the associated view

Depending on the method you used to generate the graph and the type of graph being displayed, some toolbar functionality might not be available. For example, the Drill-down command is not available when you generate a graph by selecting data in a view.

Icon	Name	Description
	Graph Type	Used to select the type of graph that you want to display
	Graph Properties	Used to edit graph properties such as font, background, frame, and whether you want the graph display to include axes, legends, or grid lines

Icon	Name	Description
	Legend Properties	Used to edit legend properties such as font, border, and color schemes for each field displayed in the graph
	Axis Properties	Used to edit axis properties such as font, style and scale
	Format Data	Used to edit data format properties such as font, which fields to include in the graph, orientation, labels, and color schemes for data series
	Label Properties	Used to edit label properties such as font, border, and orientation
	Show/Hide Legend	Used to toggle the display of the graph legend
	Show/Hide Axis	Used to toggle the display of the graph axis
	Print Graph	Used to print the graph to any installed Windows print device
	Save Graph as Bitmap	Used to save the graph as a bitmap image file
	Copy Graph to Clipboard	Used to copy the graph to the clipboard for pasting into other applications
	Edit Command	Used to edit the command that you executed to generate the graph
	Rotate Chart Left	Used to rotate a pie chart to the left
	Rotate Chart Right	Used to rotate a pie chart to the right
	Drill-down	Used to open selected graph segments in a table view for analysis

Change graph type

The default graph type is a 3D bar graph. Depending on the type of data you are working with, you can display graphs in the formats listed below. Only options that apply to the data in your graph are available. Options that do not apply are disabled.

> **Note**
> If you want to change the default type of graph that Analytics generates, enter an appropriate SET GRAPH command in the command line.

Icon	Graph type	Icon	Graph type	Icon	Graph type
	2D bar graph		3D stacked graph		3D layered graph
	3D bar graph		2D pie chart		Line graph
	2D stacked graph		3D pie chart		Benford graph

Show me how

1. Click **Graph Type**  .

> **Note**
> If the results you are graphing contain a single item, Analytics disables the stacked and 3-D layered options. If the results contain multiple items, stacked and layered options are available but pie chart options are unavailable.

2. Click one of the available graph formats.
3. Click **OK**.

> **Tip**
> To separate the sections of a pie graph, right-click on a section and select **Explode Pie**.

Change graph properties

The **Graph Properties** dialog box allows you to change the background color, axis properties, and borders of your graph. You can also change the font used for all text elements on the graph.

Show me how

1. Click **Graph Properties**  .
2. Click the appropriate tab for the graph property you want to edit.
   - **Global Font** – Change graph text font, style, size, color, or background settings for all text elements on the graph.
   - **Background** – Change details of the graph display area's border and background.

- **Frame** – Change border and background settings for the frame containing graphed results data.
- **Options** – Select options for displaying the axis, legend, and grid lines.

3. Click **OK**.

Change graph legend properties

The **Legend Properties** dialog box allows you change the legend font and color, or customize the legend box.

Show me how

1. Click **Legend Properties**  .
2. Click the appropriate tab for the legend property you want to edit:
   - **Legend Attributes** – Change border attributes including style, color, thickness, and drop shadow.
   - **Font** – Change legend text font, style, size, color, or background settings.
   - **Data Series** – Change color schemes for the graphical representation of fields and toggle the option to display them as either transparent or solid.
3. Click **OK**.

> **Note**
> You can right-click the legend to choose fields to display in the legend or to hide the legend. You can also click **Show/Hide Legend** in the toolbar.

Change graph axis properties

The **Axis Properties** dialog box allows you to change the style, scale, and font of a graph axis.

Show me how

1. Click **Axis Properties**  .
2. Click the appropriate tab for the axis property you want to edit:
   - **Axis Style** – Change properties such as style, color, thickness, and the optional inclusion of tick marks to delineate regular intervals on the axis.
   - **Axis Scale** – Specify minimum and maximum values to display on the axis and major units by which the axis is divided. Select **Auto** to have these values automatically assigned. You can also specify the orientation of any text values you choose to display.
   - **Font** – Change axis text font, style, size, color, or background settings.
3. Click **OK**.

> **Tip**
> You can right-click the axis to switch between a vertical and horizontal representation of the axis. You can also click **Show/Hide Axis** in the toolbar.

Change graph data display properties

The **Format Data** dialog box allows you to select which fields are displayed in a graph, and change the colors assigned to each field. For example, you can display fewer fields, or display a field that is hidden by default.

You can also change the font size, style, color, and orientation of the data labels, or hide them from view altogether.

Show me how

1. Click **Format Data**  .
2. Click the appropriate tab for the data display property you want to edit:
   - **Data** – Move available fields that you want displayed in a graph to the **Selected fields** list. Use the **Up** and **Down** buttons to change the order in which the fields are displayed in the graph and in the legend.

     > **Tip**
     > You can also choose which available fields to display by right-clicking the graph display area and selecting **Select Fields**.

   - **Options** – Change label text orientation options.
   - **Font** – Change data display text font, style, size, or color of text on the *x* axis.
   - **Data Series** – Change color schemes for the graphical representation of fields and toggle the option to display them as either transparent or solid.
3. Click **OK**.

Add graph labels

You can add and move graph labels as necessary.

Show me how

1. Right-click the area of the graph where you want to place the label and select **Add Label**.
2. Enter the label text in the text box.
3. Click **OK**.
4. If you want to move the label, do the following:
   a. Click the label so that the border of the label box is visible.
   b. Position the pointer on the border of the label box so that it changes to a four-headed arrow.
   c. Drag the label box to the position on the graph where you want to place the label.

Change graph label properties

The **Label Properties** dialog box allows you to change label properties. Labels describe elements of graphs generated from a command or a view. You can add new labels to a graph by right-clicking in

the graph display area and selecting **Add Label**.

Show me how

1. Click the label you want to change and click **Label Properties**  .
2. Click the appropriate tab for the label property you want to edit:
   - **Orientation** – Rotate the label orientation by degrees, or change the vertical or horizontal alignment of label text.
   - **Font** – Change label text font, style, size, color, or background settings of the selected label.
   - **Attributes** – Change border attributes for the label, including style, color, thickness, and drop shadow.
3. Click **OK**.

> **Tip**
> Right-click a label to cut, copy, paste, or delete the label.

Rotate pie charts

You can rotate pie charts left or right. The rotate options are only available for pie charts.

Show me how

1. Depending on the direction you want to rotate the pie chart, do one of the following:
   - To rotate left, click **Rotate Chart Left**  .
   - To rotate right, click **Rotate Chart Right**  .
2. Repeat as necessary to continue rotating the pie chart in 22.5 degree increments.

Drilling down into graphed data

You can use the Drill-down option to view the data represented by selected portions of a graph. Analytics uses the graph to define a filter that shows only the selected graph data, and displays the filtered data in the open view.

> **Note**
> Drill-down functionality is not available for graphs created by selecting data in views. This restriction exists because the data used to create the graph is already selected in the view.

To use the graph drill-down feature:

1. Do one of the following:
   - Click a segment of a bar graph or pie chart to select it.
   - **Shift+click** to select multiple adjacent segments of a bar graph or pie chart.
   - **Ctrl+click** to select multiple non-adjacent segments of a bar graph or pie chart.
2. Click **Drill-down**  .

   > **Tip**
   > If you want to analyze table data for a single graph segment, you can double-click the appropriate segment.

   Analytics displays a filtered view of the data selected from the graph. You can continue to analyze the selected data using other Analytics commands and functions. To return to the original table view, click **Remove Filter**.

Editing graph commands

The **Edit Command** button allows you to edit the command that generated the graph.

> **Note**
> The **Edit Command** button is enabled only for graphs generated from an Analytics command.

To edit the command a graph was generated from:

1. Click **Edit Command**  .
2. Click the appropriate tab for the command parameters you want to edit. For example, if you generated the graph from the Classify command, you can modify the settings in the **Main** tab and the **More** tab in the **Classify** dialog box to generate a graph using the new parameters.
3. Click **OK**.

Copying graphs to the clipboard

You can copy a graph to the clipboard for pasting into other documents. To avoid losing the clipboard copy of your graph, open an application that can read bitmap files and paste the graph in a file.

To copy a graph to the clipboard:

- Click **Copy Graph to Clipboard**  .

Saving graphs as images

You can save a graph as a bitmap file to insert in other documents, or to keep as a snapshot of your graphed data.

To save a graph:

1. Click **Save Graph as Bitmap**  .
2. In the **Save 'Graph' As** dialog box, navigate to an appropriate folder and enter a name for the graph you are saving.
3. Click **Save**.

Printing graphs

The **Print Graph** command allows you to print a graph on any installed printer device. For maximum print resolution, maximize the graph window before printing.

To print a graph:

1. Click **Print Graph**  .
2. In the **Print** dialog box, select an appropriate printer device, change print properties if necessary, and choose the number of copies to print.
3. Click **OK**.

Connecting to Analytics from a third-party reporting application

You can use any ODBC-compliant reporting application to connect to Analytics data and report your findings. The applications you can use include:

- Tableau
- Microsoft Power BI Desktop
- Excel
- Crystal Reports
- Qlik
- MicroStrategy

How does it work?

In your chosen reporting application you use the ODBC feature to create a connection to any Analytics project, using the ACL Connector for Analytics.

Once connected, you can select tables and fields from the Analytics project, and use any additional ODBC features that are available in the third-party application, such as joining and filtering.

> **Note**
>
> The data connection capabilities of third-party reporting applications differ. For example, some applications require that to connect to multiple tables you must join them, while other applications support connecting to multiple tables individually.
>
> The ACL Connector for Analytics supports connecting to local Analytics tables only. It does not support connecting to server tables in an Analytics project.

Optimal performance

You get the best performance from the ACL Connector for Analytics by limiting the size of the Analytics data sets you connect to. The connector is designed to support a wide range of reporting tools, but it is intended to work with the smaller data sets typical of results rather than with entire source data tables. Joining tables in the process of connecting is particularly resource-intensive.

Analytics data is stored in flat files, which are slower to access with ODBC than databases.

Create a data connection (DSN)

Depending on which reporting application you use, you may have to first create a data connection in Windows before you can connect from the reporting application to Analytics.

For example:

- Microsoft Power BI Desktop requires that you first create a data connection if you want to avoid manually entering a connection string.
- Tableau and Excel do not require that you create a data connection because they automatically create the connection for you.

The data connection is a DSN, which stands for Data Source Name.

> **Tip**
> You may choose to manually create data connections for Tableau or Excel as a way of saving multiple connections to different Analytics projects.

1. From the **Administrative Tools** folder of your Windows operating system, open the version of the **ODBC Data Source Administrator** that matches the bitness of the third-party application you want to connect from (32-bit or 64-bit).

   For example, if you want to connect from a 32-bit version of Excel, open the 32-bit **ODBC Data Source Administrator**.

   > **Caution**
   > If you create the data connection in the wrong version of the **ODBC Data Source Administrator**, the connection is not visible or accessible in the third-party application. Or it may be accessible, but causes a connection error.

2. In the **ODBC Data Source Administrator**, do one of the following:
   - Select the **System DSN** tab if you want the data connection to be available to anyone who uses the computer.
   - Select the **User DSN** tab if only you will use the data connection.

   > **Note**
   > A default Analytics data connection named **ACL ODBC** or **ACL ODBC 64** already exists in the **System DSN** tab. Do not modify this default data connection. For more information, see "Default Analytics data connection" on the next page.

3. Click **Add**, select **ACL Connector for Analytics**, and click **Finish**.
4. In the **ACL Data Store Interface DSN Setup** dialog box, enter the following information:
   - **Data Source Name** – enter a meaningful name such as "Analytics General Ledger project".
   - **Description** – enter a meaningful description of the Analytics project, such as "General Ledger audit 2017".

- **ACL Project File** – click **Browse** and select an Analytics project in the **Open Project File** dialog box.

5. Click **OK**.

   The new data connection to the specified Analytics project is created and is now available to select in a third-party reporting application.

   If required, you can create additional data connections to other Analytics projects.

6. Click **OK** to exit the **ODBC Data Source Administrator**.

Default Analytics data connection

When you install Analytics, a 32-bit and a 64-bit data connection (DSN) with the following names are created on your computer:

- **ACL ODBC** (32-bit)
- **ACL ODBC 64** (64-bit)

These are Analytics data connections with an unspecified Analytics project. You can use them to connect to different Analytics projects on the fly – that is, you can select the Analytics project to connect to at the time that you make the connection. Some reporting applications may not support this use, and may require a data connection with a particular Analytics project specified in advance.

> **Note**
>
> Do not add connection information to either of the default data connections if you want to retain the ability to connect to different Analytics projects on the fly.

Connecting to an Analytics project

The instructions below provide three examples of connecting to an Analytics project using a third-party application:

- "Connect from Tableau Desktop 10.1" below
- "Connect from Microsoft Power BI Desktop 2.42" on page 1352
- "Connect from Excel" on page 1353

> **Note**
>
> These instructions provide general guidance only and are specific to the versions of the third-party applications indicated.
>
> For detailed information about creating ODBC connections in a third-party application, consult the Help for the application.

Connect from Tableau Desktop 10.1

1. In the Tableau **Connect** panel, under **To a Server**, click **More**.
2. Click **Other Databases (ODBC)**.

3. In the **Other Databases (ODBC)** dialog box, select **DSN**, and in the **DSN** dropdown list select one of the following:
   - To use a pre-existing data connection to an Analytics project, select the name of the connection, and click **Connect**.

     > **Note**
     > You must already have created the data connection. For more information, see "Create a data connection (DSN)" on page 1349.

   - To create a data connection to an Analytics project on the fly, select **ACL ODBC** or **ACL ODBC 64** and click **Connect**.

     > **Note**
     > If both **ACL ODBC** and **ACL ODBC 64** appear, select the one that matches the bitness of your version of Tableau (32-bit or 64-bit). For more information, see "Default Analytics data connection" on the previous page.

4. If you selected **ACL ODBC** or **ACL ODBC 64**, in the **Open Project File** dialog box, navigate to an Analytics project (.acl), select it and click **Open**.
5. In the **Other Databases (ODBC)** dialog box, click **Sign In**.

   Tableau connects to the Analytics project.
6. Optional. If you want to connect to more than one Analytics project at the same time, in the **Data Source** tab, in the **Connections** panel, click **Add**, and repeat steps 2 to 5.
7. In the **Data Source** tab, in the **Database** dropdown list, select the Analytics project you are connected to.

   If you are connected to more than one Analytics project, select the appropriate project first in the **Connections** panel.
8. In the **Table** panel, do one of the following:
   - **To list all the tables in the Analytics project:** click **Search**  .
   - **To search for a specific table:** type the name of a table and press **Enter**.

     > **Tip**
     > The table name search is case-sensitive.
     > If an **Exact** search is not returning any tables, try **Contains** or **Starts with**.

9. Drag a returned Analytics table to the working area.
10. Click **Update Now** in the data preview area to see the data in the Analytics table.
11. Perform any additional tasks you require:
    - Add additional tables to the working area and join them

      If required, you can join tables from different Analytics projects.
    - Filter data
    - Update field names

For detailed information about joining tables or other data preparation tasks, consult the Tableau Desktop Help.

> **Note**
> Joining can be slow if one or both tables are large.

12. Click **File > Save** and save the Tableau workbook.

Connect from Microsoft Power BI Desktop 2.42

1. In the Power BI **Home** tab, click the **Get Data** dropdown list, and select **More**.
2. In the **Get Data** dialog box, select **Other > ODBC**, and click **Connect**.
3. In the **From ODBC** dialog box, do one of the following:
   - To use a pre-existing data connection to an Analytics project, select the name of the connection from the **Data source name (DSN)** dropdown list, and click **OK**.

     > **Note**
     > You must already have created the data connection. For more information, see "Create a data connection (DSN)" on page 1349.

   - To create a data connection to an Analytics project on the fly, select **ACL ODBC** or **ACL ODBC 64** from the **Data source name (DSN)** dropdown list, click **Advanced options**, enter the appropriate **Connection string**, and click **OK**.

     The connection string must use this format: `DBF=;DBQ=<Analytics project path and filename.acl>`

     For example: `DBF=;DBQ=C:\Users\john_smith\Documents\ACL Data\Sample Data Files\Sample Project.acl`

     > **Note**
     > If both **ACL ODBC** and **ACL ODBC 64** appear, select the one that matches the bitness of your version of Power BI (32-bit or 64-bit). For more information, see "Default Analytics data connection" on page 1350.

4. If the **ODBC driver** dialog box appears, do the following:
   a. Select **Windows**.
   b. Leave **Use my current credentials** selected.
   c. Click **Connect**.

   Power BI connects to the Analytics project.
5. In the **Navigator** dialog box, expand the node containing the Analytics project tables and select one or more tables to connect to.

   When you highlight a table, a preview of the table data appears in the right-side Preview pane.

6. Do one of the following:
   - Click **Load** to load the selected table or tables into Power BI.
   - Click **Edit** to edit the ODBC query. When you have finished editing the query, click **Close & Apply**.

   Multiple tables are separately loaded into Power BI. If required, you can relate tables in Power BI. In some cases, table relations are automatically generated.

   For detailed information about relating tables or editing the ODBC query, consult the Power BI Desktop Help.
7. Optional. If you want to connect to more than one Analytics project at the same time, repeat steps 1 to 6.

   If required, you can relate tables from different Analytics projects in Power BI.
8. Save the Power BI file.

Connect from Excel

> **Note**
> The steps for connecting from Excel may vary slightly from the steps below, depending on your version of Excel.

1. In the Excel **Data** tab, click the **From Other Sources** dropdown list, and select **From Microsoft Query**.
2. In the **Choose Data Source** dialog box, make sure **Use the Query Wizard to create/edit queries** is selected.
3. In the **Databases** tab, do one of the following:
   - To use a pre-existing data connection to an Analytics project, select the name of the connection, and click **OK**.

     > **Note**
     > You must already have created the data connection. For more information, see "Create a data connection (DSN)" on page 1349.

   - To create a data connection to an Analytics project on the fly, select **ACL ODBC** or **ACL ODBC 64**, and click **OK**.

     > **Note**
     > If both **ACL ODBC** and **ACL ODBC 64** appear, select the one that matches the bitness of your version of Excel (32-bit or 64-bit). For more information, see "Default Analytics data connection" on page 1350.

4. If you selected **ACL ODBC** or **ACL ODBC 64**, in the **Open Project File** dialog box, navigate to an Analytics project (.acl), select it and click **Open**.
5. In the **Query Wizard**, follow the on-screen instructions to do the following:
   - Select the tables or columns that you want to import from the Analytics project.
   - Join tables if you select more than one Analytics table.

- Optional. Filter the data that will be imported.
- Optional. Specify a sort order for the imported data.

For detailed information about using the **Query Wizard**, consult the Excel Help.

> **Note**
> If you want to connect to multiple Analytics tables without joining them during the ODBC connection process, you must perform separate connection operations.

6. In the **Import Data** dialog box, specify any options you require and click **OK**.

   Excel runs the ODBC query and populates an Excel worksheet with the Analytics data.
7. Click **File > Save** and save the Excel workbook.

Reference information

This section contains information that applies throughout Analytics:

"Character and size limits in Analytics" on the facing page	Application-enforced limits to certain input ranges and user-defined parameters in Analytics
"Reserved keywords" on page 1364	Keywords that are reserved for Analytics internal processes and cannot be used for field or variable names
"Variables created by Analytics commands" on page 1366	System variables, and stored values, that are automatically created when you execute certain Analytics commands
"Keyboard shortcuts" on page 1370	Keyboard shortcuts that can be used in Analytics

Character and size limits in Analytics

There are limits to certain input ranges and user-defined parameters in Analytics. The sections below explain the limits and the results when you exceed them.

Remarks

These additional details apply to Analytics character and size limits:

- **Non-Unicode versus Unicode** – specified limits apply to both the non-Unicode and Unicode editions of Analytics, except where noted.
- **Characters and bytes** – Character limits apply to the single-byte character encoding used in the non-Unicode edition, and the double-byte character encoding used in the Unicode edition. In other words, a 256-character limit means 256 characters in either edition.
- **Location of source data** – Analytics limits are the same whether the source data resides in an Analytics data file (.fil), or in an external file or a database. External data sources may impose their own limits that differ from the Analytics limits.

Analytics project limits

Parameter	Limit	Result when limit exceeded
Maximum file size of an Analytics project	1GB (tested successfully)	
Maximum number of tables in an Analytics project	5,150 (tested successfully)	
Maximum length of an Analytics project path and project name	259 characters Includes the file path, project name, and file extension (.acl)	Truncation to 259 characters
Maximum length of an Analytics project path and command log name	259 characters Includes the file path, command log name, and file extension (.log)	Error message
Maximum length of an Analytics project folder name	64 characters	Truncation to 64 characters
Valid characters in an Analytics project folder name	Alphanumeric characters, and the underscore character (_)	Replacement of an invalid character with an underscore

Parameter	Limit	Result when limit exceeded
	The name cannot contain any special characters or spaces, or start with a number	character (_)
Maximum length of a note (table, layout, view, project)	32,765 characters	Truncation to 32,765 characters
Maximum length of a field note	4,996 characters	Truncation to 4,996 characters

Table limits

Parameter	Limit	Result when limit exceeded
Maximum number of records in a table	2,147,483,647 (tested success-fully)	
Maximum record length	32,767 characters (non-Unicode edition) 16,383 characters (Unicode edition)	Analytics resets to 32,767/16,383 characters
Maximum number of fields in a table	1,498 (tested successfully)	
Maximum length of a table layout name	64 characters	Truncation to 64 characters
Valid characters in a table layout name	Alphanumeric characters, and the underscore character (_) The name cannot contain any special characters or spaces, or start with a number	Replacement of an invalid character with an underscore character (_)
Maximum length of a source data file path and name	255 characters	Analytics stops working

View limits

Parameter	Limit	Result when limit exceeded
Maximum number of columns in a view	No specific limit on the number of columns Columns cannot exceed the	Error message

Parameter	Limit	Result when limit exceeded
	maximum record length of 32,767 characters (non-Unicode)/16,383 characters (Unicode) **Note** After the initial import of data, the default view displays a maximum of 256 columns, even if a greater number of columns were imported. If required, you can manually add the additional columns to the view. For more information see "Add columns to a view" on page 778.	
Maximum length of a view name	64 characters	Truncation to 64 characters
Valid characters in a view name	Alphanumeric characters, and the underscore character (_) The name cannot contain any special characters or spaces, or start with a number	Replacement of an invalid character with an underscore character (_)

Field and column limits

Parameter	Limit	Result when limit exceeded
Maximum field length	32,767 characters (non-Unicode edition) 16,383 characters (Unicode edition)	Error message
Maximum length of a field name	256 characters 64 characters for ODBC imports 64 characters for exports to Microsoft Excel and Access	Truncation to 256 characters Truncation to 64 characters Error message
Valid characters in a field name	Alphanumeric characters, and the underscore character (_) The name cannot contain any special characters or spaces, or start with a number	Replacement of an invalid character with an underscore character (_)

Parameter	Limit	Result when limit exceeded
Maximum length of an Alternate Column Title (display name)	256 characters	Truncation to 256 characters
Maximum width of a computed field	255 characters	Error message and value resets to 1 if value entered exceeds 255 characters
Maximum width of a column in a view	255 characters	Error message and value resets to 1 if value entered exceeds 255 characters Columns can temporarily be extended beyond 255 characters by dragging the column boundary in the view, but will be reset in the Modify Columns dialog box
Maximum number of digits in a numeric field (including decimal places)	22	Truncation or error message

Filter limits

Parameter	Limit	Result when limit exceeded
Maximum length of a filter name	256 characters	Truncation to 256 characters
Valid characters in a filter name	Alphanumeric characters, and the underscore character (_) The name cannot contain any special characters or spaces, or start with a number	Replacement of an invalid character with an underscore character (_)

Index limits

Parameter	Limit	Result when limit exceeded
Maximum length of an index name	64 characters	Truncation to 64 characters
Valid characters in an index name	Alphanumeric characters, and the underscore character (_) The name cannot contain any special characters or spaces, or start with a number	Replacement of an invalid character with an underscore character (_)

Parameter	Limit	Result when limit exceeded
Maximum length of index key field	247 characters	Error message
Maximum number of index key fields	246 characters total length	Error message

Quick sort limits

Parameter	Limit	Result when limit exceeded
Maximum length of quick sort field	247 characters	**Quick Sort** menu option disabled

Expression limits

Parameter	Limit	Result when limit exceeded
Maximum number of characters in an expression	8188 characters	Error message
Maximum number of characters in an IF or WHILE condition	4094 characters	Error message
Maximum number of characters in an individual Condition or Value	4094 characters	Error message and resets to 2 KB
Maximum number of characters in the conditions contained in a conditional computed field	32,000 characters	Error message

Variable limits

Parameter	Limit	Result when limit exceeded
Maximum length of a variable name	31 characters	Error message
Valid characters in a variable name	Alphanumeric characters, and the underscore character (_) The name cannot contain any special characters or spaces, or start with a number	Replacement of an invalid character with an underscore character (_), or error message

Script limits

Parameter	Limit	Result when limit exceeded
Maximum length of a script name	64 characters	Truncation to 64 characters
Valid characters in a script name	Alphanumeric characters, and the underscore character (_) The name cannot contain any special characters or spaces, or start with a number	Replacement of an invalid character with an underscore character (_)

Workspace limits

Parameter	Limit	Result when limit exceeded
Maximum length of a workspace name	64 characters	Truncation to 64 characters
Valid characters in a workspace name	Alphanumeric characters, and the underscore character (_) The name cannot contain any special characters or spaces, or start with a number	Replacement of an invalid character with an underscore character (_)

Date limits

Parameter	Limit	Result when limit exceeded
Range of supported dates	1 Jan 1900 to 31 Dec 9999 (inclusive)	Error message

Dialog Builder limits

Parameter	Limit	Result when limit exceeded
Maximum number of Drop-down list options	32 list items	Will not accept additional text beyond 32 items in the Label List

Parameter	Limit	Result when limit exceeded
Maximum size of a single Drop-down Item Label	256 characters	Truncated to 256 characters

Option dialog box limits

Parameter	Limit	Result when limit exceeded
Maximum Buffer Size	255 KB	Error message

Reserved keywords

Analytics reserves certain keywords for special purposes. You cannot name fields or variables with these reserved keyword values.

If you add a suffix to a reserved keyword, then you can use it as a field or variable name. For example, the name "Field" is not permitted, but "Field_1" or "Field_2" are permitted.

> **Note**
> In some cases, you are also prevented from using abbreviations of the reserved keywords, such as "Can" (CANCEL), "Form" (FORMAT), or "Rec" (RECORD).

Reserved Keyword	Purpose in Analytics
ALL	Refers to all previously defined fields.
AND	The logical AND operator.
AS	Assigns a display name to the output field or expression.
AXRunByUser	A system variable that stores the username of the user running an analytic script on AX Server in the format "*domain\username*".
CANCEL	Cancels the current command.
D	Specifies a descending sort order for the preceding expression or field name.
END	Concludes the input stream and acts like a null line.
EXPR	The prefix for the name of a default output field.
F	Refers to the *false* value of a logical expression.
FIELD/FIELDS	Part of the EXPORT, EXTRACT, JOIN, and SAMPLE commands.
FORMAT	An older name for an Analytics table layout. Cannot be used as a name for an Analytics table.
IF	Specifies a condition.
LINE	Used by the DEFINE COLUMN command to specify whether a field breaks over a specified number of lines.
NODUPS	Suppresses duplicate display values in the break field in an Analytics report.
NOT	The logical NOT operator.

Reserved Keyword	Purpose in Analytics
NOZEROS	Displays or prints zero values in a numeric field or report as blank.
ON	Precedes a field list.
OR	The logical OR operator.
OTHER	Indicates which fields or expressions to include, but not subtotal, in the output of the SUMMARIZE command.
PAGE	Used by the REPORT command to create page breaks.
PICTURE	Specifies a format for a numeric field.
PRIMARY	Specifies a certain type of join.
RECORD	Refers to the entire input record as it exists.
RECORD_ LENGTH	Stores record-length values for use in record-processing operations.
SECONDARY	Specifies a certain type of join.
SUPPRESS	Suppresses the output of numeric totals.
T	Refers to the *true* value of a logical expression.
TAPE	Refers to an older method of accessing data with Analytics. Cannot be used as a name for an Analytics table.
TO	Designates an output file for any command.
WIDTH	Changes the default print width of a specified field or expression.

Variables created by Analytics commands

When you execute certain commands, either by entering information in dialog boxes in Analytics or by running scripts, system variables are automatically created by Analytics. You can use these variables, and the values they contain, when processing subsequent Analytics commands.

The value in a system variable is replaced with an updated value if you execute the same command again.

"Analytics system variables" on the facing page lists the system variables created by Analytics.

> **Note**
> System variables, and the values they contain, are retained for the duration of the current Analytics session only.

Displaying the current value of variables

You can use any of the following methods to display the current values of all system-defined and user-defined variables in an Analytics project:

- Select the **Variables** tab in the **Navigator**
- Enter `DISPLAY VARIABLES` in the command line
- Click **Display Variables**  on the toolbar (requires that you first add the button to the toolbar)

The second and third methods also display the remaining memory available to store variables.

Incrementally numbered system variables

For system variable names that include *n* in "Analytics system variables" on the facing page, *n* is always 1 if commands are executed outside a group – for example, **TOTAL1**.

If you use a group to execute multiple commands, any system variables that result are numbered based on the line number of the command that creates the variable. The first command in a group is considered to be on line number 2.

For example:

- If the Total command is the third command in a group, the results are contained in the variable **TOTAL4**.
- If a second Total command is the fifth command in the group, the results are contained in the variable **TOTAL6**.

Analytics system variables

The following table lists the system variables created by Analytics, the commands that generate them, and the values the variables contain.

> **Note**
> If you run the Statistics command on more than one field simultaneously, the system variables contain values for the first field you specify.

System variable	Command	Value
WRITE*n*	◦ Any command that outputs a table ◦ Verify ◦ Sequence	◦ The number of records in the output table ◦ The number of data validity errors (Verify) ◦ The number of sequence errors (Sequence)
OUTPUTFOLDER	◦ Any command that outputs an Analytics table	The path to the Analytics project folder in the **Navigator** that contains the output table. This a DOS-style path that uses the format **/folder-name/subfoldername**, in which the initial slash (/) indicates the root level in the **Overview** tab. **Tip** Use the SET FOLDER command to specify a different output folder or to create a new output folder.
COUNT*n*	◦ Count ◦ Statistics	The number of records tallied.
ACL_Ver_Major	◦ Display Version (Analytics version numbers are in the format *major.minor.patch*)	The major version of Analytics that is currently running.
ACL_Ver_Minor		The minor version of Analytics that is currently running.
ACL_Ver_Patch		The patch version of Analytics that is currently running.
ACL_Ver_Type		The edition of Analytics that is currently running: ◦ A value of '0' indicates the non-Unicode edition ◦ A value of '1' indicates the Unicode edition
MLE*n*	◦ Evaluate	**Monetary unit sampling** Most Likely Error amount (projected misstatement) **Record sampling** Upper error limit frequency rate (computed upper deviation rate)
UEL*n*		**Monetary unit sampling**

System variable	Command	Value
		Upper Error Limit amount (upper misstatement limit)
RETURN_CODE	◦ Execute	The code returned by an external application or process run using the Execute command. Return codes are numbers generated by the external application or process and sent back to Analytics to indicate the outcome of the external process. Analytics does not generate the return code. Typical return codes are integer values that map to specific notifications or error messages. For example, the return code "0" could mean "The operation completed successfully". The return code "2" could mean "The system cannot find the file specified". Specific return codes and their meanings vary depending on the external application or process. Lists of return codes, also called 'error codes' or 'exit codes', and their meanings, can often be found in the documentation for the associated external application. Lists of return codes can also be found on the Internet. The RETURN_CODE variable is created when the Execute command is used synchronously, but not when the command is used asynchronously.
GAPDUP*n*	◦ Gaps ◦ Duplicates ◦ Fuzzy Duplicates	The total number of gaps, duplicates, or fuzzy duplicate groups.
SAMPINT*n*	◦ Size	The required sample interval.
SAMPSIZE*n*		The required sample size.
ABS*n*	◦ Statistics	The absolute value of the first specified field.
AVERAGE*n*		The mean value of the first specified field.
HIGH*n*		The 5th highest value in the first specified field. The 5th highest is the default setting. The setting can be changed using the **# of High/Low** option in the **Statistics** dialog box. **Note** When Analytics identifies the highest value, duplicate values are not factored out. For example, if values in descending order are 100, 100, 99, 98, the 3rd highest value is 99, not 98.
LOW*n*		The 5th lowest value in the first specified field. The 5th lowest is the default setting. The setting can be

System variable	Command	Value
		changed using the **# of High/Low** option in the **Statistics** dialog box. **Note** When Analytics identifies the lowest value, duplicate values are not factored out. For example, if values in ascending order are 1, 1, 2, 3, the 3rd lowest value is 2, not 3.
MAX*n*		The maximum value in the first specified field.
MEDIAN*n*		The median value in the first specified field.
MIN*n*		The minimum value in the first specified field.
MODE*n*		The most frequently occurring value in the first specified field.
Q25*n*		The first quartile value (lower quartile value) in the first specified field.
Q75*n*		The third quartile value (upper quartile value) in the first specified field.
RANGE*n*		The difference between the maximum and minimum values in the first specified field.
STDDEV*n*		The standard deviation of the first specified field.
TOTAL*n*	◦ Total ◦ Statistics	The sum total of the values in the first specified field.

Other system variables

The following variables are system-generated but not created by commands:

- **AXRunByUser** – available in scripts running on AX Server, this variable stores the username of the user running the analytic in the format "*domain**username*"
- **OUTPUTFOLDER** – the current Analytics project output folder

Keyboard shortcuts

The following table lists the keyboard shortcuts that can be used in Analytics.

Shortcut	Location in Analytics	Action
Alt+F4	global	Exit Analytics
Ctrl+Tab	global	Display the next tab
Ctrl+Shift+Tab	global	Display the previous tab
Ctrl+PgDn	**Navigator**	Toggle between the **Overview** tab, the **Log** tab, and the **Variables** tab
	view	Display the next view (if the table has more than one view)
	command results	Display the results as a graph (for commands with graphable results)
Ctrl+PgUp	**Navigator**	Toggle between the **Overview** tab, the **Log** tab, and the **Variables** tab
	view	Display the previous view (if the table has more than one view)
	command results	Display the results as text
Ctrl+1	global	Run a script
Ctrl+2	global	Create a script from table history
Ctrl+3	global	Open the **Count** dialog box
Ctrl+4	global	Open the **Total** dialog box
Ctrl+5	global	Open the **Stratify** dialog box
Ctrl+6	global	Open the **Sequence** dialog box
Ctrl+7	global	Open the **Join** dialog box
Ctrl+8	global	Open the **Histogram** dialog box
Ctrl+9	global	Open the **Sample** dialog box
Ctrl+0	global	Open the **Summarize** dialog box
Ctrl+B	global	Open the most recent Analytics project

Shortcut	Location in Analytics	Action
Ctrl+I	global	Open the **Table Layout** dialog box
Ctrl+O	global	Open an Analytics project
Ctrl+R	global	Open the **Report** dialog box
Ctrl+S	global	Save the current Analytics project
Ctrl+Y	Script Editor	Multiple Redo
Ctrl+Z	Script Editor	Multiple Undo
F1	global	Display context-sensitive online Help
F2	command line Filter text box Script Editor **Expression** text box	Open the **Insert Fields** dialog box
F4	command line Filter text box Script Editor **Expression** text box	Open the **Insert Project Item** dialog box
F5	Script Editor	Run a script
F8	command line Filter text box Script Editor **Expression** text box	Open the **Date & Time Selector** dialog box
F9	Script Editor	Insert/remove a break point
F10	global Script Editor	Show/hide the status bar Step through a script one line at a time
Enter	**Overview** tab	Open the selected item
	command dialog boxes command line	Execute the command

This page intentionally left blank

Scripting in Analytics

Scripting in Analytics

Scripting is how you automate work in Analytics. You can gain a lot of value using Analytics in an ad hoc or manual fashion without ever writing a script. However, to gain the most value, power, and efficiency from Analytics you need to script.

The good news is that Analytics provides tools to make scripting relatively easy, even for a novice.

The power of Analytics scripts

An Analytics script is a series of ACLScript commands that performs a particular task, or several related tasks. Scripts are the key to harnessing the power of Analytics:

Run more than one command at a time	Scripts allow you to assemble multiple Analytics commands and run them in an unbroken sequence.
Automate repetition	Scripts can be run repeatedly, which represents a huge savings of labor if you perform the same analytic tests, and the same analysis-related tasks such as importing and preparing particular data files, on a regular basis.
Ensure consistency	Scripts ensure consistency by executing the same commands in the same sequence every time they are run.
Share analysis	Scripts are portable and sharable. They can be sent to other users, made available in network locations, and copied between Analytics projects.
Disseminate expertise	Scripts allow expert Analytics users to develop analytic tests and share the tests with non-scriptwriting users.
Allow user interaction	Scripts can be designed to prompt users for input, allowing users to run them against their own uniquely named tables and fields, using their own input criteria.
Schedule unattended execution	Scripts can be scheduled, and can run unattended, which allows companies to perform time-intensive data processing overnight, and to set up programs of automated or continuous analysis.

Core scripting resource

The core resource for anyone writing Analytics scripts is the ACLScript language reference, which fully explains every Analytics command, function, and analytic tag, with numerous examples and code snippets:

- **"Commands overview" on page 1526** – every Analytics command
- **"Functions overview" on page 2016** – every Analytics function

- **"Analytic scripts overview" on page 2460** – guidance for developing analytic scripts and analysis apps that can run in Robots, Analytics Exchange, or the Analysis App window

Scripting tutorials

Several tutorials are available to help you get started with scripting in Analytics. See "Get started with scripting" on page 1378.

Working with scripts

For an explanation of the basics of creating scripts in the Analytics user interface, and the various scriptwriting tools, see "Working with scripts" on page 1480.

This page intentionally left blank

Get started with scripting

If you are not sure how to get started with scripting in Analytics, this section provides information to help you, whatever your level is. New users can consult the beginner tutorials, while more advanced users can browse the basics section for a quick overview of how ACLScript works.

The right content for the right audience

Depending on your scripting background and your confidence with Analytics, some topics in this section are better suited to your needs than others.

Complete beginners

For users that want a slow and steady introduction to scripting and Analytics, see the following sections for some interactive beginner-level tutorials:

- "Scripting for complete beginners" on the facing page
- "How to use functions" on page 1424
- "Analytics scripting basics" on page 1397

Intermediate to advanced users

If you are familiar with Analytics or with scripting in general, the following sections are available for quick reference and more involved tutorials:

- "Analytics scripting basics" on page 1397
- "Advanced use of functions" on page 1446
- "Top 30 Analytics functions" on page 1465
- "Search and filter using Analytics functions" on page 1173

Scripting for complete beginners

You can do a lot of powerful things with ACLScript and there are many commands and functions at your disposal, so it is a good idea to start small and learn the practicalities of scripting in Analytics.

Installing Analytics

Before you can start scripting, you need to install Analytics and get it running on your computer. For more information, see "ACL for Windows Installation and Activation Guide" on page 2566.

Opening a sample project

Throughout this section, we assume that you have installed and activated Analytics and that you have access to the sample data that ships with the application.

To open the sample project, navigate to the folder containing the sample projects on your computer and double-click `Sample Project.ACL`.

> **Note**
> By default, the sample data projects are installed at `C:\Users\username\Documents\ACL Data\Sample Data Files` on your local file system.

What is a script?

A script is a list of commands that run in Analytics. Scripts are useful for performing a series of tasks automatically so that you do not need to run each command manually through the user interface. For more on scripts, see "What is a script?" on the next page

Your first Analytics script

Complete this short tutorial to learn the basics of ACLScript. The tutorial deals with opening a table and extracting a subset of records. To view the tutorial, see "Your first Analytics script" on page 1384.

What is a script?

A script is a series of Analytics commands that are executed sequentially and used to automate work within Analytics. Any Analytics command can be contained in a script.

Why should I use a script?

There are a variety of benefits to using a script.

Automate processes

Do you need to perform a series of repetitive tasks or routines on a regular basis? Are you currently performing these tasks manually? If so, you can probably use a script to automate these types of processes. By using a script, you can avoid manual efforts associated with complex routines. The more complex the routine, the more time will be saved by running a script.

Schedule processes

Scheduling scripts is often essential when you are dealing with large data sets. If you are using Analytics Exchange, you can run scripts on a schedule, even outside of work hours. You can also schedule a single script or series of scripts to run at a specific date and time.

Improve accuracy

When performed manually, complex data analysis routines are prone to human error. By using a script, you can ensure process consistency and precision. You can also be absolutely certain that the same instructions will be executed in the same order each time the same script is run.

Reduce complexity

Scripts are able to process complex file structures and make complex computations on data fields. Sometimes, more complex analysis can only be performed with a script. For example, continuous monitoring programs often require scripts to automate processes.

Share analysis

Scripts are portable and sharable. They can be sent to other users, made available in network locations, and copied between Analytics projects.

Allow user interaction

Scripts can be designed to prompt users for input, allowing users to run them against their own uniquely named tables and fields, using their own input criteria.

Capture documentation

Scripts are a great source of documentation for audit reviews, and can be used as part of an audit trail. By creating a script, you are documenting the process of creating the results of an analytic test - which is something that can be easily referenced in the future. You can also add comments to scripts to further supplement the documentation.

Common processes that can be automated by scripts

A script may be something as simple as running a command on a single field, or it may be substantial enough to perform the bulk of the work to achieve your analysis objectives.

Scripts are most commonly used to perform one or more of the following processes:

Import data

You can use a script to import various source files into Analytics, including fixed-width, delimited, report/PDF, Excel, and files accessed via ODBC.

```
COMMENT *** Imports data from a Microsoft Access database file to an Analytics
table named employees_list.
IMPORT ACCESS TO employees_list PASSWORD 1 "C:\ACL DATA\Sample Data Files\em-
ployees_list.fil" FROM "Employees_List.mdb" TABLE "[Employees_List]" CHARMAX
60 MEMOMAX 70
```

Prepare data

You can use a script to prepare data for analysis. Scripts can be used to standardize fields prior to joining or relating tables, remove leading or trailing spaces from values, remove unwanted characters, and convert data types of fields.

```
COMMENT *** Creates a new computed field containing the PO_No value. All lead-
ing blank spaces are removed so that the value is properly left justified.
DEFINE FIELD c_PO_No COMPUTED ALLTRIM(PO_No)
```

Analyze data

Scripts use data analysis commands and functions to achieve analysis objectives. You can use a script to group records, make comparisons, and identify issues, trends, or outliers.

```
COMMENT *** Opens Sales2016Actual table, classifies on Customer Number, sub-
totals on Sales Order Amount, and sends the results to Sales2016ByCustomer.
OPEN Sales2016Actual
CLASSIFY ON Customer_Number SUBTOTAL Sales_Order_Amount TO Sales2016ByCustomer
```

Example script

Scenario

Each month, a client provides you with vendor, invoice, and purchase order information. You need to verify the integrity of the data by ensuring that there are no blanks in the purchase order field.

You decide this is a good opportunity to write a script, given the repetitive nature of the task. You want to have all fields available for analysis and be able to search the purchase order field for blanks.

Process

You create a script that performs the following actions:

1. Opens the Invoice_Amts table.
2. Searches the purchase order field (`PO_No`) for blanks.
3. Extracts records with blank purchase order numbers to a new table (r_Blank_Purchase_Orders), allowing you to follow up with exceptions.

> **Tip**
>
> To easily identify tables, you can use the following naming conventions:
>
> - **Prepared table** - prefix table name with p_
> - **Temporary table** - prefix table name with t_
> - **Results table** - prefix table name with r_

Result

```
COMMENT *** Opens table "Invoice_Amts".
OPEN Invoice_Amts

COMMENT *** Searches for blanks in the purchase order field.
SET FILTER TO ISBLANK(PO_No)

COMMENT *** Extracts results to a new table called "r_Blank_Purchase_Orders".
EXTRACT FIELDS Vendor_Name Invoice_No Payment_Date Invoice_Date Invoice_Amt
Vendor_Name Invoice_No PO_No TO r_Blank_Purchase_Orders
```

Next steps

Complete the short tutorial ""Your first Analytics script" on the next page" and try creating your own script.

Your first Analytics script

This short, simple tutorial shows the basics of ACLScript. The tutorial only deals with opening a table and extracting a subset of records, but ACLScript is capable of much more.

What do you need?

In this tutorial we assume that you have installed and activated Analytics and that you have access to the sample data that ships with the application.

> **Note**
> By default, the sample data projects are installed at `C:\Users\username\Documents\ACL Data\Sample Data Files` on your local file system.

Setting up

Open the sample Analytics project

1. Open ACL for Windows.
2. Click **Open Analytic Project** and from the `ACL Data\Sample Data Files` folder, select **Sample Project.ACL**.

Create your first script

1. In the **Navigator**, from the **Overview** tab, right-click the **Scripts** folder and select **New > Script**.

   New_Script is added to the Navigator and opens in the script editor.
2. Right-click **New_Script**, select **Rename**, and enter `extract_invoices`.

The script logic

In this script, we are going to use ACLScript to:

1. Open the **Ap_Trans** table.
2. Copy all records from the table with an invoice amount greater than 1000.00 and store them in a new table called **Ap_Trans_High**.
3. Open the new table to inspect the results.

Open the Ap_Trans table

Data is stored in tables, so to work with data we need to first open a table. The `OPEN` command signals that you are working with the specified table and makes the table's data available to your script commands:

```
OPEN Ap_Trans
```

Copy this line, paste it into the script editor, and then click **Run**  on the editor toolbar.

If the `Ap_Trans` table opens, your script is working. Close the table and continue.

Extract all records to Ap_Trans_High and close Ap_Trans

Now that the script is working with the `Ap_Trans` table, we can use the `EXTRACT` command to copy records from **Ap_Trans** to a new table called **Ap_Trans_High**:

```
EXTRACT RECORD TO 'Ap_Trans_High'
CLOSE Ap_Trans
```

Copy this line, paste it into the script editor on a line after the `OPEN` command, and then click **Run**  on the editor toolbar.

You should see the `Ap_Trans_High` table appear in the **Navigator** under **Tables > Accounts_Payable**. This new table contains the copied records from `Ap_Trans`.

Extract the subset of records to Ap_Trans_High and close Ap_Trans

At this point, we can add a conditional `IF` parameter to the `EXTRACT` command so that we only copy invoice records with amounts that exceed 1000.00.

Notice how we use the `IF` parameter to test whether the value of the `Invoice_Amount` field is greater than 1000.00. If this test does not evaluate to true, the record is not extracted:

```
EXTRACT RECORD TO 'Ap_Trans_High' IF Invoice_Amount > 1000.00
CLOSE Ap_Trans
```

Copy this line, replace the existing `EXTRACT` command in the script editor with it, and then click **Run** on the editor toolbar.

When prompted, click Yes to overwrite the `Ap_Trans_High` table. The `Ap_Trans_High` table now contains the copied records with amounts exceeding 1000.00 from `Ap_Trans`.

Open the Ap_Trans_High table

We will end the script by opening the new table **Ap_Trans_High** so that you can inspect the results of the `EXTRACT` command. As this is the last action in the script, the table opens and you can review the records:

```
OPEN Ap_Trans_High
```

Copy this line, paste it into the script editor on a line after the `EXTRACT` command, and then click **Run** on the editor toolbar.

The `Ap_Trans_High` now opens when the script completes and you can review the extracted records from `Ap_Trans`.

The full script

```
OPEN Ap_Trans

EXTRACT RECORD TO 'Ap_Trans_High' IF Invoice_Amount > 1000.00
CLOSE Ap_Trans

OPEN Ap_Trans_High
```

Where to next?

- For an overview of the basics of scripting in Analytics, see "Analytics scripting basics" on page 1397
- For advanced training, see the scripting course in Academy

Comparing text data

When you are working with text, it is common to compare one value against another. Because comparison is case-sensitive, text stored in various case formats can be challenging to compare. Analytics provides functions that make comparison more reliable by converting the text you are comparing to normalized case formats.

Text comparison operators

When scripting in Analytics, you can use two operators to compare text values:

- **equality operator (=)** - evaluates to true if the value on the left-hand side of the equality operator is exactly the same as the value on the right-hand side
- **inequality operator (<>)** - evaluates to true if the value on the left-hand side of the inequality operator is not exactly the same as the value on the right-hand side

Both operators are case-sensitive and compare two values: `valueOne <> valueTwo`.

Filtering a table using text comparison

You are working with the following table and you need to filter it so that only records for the Finance department are shown:

Department	Max_Hourly	Min_Hourly	Position
Executive	205.13	166.67	CEO & President
Executive	141.03	89.74	VP, Finance
Finance	24.62	20.51	Accountant
finance	23.08	17.95	Clerk, Cash Disbursements
finance	18.46	14.67	Clerk, Payables
Finance	18.46	14.67	Clerk, Purchasing
Information Systems	23.08	14.36	Technical Support
Information Systems	30.77	23.08	Web Administrator

To filter the table, you create a simple expression using the equality operator (=):

```
COMMENT filters the table to show records where Department is "Finance"
SET FILTER TO Department = "Finance"
```

First filter results

Based on this filter, you expect to see four records in the filtered table, but instead you only see two:

Department	Max_Hourly	Min_Hourly	Position
Finance	24.62	20.51	Accountant
Finance	18.46	14.67	Clerk, Purchasing

Because the equality operator is case-sensitive, records where the Department field contains "finance" are excluded from the results. You need to include these records in the results as well.

Using the LOWER( ) function to help filter the table

To help you perform comparisons, Analytics provides functions that make comparison more reliable by converting the text you are comparing to a known case format, such as lowercase.

To filter the table so that your results include all employees from the Finance department, regardless of case format, you use the same expression but you use the `LOWER( )` function to convert all the values to lowercase:

```
COMMENT filters the table to show records where Department is "finance"
SET FILTER TO LOWER(Department) = "finance"
```

When the expression is evaluated, `LOWER("Finance")` becomes `"finance"` and is then compared to the string on the right-hand side of the equality operator.

Second filter results

When using the `LOWER( )` function in the expression, the filter includes all employees from the Finance department:

Department	Max_Hourly	Min_Hourly	Position
Finance	24.62	20.51	Accountant

Department	Max_Hourly	Min_Hourly	Position
finance	23.08	17.95	Clerk, Cash Disbursements
finance	18.46	14.67	Clerk, Payables
Finance	18.46	14.67	Clerk, Purchasing

Filtering blank date values

Sometimes data is incomplete or optional and therefore fields in Analytics tables may contain empty or blank values. When an expression in Analytics encounters a blank date value, the blank date is treated as the minimum system date value 1900-01-01. Whenever you write an expression that compares dates, you must account for this behavior.

The sales order table

You are working with the following table of sales orders and you want to filter it so that you can analyze all orders that were submitted before the year 2011. If a record has no order date, then it represents a canceled order and you do not want to include it in your filtered records:

Category	Order_Date	Order_ID	Quantity
Office Supplies		3	6
Office Supplies		293	49
Office Supplies	07/23/2012	293	27
Technology	10/15/2010	483	30
Office Supplies	08/28/2010	515	19
Furniture	08/28/2010	515	21
Office Supplies	06/17/2011	613	12
Office Supplies	06/17/2011	613	22
Office Supplies	03/24/2012	643	21
Office Supplies	02/26/2009	678	44

Using a simple filter

When you first attempt to filter the table, you use the following simple expression to exclude any orders from 2011 and later:

> **Tip**
> Notice the backquotes ` that surround the literal date value. You must always surround literal datetime values with this qualifier. For more information, see "Data types" on page 1404.

```
COMMENT filters out records with an order date of Jan 1 2011 or later
SET FILTER TO Order_Date < `20110101`
```

First filter results

Because Analytics treats blank date values as 1900-01-01, and January 1, 1900 is earlier than January 1, 2011, your results include the records with blank **Order_Date** values that you want to exclude:

Category	Order_Date	Order_ID	Quantity
Office Supplies		3	6
Office Supplies		293	49
Technology	10/15/2010	483	30
Office Supplies	08/28/2010	515	19
Furniture	08/28/2010	515	21
Office Supplies	02/26/2009	678	44

Checking for blanks while filtering

Using functions, you can exclude blank date values before you filter out records from 2011 or later.

The `ISBLANK( )` function returns true if a text value is blank, so with some manipulation of the **Order_Date** field, you can exclude blank values:

```
COMMENT excludes blank dates and order dates from 2011 and later
SET FILTER TO NOT ISBLANK(DATETIME(Order_Date)) AND Order_Date < `20110101`
```

When this expression evaluates, the functions run from inside out and several things happen:

1. The `DATETIME( )` function converts the `Order_Date` date value to a text value (`20100828` becomes "20100828").
2. The `ISBLANK( )` function checks if the text value is blank and evaluates to either true or false.

3. The `NOT` operator flips the logical value returned from `ISBLANK( )` so that:
   - If the `Order_Date` is blank (true), then the value is flipped to false and the filter excludes the record
   - If the `Order_Date` is not blank (false), then the value is flipped to true and the filter checks if the date is earlier than 2011 and includes all records that have an **Order_Date** value before January 1, 2011

> **Tip**
> Only those records where the sub-expressions evaluate to true on both sides of the AND operator are included. If either of the sub-expressions evaluate to false, the record is excluded.

Second filter results

The second filter excludes blank date values before testing whether orders were placed before 2011, so the results do not include the canceled orders that were included by the first filter.

Category	Order_Date	Order_ID	Quantity
Technology	10/15/2010	483	30
Office Supplies	08/28/2010	515	19
Furniture	08/28/2010	515	21
Office Supplies	02/26/2009	678	44

Making decisions in scripts

Everything we do involves decision making, and scripting is no different. Sometimes you only want a command to run if some other condition is true, or you may only want to process some records in a table depending on the data they contain. ACLScript provides several methods for decision making in scripts, and they all use conditional expressions.

What is a conditional expression?

A conditional expression is any expression that evaluates to either true or false. Conditional expressions dictate which actions are taken in a script, and are specified by the script writer.

This is a fairly technical definition, but a simple real-world example can help demonstrate what it means:

Example

You are walking down the street and you see someone that you know. It is polite to greet this person, but do you say "Good morning" or do you say "Good afternoon" when you meet?

The answer depends on a simple condition: is it 12:00 PM or later? If the answer is **yes**, then you say "Good afternoon", otherwise you say "Good morning".

In this example, the conditional expression determines the action you take (which greeting you use) based on whether or not it evaluates to true (yes).

The expression from the preceding example could be translated into ACLScript as follows:

```
COMMENT checks if the current time is 12:00 PM or later
NOW() >= `t120000`
```

You can run this example by copying the following line, and then pasting it into the command line of Analytics. Depending on the time of day you do this, the expression evaluates to either true or false:

```
DISPLAY NOW() >= `t120000`
```

> **Tip**
> If the command line is not visible, select **Window > Command Line**.

Once you run the example, you can try changing the literal time value of 12:00 PM in the expression so that it evaluates to the opposite.

Deciding if a command should run

Analytics provides the `IF` command so that you can decide whether or not a command should run. The command requires two inputs:

- a conditional expression
- a command to run if the expression is true

If the conditional expression evaluates to false, then the command does not run.

Saying "Good afternoon"

Continuing with the example from above, try pasting the following code into the command line:

```
IF NOW() >= `t120000` DISPLAY "Good afternoon"
```

If the time is after 12:00 PM, then the `DISPLAY` command prints "Good afternoon" to the output tab. But if it is still morning where you are, then nothing is printed. The script does not run the `DISPLAY` command.

Saying "Good morning"

If your expression evaluated to false, then you may be wondering how to make the command print "Good morning". While some scripting languages provide an "else" construct to handle both true and false cases, ACLScript does not. Instead, you use a second `IF` command with the opposite expression.

Try pasting this expression into the command line:

```
IF NOW() < `t120000` DISPLAY "Good morning"
```

This example works like before, except now if the time is *before* 12:00 PM, then the `DISPLAY` command prints "Good morning".

How does this look in a script?

So far the examples have been limited to the `DISPLAY` command, which is only available from the command line. But in a script, the same principles apply. Instead of printing the greeting to the display tab, in this example the script stores the greeting in a variable called `v_greeting`:

```
COMMENT stores the correct greeting depending on the time of day
IF NOW() >= `t120000` ASSIGN v_greeting = "Good afternoon"
IF NOW() < `t120000` ASSIGN v_greeting = "Good morning"
```

If this script runs before 12:00 PM, then the value stored in the variable is "Good morning", and if it runs at 12:00 PM or later, then the value stored in the variable is "Good afternoon". Try pasting this into your script editor and running it. You can check the value of the variable on the **Variables** tab after it runs.

Deciding which records to process

There are times when you want to decide if the script runs a command or not, as shown above, but there are also time when you want a command to run only on certain records in a table. This is another decision making scenario, but it differs from the one that uses the `IF` command.

In situations where you want to selectively process records, ACLScript provides the `IF` parameter on many commands. When you use this approach, the command requires that you specify a conditional expression as input. The expression is tested against each record in the table, and when it evaluates to true, the record is processed:

```
COMMENT sums the Amount field for records that have a Quantity greater than 5
TOTAL Amount IF Quantity > 5
```

Calculating transactions that occur in the afternoon

You can use the same conditional expression `NOW( ) >= `t120000`` to calculate the sum of all transactions in a table that occurred in the afternoon. Consider the following table of transaction data:

Transaction_Amount	Unit_Cost	Product_No	Transaction_Date	Quantity
618.3	6.87	070104397	2000-11-17 12:00	90
6705.12	6.87	070104677	2000-11-17 9:30	976
7955.46	6.87	070104657	2000-11-17 14:45	1158
4870.83	6.87	070104327	2000-11-17 15:00	709
10531.71	6.87	070104377	2000-11-17 9:57	1533
5734	47	030414313	2000-10-30 1:00	122
2196	18	030414283	2000-10-30 18:25	122

To calculate the sum of the `Transaction_Amount` field, you use the `TOTAL` command:

```
COMMENT Sums the Transaction_Amount field
TOTAL Transaction_Amount
```

This command processes every record in the table and calculates a total of 38,611.42, which is the sum of all transactions.

To add some decision-making to the command, and process only those transactions that occurred at 12:00 PM or later, you add the `IF` parameter to `TOTAL`. You use the same conditional expression from the example at the start, but replace `NOW( )` with the time part of the transaction date:

```
COMMENT Sums the Transaction_Amount field for all transactions occuring in the
afternoon
COMMENT uses functions to extract the time part of the data in the Trans-
action_Date field
TOTAL Transaction_Amount IF CTOT(TIME(Transaction_Date)) >= `t120000`
```

In the command, you have to use some functions to isolate the time part of the transaction date, but once you do that, the decision is the same conditional expression from the example at the start: is it 12:00 PM or later? If the answer is **yes**, then the amount is included in the sum.

This command calculates a total of 15,640.59, which is the sum of all afternoon transactions in the table.

Analytics scripting basics

ACLScript is a command language that allows you to program and automate Analytics commands. The structure and components of ACLScript are simple yet powerful.

> **Note**
> If you are completely new to scripting, consider visiting the Academy for some basic training before jumping into this content. For courses on scripting and using Analytics, visit www.highbond.com.

Commands

Every line in a script executes an ACLScript command and starts with the command name. A command is an instruction to execute an operation in Analytics.

The command name is followed by one or more parameters that are specified as `parameter_name parameter_value`.

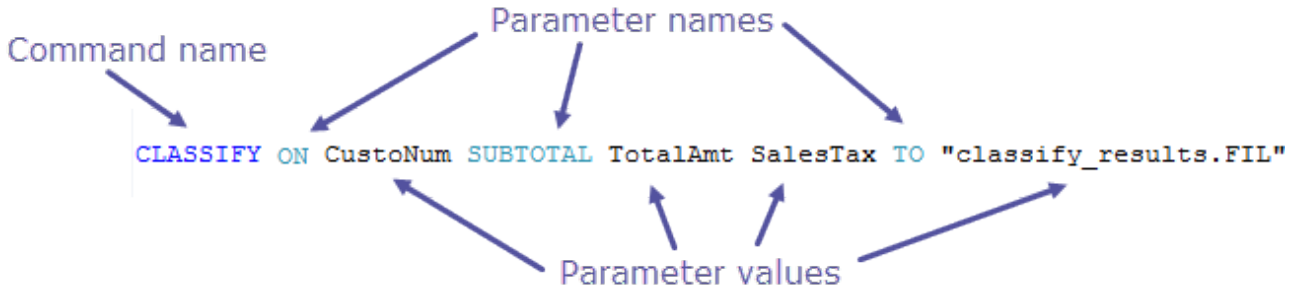
> **Tip**
> Depending on the command, some parameters are required and some are optional. You do not need to specify optional parameters. If they are omitted, the command executes without them. However, if you omit a required parameter, Analytics uses the default value for that parameter.

Example using the CLASSIFY command

The following example shows the `CLASSIFY` command along with following parameters:

- `ON` - specifies which field of the table to classify on
- `SUBTOTAL` - specifies optional fields to subtotal in the output
- `TO` - specifies the table to write the results of the `CLASSIFY` command to

Notice how each parameter is followed by one or more parameter values:



Important command syntax notes

- the first word in a script line must be a command name
- for most commands, the order of parameters that follow the command name does not matter
- most commands require that you open the target table before executing the command, precede these commands with `OPEN table_name`

Comments

Like any scripting language, you can add comments in ACLScript With the `COMMENT` keyword. Use comments to make your code easier to understand and to communicate with anyone who may try to read, use, or understand your script. ACLScript supports two types of comments:

- **single line comments** – all text following `COMMENT` is ignored until the end of the line is reached
- **multiple line comment blocks** – begin with `COMMENT` and each subsequent line is ignored until the `END` keyword, or a blank line, is reached

For more information and examples, see "Comments" on page 1402.

Data types

ACLScript supports four basic data types:

- **logical** – the simplest data type. Logical data expresses a truth value of either true or false
- **numeric** – contain digits from 0 to 9 and, optionally, a negative sign and a decimal point
- **character** – a series of one or more characters
- **datetime** – a date, datetime, or time value expressed in a supported format

Each data type is treated differently by Analytics and can be used in different commands and functions. For more information about data types, see "Data types" on page 1404.

Expressions

An expression is any statement that has a value. The simplest form of expression is a literal such as `2` or `"test"`, however expressions usually appear as calculations and can be as complex as any valid combination of operators, conditions, functions, and values that you can imagine:

```
((2 + (3 - 2)) * 2) > ROOT(9,0)
```

Expressions are typically used in Analytics to populate computed fields or as input for conditional logic. For more information about expressions, see "Expressions" on page 1405.

Functions

Functions are built-in routines that accept a given number of parameters and return a single value. Use functions to manipulate field contents and variables that are used in commands.

> **Note**
> Functions do not modify field data, functions generate and return a new value based on a calculation or algorithm that uses field data or variables as input. Use the value the function returns as input for a command.

Functions start with the function name followed directly by an opening parenthesis, a comma-separated list of 0 or more values that are passed into the function as arguments, and a closing parenthesis.

Example

The `BETWEEN(value, min, max)` function takes three arguments and returns true if the value falls within the range or false if it falls outside the range:

- `value` - the expression or field to test
- `min` - the minimum of the range
- `max` - the maximum of the range

```
BETWEEN(amount, 500, 5000)
```

For more information about functions, see "Functions" on page 1409.

Variables

A variable is temporary storage location used to hold a value. Variables have an associated identifier that lets you reference and work with the value stored in your computer's memory.

ACLScript uses the `ASSIGN` command to create a variable and assign it a value at the same time:

```
ASSIGN v_age_in_years = 3
```

For simplicity you can omit the `ASSIGN` keyword, however `ASSIGN` is implicitly used and the same command runs:

```
v_age_in_years = 3
```

> **Note**
> ACLScript does not support null values. All variables must have an associated value of one of the supported data types. The script interpreter evaluates the data type using the data format and qualifier you use to assign the value. For more information, see "Data types" on page 1404.

Using variables

Once a variable is created, you can reference it anywhere you reference field names or variables. You can also reassign it a new value using the `ASSIGN` command.

```
EXTRACT RECORD TO 'result.fil' IF age > v_age_in_years
v_age_in_years = 5
```

You can also use string interpolation, or variable substitution, to include a variable in a string literal by wrapping the variable name in `%` characters. When Analytics encounters the substituted variable, it replaces the placeholder with its corresponding value:

```
ASSIGN v_table = erp_data
OPEN %v_table%
```

For more information about variables, see "Variables" on page 1411.

Control structures

A control structure is a component of a script that decides which direction to take based on given parameters. ACLScript provides both conditional logic and looping structures.

Conditional logic

ACLScript implements conditional logic as an `IF` command and as an optional parameter on many commands in the language.

> **Tip**
> You use the `IF` command to control if a command runs or not while you use the `IF` parameter to decide which records in a table a command runs against.

IF command

```
IF v_counter > 10 CLASSIFY ON customer_no
```

IF parameter

```
CLASSIFY ON customer_no IF state = 'NY'
```

Looping

The `LOOP` command provides the looping control structure in ACLScript. This command processes the statements inside the loop for as long as the control test expression evaluates to true.

For more information about control structures, see "Control structures" on page 1413

Comments

Like an scripting language, you can add comments in ACLScript With the `COMMENT` keyword. Use comments to make your code easier to understand and to communicate with anyone who may try to read, use, or understand your script.

Comment types

ACLScript supports two types of comments:

- **single line comments** – all text following `COMMENT` is ignored until the end of the line is reached
- **multiple line comment blocks** – begin with `COMMENT` and each subsequent line is ignored until the `END` keyword, or a blank line, is reached

Single line comments

Use single line comments to describe individual steps in your script or to describe variables:

```
COMMENT *** the start date for the analysis period
v_Start_Date = `20150101`
```

Multiple line comment blocks

Use multiple line comment blocks to describe scripts or script sections.

```
COMMENT
**********************
** This section of the script prepares data for import
**********************
END
```

Header comment blocks

It is good practice to include a header comment block that contains key script information at the start of each script:

```
COMMENT
*************************************************************
*** Script Name:   {App_ID}{Script name}
*** Parameters:    {Detailed description}
*** Output:        {Describe parameters}
*** Written By:    {Name}, ABC Corporation, {Month YYYY}
*** Modified By:   {Name}, ABC Corporation, script purpose and logic
*** Version:       1.1.1 {app_ver.script_ver.defect.fix}
*************************************************************
END
```

Data types

ACLScript supports four basic data types: logical, numeric, character, and datetime.

Type	Description	Limit	Qualifier	Examples
Character	A series of one or more characters.	32,767 bytes	Single quotation marks, or double quotation marks	◦ 'John Doe' ◦ "John Doe"
Numeric	Numeric values contain digits from 0 to 9 and, optionally, a negative sign and a decimal point.	22 digits	No qualifier	◦ 100 ◦ -5 ◦ 5.01 ◦ 22222.1232
Datetime	A date, datetime, or time value expressed in a supported format.	◦ Minimum = 1900-01-01 ◦ Maximum = 9999-12-31	◦ Backquotes ◦ Leading 't', or a single blank space, for time values	◦ `20160101` ◦ `141231` ◦ `t2359` ◦ `20141231T235959` ◦ `20141231 235959`
Logical	The simplest data type. Logical data expresses a truth value of either true or false. Comparison operators such as '=', '>', and '<' return logical values.	◦ T ◦ F	No qualifier	`ASSIGN v_truth = 5 > 4` evaluates to `T`

Expressions

An expression is any statement that has a value. The simplest form of expression is a literal, however expressions can be as complex as any legal combination of operators, conditions, functions, and values you can imagine.

Expression components

Literal values

A literal value is a value written exactly as it is meant to be interpreted, such as the character literal `'my value'`. For information about literals, see "Data types" on the previous page.

Operators

Operators are symbols that tell the script interpreter to perform arithmetic, string, comparison, or logical evaluation of the specified values:

Operator type in order of precedence	Operators in order of precedence	Examples
Parenthesis	◦ () specifies precedence ◦ () function operator	`(5 + 3) * 2`
Unary	◦ NOT logical ◦ - negation	`v_truth = NOT (3 < 2)`
Arithmetic	◦ ^ exponentiation ◦ * multiplies, / divides ◦ + adds, - subtracts **Note** Multiplicative operators have equal precedence with each other and evaluate from left to right. Additive operators have equal precedence with each other and evaluate from left to right.	`1 + 5 - 3 * 2`
String	+ concatenates	`"This is" + " my script"`

Operator type in order of precedence	Operators in order of precedence	Examples
Comparative	◦ < less than ◦ > greater than ◦ = equality ◦ >= greater than or equal to ◦ <= less than or equal to ◦ <> not equal **Note** Comparative operators have equal precedence with each other and evaluate from left to right.	`IF amount <> 100`
Binary logical	◦ AND or & ◦ OR or |	`IF amount > 5 AND amount < 10`

Functions

Expressions are evaluated using the values returned by functions. Functions execute with the highest precedence of any expression component. For more information about functions, see "Functions" on page 1409.

Example expressions

Evaluates to 6

```
(2 + (3 - 2)) * 2
```

Evaluates to true

```
((2 + (3 - 2)) * 2) > ROOT(9,0)
```

Evaluates to 'ACLScript tutorial'

```
'AC' + 'LScri' + 'pt ' + 'tutorial'
```

Defining computed fields with expressions

Use **computed fields** to create additional data fields in the currently open table with an expression. A computed field is a field that is appended to the open table and populated with the value of the specified expression.

Computed field syntax

```
DEFINE FIELD name COMPUTED expression
```

- **name** - the name of the computed field to generate
- **expression** - the computation or calculation used to generate the value for the field

Example computed field

```
DEFINE FIELD c_full_name COMPUTED first_name + ' ' + last_name
```

> **Tip**
> Prefix computed field names with `c_` to identify them as computed data rather than original source data.

Defining conditional computed field values

You can also use conditions with computed fields to define the value for different cases:

```
DEFINE FIELD c_total COMPUTED

  amount * ca_tax_rate IF state = 'CA'
  amount * ny_tax_rate IF state = 'NY' OR state = 'NJ'
amount * general_rate
```

When the first conditional expression evaluates to true, the value specified for that case is used. In this example, `amount * general_rate` is the default value used when neither of the conditional expressions evaluate to true.

> **Note**
> You must add an empty line between the line command and the conditions unless you include the `IF`, `WIDTH`, `PIC`, or `AS` parameters on the `DEFINE FIELD` command. For more information, see "DEFINE FIELD . . . COMPUTED command" on page 1633.

Functions

Functions are built-in routines that accept a given number of parameters and return a single value. Use functions to manipulate field contents and variables that are used in commands.

> **Note**
> Functions do not modify field data, functions generate and return a new value based on a calculation or algorithm that uses field data or variables as input. Use the value the function returns as input for a command.

Function syntax

Functions start with the function name followed directly by an opening parenthesis, a comma-separated list of 0 or more values that are passed into the function as arguments, and a closing parenthesis.

Example

The `BETWEEN(value, min, max)` function takes three arguments and returns true if the value falls within the range or false if it falls outside the range:

- `value` - the expression or field to test
- `min` - the minimum of the range
- `max` - the maximum of the range

```
BETWEEN(amount, 500, 5000)
```

Function arguments

An argument of a function is a specific input value passed into the function.

Function arguments are passed to functions via an argument list. This is a comma-delimited list of literal values, variables, or expressions that evaluate to values of the parameter data type. For more information about working with data types, see "Data types" on page 1404.

> **Note**
> If your project works with European number formats, or if you are writing scripts that are portable across regions, separate function arguments with a space character instead of a comma unless you are passing in a signed numeric value. Functions accepting signed numeric values require an explicit delimiter.

Functions vs commands

The distinction between commands and functions is subtle but critical to using ACLScript:

Functions	Commands
Use fields, values, or records as input and generate a new value that is returned.	Use tables as input and generate new records and tables.
Used in expressions, computed fields, command parameter values, variables, and filters to assist and modify command execution.	Used to analyze data, import data, and produce results.
Cannot be an independent step in a script.	Can be an independent step in a script.

Variables

A variable is temporary storage location used to hold a value. Variables have an associated identifier that lets you reference and work with the value stored in your computer's memory.

How variables work in ACLScript

Creating a variable and assigning a value

ACLScript uses the `ASSIGN` command to create a variable and assign it a value at the same time:

```
ASSIGN v_age_in_years = 3
```

For simplicity you can omit the `ASSIGN` keyword, however `ASSIGN` is implicitly used and the same command runs:

```
v_age_in_years = 3
```

> **Note**
> ACLScript does not support null values. All variables must have an associated value of one of the supported data types. The script interpreter evaluates the data type using the data format and qualifier you use to assign the value. For more information, see "Data types" on page 1404.

Using variables

Once a variable is created, you can reference it anywhere you reference field names or variables. You can also reassign it a new value using the `ASSIGN` command.

```
EXTRACT RECORD TO 'result.fil' IF age > v_age_in_years
v_age_in_years = 5
```

You can also use string interpolation, or variable substitution, to include a variable in a string literal by wrapping the variable name in `%` characters. When Analytics encounters the substituted variable, it replaces the placeholder with its corresponding value:

```
ASSIGN v_table = erp_data
OPEN %v_table%
```

Types of variables

Analytics uses the following types of variables:

- **system-generated variables** – automatically created after executing a command
- **permanent variables** – remain in your computer's memory until you delete them and persist after closing the Analytics project

> **Note**
> To define a permanent variable, prefix the identifier with an '_': `_v_company_name = 'Acme'`.

- **session variables** – remain in your computer's memory until you delete them or until the Analytics project is closed

Variable identifiers

Variable identifiers are case-insensitive and follow certain conventions related to the type of variable:

- system-generated variable identifiers use all caps: `OUTPUTFOLDER`
- permanent variable identifiers must have a '_' prefix: `_v_permanent`
- session variable identifiers use the format `v_varname` by convention but you are not restricted to this naming convention

Viewing variable values

During script development or while debugging, it can be useful to track variable values as the script executes. To capture variable values in the script log file, use the `DISPLAY` command:

```
DISPLAY v_age_in_years
```

When the script encounters this command, it writes the command to the log file. To view the variable value at that stage of script execution, click the entry in the log.

> **Tip**
> You can also use variables to help debug by inserting breakpoints in your script and inspecting the variable values on the **Variables** tab of the **Navigator**.

Control structures

A control structure is a component of a script that decides which direction to take based on given parameters. ACLScript provides both conditional IF logic and looping structures.

Conditional logic using IF

ACLScript implements conditional logic as an `IF` command and as an optional parameter on many commands in the language:

- **command** – controls whether or not a command runs
- **parameter** – decides which records in a table to execute the command against

The IF command

When using the `IF` command, you specify a conditional expression followed by the command to execute if the expression evaluates to true:

```
IF v_counter > 10 CLASSIFY ON customer_no
```

This conditional structure controls which code executes, so you can use the `IF` command when you want to process an entire table based on the test expression. If the expression evaluates as true, the command is run against all records in the table. For more information about the `IF` command, see "IF command" on page 1754.

IF parameter

Many commands accept an optional `IF` parameter that you can use to filter which records the command is executed against:

```
CLASSIFY ON customer_no IF state = 'NY'
```

When this statement executes, the script classifies all records in the table where the value of the `state` field is `'NY'`.

Looping

The LOOP command

The `LOOP` command provides the looping control structure in ACLScript.

> **Note**
> The `LOOP` command must execute within the `GROUP` command, it cannot standalone.

This command processes the statements inside the loop for as long as the specified `WHILE` expression is true:

```
ASSIGN v_counter = 10
GROUP
  LOOP WHILE v_counter > 0
    v_total = v_total + amount
    v_counter = v_counter - 1
  END
END
```

This structure iterates 10 times and adds the value of the `amount` field to the variable `v_total`. At the end of each iteration, the `v_counter` variable is decremented by 1 and then tested in the `WHILE` expression. Once the expression evaluates to false, the loop completes and the script progresses.

When the loop completes, `v_total` holds the sum of the 10 records' `amount` fields.

For more information about looping, see "LOOP command" on page 1868.

LOOPING with a subscript

Sometimes the `LOOP` command does not provide the exact looping functionality you may require. In this case, you can also call a separate Analytics script to execute a loop using the `DO SCRIPT` command: `DO SCRIPT scriptName WHILE conditionalTest`.

You can use one of the following common methods to control when your loop ends:

- **flag** – the loop continues until the logical flag variable is set to FALSE
- **counter** – the loop continues until an incrementing or decrementing variable crosses a conditional threshold

For more information about calling subscripts, see "DO SCRIPT command" on page 1673.

Example

You need to import all the CSV files in the `C:\data` folder into your project. You can use the `DIRECTORY` command to get a list of files from the folder, however you cannot use the `IMPORT` command inside the `GROUP` structure. You need an alternative way of looping through the table that `DIRECTORY` creates.

To achieve this, you create a main script that:

1. Executes the `DIRECTORY` command and saves the results to a table.
2. Gets the number of records in the table to use as a counter.
3. Calls a subscript once per record in the table to execute the `IMPORT` command against the current record.

Main script

```
COMMENT Main script

DIRECTORY "C:\data\*.csv" TO T_Table_To_Loop
OPEN T_Table_To_Loop
COUNT
v_Num_Records = COUNT1
v_Counter = 1
DO SCRIPT Import_Subscript WHILE v_Counter <= v_Num_Records
```

Import subscript

```
COMMENT Import_Subscript

OPEN T_Table_To_Loop
LOCATE RECORD v_Counter

COMMENT code to import CSV file in record goes here...

ASSIGN v_Counter = v_Counter + 1
```

Variables are shared among all scripts that run in the project, so the main script calls the subscript until the value of `v_Counter` exceeds the value of `v_Num_Records`. Each time the subscript executes, it increments `v_Counter`.

This structure allows you to call the `IMPORT` command against each record while looping through the table. When the main script completes, you have imported all CSV files from the `C:\data` folder.

Grouping and looping

The GROUP and LOOP commands provide two ways to execute a series of commands repeatedly. GROUP performs a single iteration of one or more commands against each record. LOOP performs multiple iterations of a series of commands against a single record, and can only be used inside a GROUP block.

A simple example of GROUP

You have a table of invoice data called **Ap_Trans**. Using this data, you need to calculate a running total of invoice amounts:

Vendor_Number	Vendor_Name	Invoice_Number	Date	Amount
11663	More Power Industries	5981807	2000-11-17	618.30
13808	NOVATECH Wholesale	2275301	2000-11-17	6705.12
12433	Koro International	6585673	2000-11-17	7955.46

To calculate this amount, you use the GROUP command. Inside each iteration of GROUP, you:

1. Calculate the running total as of the current record.
2. Extract the invoice number, amount, date, and running total to a results table.

```
OPEN Ap_Trans

COMMENT set the initial value of running total to zero END
ASSIGN v_running_total = 0.00

COMMENT iterate over each record in the table and then calculate and extract
the running total END
GROUP
  ASSIGN v_running_total = v_running_total + Amount
  EXTRACT Invoice_Number, Amount, Date, v_running_total AS "Running total" TO
results1
END
```

When the script runs, the commands inside the GROUP block are processed against each record in the table, from top to bottom, and the running total is calculated and extracted. If we could walk through GROUP as it runs, this is how it would look:

First iteration of GROUP: running total = 0.00 + 618.30

The GROUP adds the invoice amount of the first record to the initial running total of 0.00 and extracts the fields to the results table:

Vendor_Number	Vendor_Name	Invoice_Number	Date	Amount
11663	More Power Industries	5981807	2000-11-17	618.30
13808	NOVATECH Wholesale	2275301	2000-11-17	6705.12
12433	Koro International	6585673	2000-11-17	7955.46

Second iteration of GROUP: running total = 618.30 + 6705.12

The GROUP block adds the invoice amount of the second record to the new running total of 618.30 and extracts the fields to the results table:

Vendor_Number	Vendor_Name	Invoice_Number	Date	Amount
11663	More Power Industries	5981807	2000-11-17	618.30
13808	NOVATECH Wholesale	2275301	2000-11-17	6705.12
12433	Koro International	6585673	2000-11-17	7955.46

Third iteration of GROUP: running total = 7323.42 + 7955.46

The GROUP block adds the invoice amount of the third record to the new running total of 7323.42 and extracts the fields to the results table:

Vendor_Number	Vendor_Name	Invoice_Number	Date	Amount
11663	More Power Industries	5981807	2000-11-17	618.30
13808	NOVATECH Wholesale	2275301	2000-11-17	6705.12
12433	Koro International	6585673	2000-11-17	7955.46

Final results table

After GROUP has processed the final record in the table, you have the following results table:

Invoice_Number	Amount	Date	Running_total
5981807	618.30	2000-11-17	618.30
2275301	6705.12	2000-11-17	7323.42
6585673	7955.46	2000-11-17	15278.88

Handling different cases using GROUP IF

Using the same **AP_Trans** table as above, you now need to calculate running totals for three types of invoices:

- High value (greater than or equal to 1000.00)
- Medium value (between 100.00 and 1000.00)
- Low value (less than 100.00)

The GROUP command provides an IF/ELSE structure to handle different cases. You provide the conditional expressions to test, and if a record evaluates to true, then the commands inside the block run.

How cases are tested

Cases are tested **from top to bottom**, and a record can only be processed by one IF/ELSE block. The first case that evaluates to true for the record is the one that processes the record:

1. When GROUP processes the first record, it tests it against the first IF condition (`Amount >= 1000`). If this evaluates to true, then the code for this case runs and no other cases are tested.
2. If the first case evaluates to false, then the next ELSE IF condition (`Amount >= 100`) is tested. Likewise, if this test evaluates to true, then the code for this case runs and no other cases are tested.
3. Finally, if none of the IF or ELSE IF cases evaluate to true, then the default case in the ELSE block processes the record.

> **Note**
>
> If a record evaluates to true for more than one case, the record is only processed by the first IF/ELSE block that tests it. Records are never processed by more than one IF/ELSE block in a GROUP command.

```
OPEN Ap_Trans

COMMENT set initial values for running totals END
ASSIGN v_running_total_hi = 0.00
ASSIGN v_running_total_med = 0.00
ASSIGN v_running_total_low = 0.00

COMMENT use GROUP IF to run different ASSIGN and EXTRACT commands depending on
invoice amount END
GROUP IF Amount >= 1000
  ASSIGN v_running_total_hi = v_running_total_hi + Amount
  EXTRACT Invoice_Number, Amount, Date, v_running_total_hi AS "Running total"
TO results_hi
ELSE IF Amount >= 100
  ASSIGN v_running_total_med = v_running_total_med + Amount
  EXTRACT Invoice_Number, Amount, Date, v_running_total_med AS "Running total"
TO results_med
ELSE
  ASSIGN v_running_total_low = v_running_total_low + Amount
  EXTRACT Invoice_Number, Amount, Date, v_running_total_low AS "Running total"
TO results_low
END
```

When the script runs, the GROUP command tests the invoice amount for each record. Depending on the amount, the record is used to update one of three running totals (low, medium, high) and three result tables are produced.

LOOP inside a GROUP

When using GROUP to process the records in a table, you can use a LOOP command to execute a series of commands on a single record multiple times. LOOP is a second iteration that happens inside the iteration of GROUP, and it runs until a test condition that you specify evaluates to false.

Using LOOP to split a field

You have the following table containing invoice data and you need to isolate specific information for invoice amounts per department. One invoice may be related to more than one department, and department codes are stored in comma-delimited format in the table:

Vendor_Number	Invoice_Number	Date	Amount	Dept_Code
11663	5981807	2000-11-17	618.30	CCD,RDR
13808	2275301	2000-11-17	6705.12	CCD

Vendor_Number	Invoice_Number	Date	Amount	Dept_Code
12433	6585673	2000-11-17	7955.46	CCD,LMO,RDR

To extract the invoice amounts per department, you:

1. Use a GROUP command to process the table record by record.
2. Calculate the number of departments (*n*) associated with each record.
3. Use the LOOP command to iterate *n* times over the record to extract data for each department associated with the record.

> **Note**
>
> You must increment the v_counter variable inside LOOP. If you do not, the WHILE test always evaluates to true and the script enters an infinite loop. You can include a SET LOOP command in your scripts to guard against infinite loops. For more information, see "SET command" on page 1960.

```
COMMENT
use GROUP to count commas in each department code field as a way of identi-
fying how many departments are associated with the record
"LOOP" over each record for each code in the field, with each iteration of the
loop extracting the record with a single code to the result1 table
END
GROUP
  v_department_count = OCCURS(Dept_Code,',')
  v_counter = 0
  LOOP WHILE v_counter <= v_department_count
    v_dept = SPLIT(Dept_Code, ',', (v_counter + 1))
    EXTRACT FIELDS Invoice_Number, Amount, v_dept AS "Department" TO result1
    v_counter = v_counter + 1
  END
END
```

When the script runs, the commands inside the GROUP block are processed against each record in the table, from top to bottom. For each record, the LOOP command iterates over the record once per department code in the comma-delimited list and then extracts a record. If we could walk through GROUP and LOOP as they run, this is how it would look:

First iteration of GROUP: 2 iterations of LOOP

Vendor_Number	Invoice_Number	Date	Amount	Dept_Code
11663	5981807	2000-11-17	618.30	CCD,RDR

Vendor_Number	Invoice_Number	Date	Amount	Dept_Code
13808	2275301	2000-11-17	6705.12	CCD
12433	6585673	2000-11-17	7955.46	CCD,LMO,RDR

For the first record in the table, the value of v_department_count is 1, so LOOP iterates twice:

1. For the first iteration of the LOOP:
   - v_counter = 0
   - v_depart = CCD

   The following record is extracted and the value of v_counter is incremented to 1, therefore LOOP iterates again:

5981807	618.30	CCD

2. For the second iteration of LOOP:
   - v_counter = 1
   - v_depart = RDR

   The following record is extracted and the value of v_counter is incremented to 2, therefore LOOP does not iterate again and GROUP proceeds to the next record:

5981807	618.30	RDR

Second iteration of GROUP: 1 iteration of LOOP

Vendor_Number	Invoice_Number	Date	Amount	Dept_Code
11663	5981807	2000-11-17	618.30	CCD,RDR
13808	2275301	2000-11-17	6705.12	CCD
12433	6585673	2000-11-17	7955.46	CCD,LMO,RDR

For the second record in the table, the value of v_department_count is 0, so LOOP iterates once:

- v_counter = 0
- v_depart = CCD

The following record is extracted and the value of v_counter is incremented to 1, therefore LOOP does not iterate again and GROUP proceeds to the next record:

2275301	6705.12	CCD

Third iteration of GROUP: 3 iterations of LOOP

Vendor_Number	Invoice_Number	Date	Amount	Dept_Code
11663	5981807	2000-11-17	618.30	CCD,RDR
13808	2275301	2000-11-17	6705.12	CCD
12433	6585673	2000-11-17	7955.46	CCD,LMO,RDR

For the third record in the table, the value of v_department_count is 2, so LOOP iterates three times:

1. For the first iteration of LOOP:
   - v_counter = 0
   - v_depart = CCD

   The following record is extracted and the value of v_counter is incremented to 1, therefore LOOP iterates again:

6585673	7955.46	CCD

2. For the second iteration of LOOP:
   - v_counter = 1
   - v_depart = LMO

   The following record is extracted and the value of v_counter is incremented to 2, therefore LOOP iterates again:

6585673	7955.46	LMO

3. For the third iteration of LOOP:
   - v_counter = 2
   - v_depart = RDR

   The following record is extracted and the value of v_counter is incremented to 3, therefore LOOP does not iterate again and GROUP reaches the end of the table:

6585673	7955.46	RDR

Final results table

After GROUP has processed each record in the table, and LOOP has iterated though all the department codes, you have the following results table:

Invoice_Number	Amount	Department
5981807	618.30	CCD
5981807	618.30	RDR

Invoice_Number	Amount	Department
2275301	6705.12	CCD
6585673	7955.46	CCD
6585673	7955.46	LMO
6585673	7955.46	RDR

How to use functions

Even if you are a new or basic user of Analytics, who doesn't write scripts, functions can give you some simple and effective ways to work with data.

Functions can be used in a number of different areas of Analytics – in filters, in computed fields, in expressions within commands, and in scripts. Don't worry if you aren't familiar with some of these areas. You will be introduced to them as part of the instruction in how to use functions.

Short tutorials

Taken together, the short tutorials in this section provide a solid introduction to Analytics functions, and show you how to use them to achieve some useful things.

You can work through the tutorials in sequence, or do only the tutorial that meets your immediate need:

I would like to . . .	Tutorial
Understand what a function is	"What is a function?" on page 1426 ◦ Conceptual understanding of a function ◦ The three basic parts of a function
Learn an easy way to familiarize with any Analytics function	"Familiarizing with different functions" on page 1430 ◦ How to quickly and easily familiarize with any Analytics function ◦ Common errors when using functions
Learn how to filter or search data using functions	"Using functions to create filters" on page 1434 ◦ Brief overview of filters ◦ Using functions to: • filter by date • filter by multiple values • filter by fuzzy values
Learn how to clean or prepare data using functions	"Using functions to clean data" on page 1438 ◦ Brief overview of cleaning data ◦ Using functions to: • remove blank spaces • remove unwanted characters
Learn how to increase efficiency and power by combining functions	"Cleaning and filtering data at the same time" on page 1442 ◦ Introduction to nested functions

Advanced use of functions

Once you understand the basics, additional tutorials explain the various ways you can put functions to use throughout Analytics.

See "Advanced use of functions" on page 1446.

What is a function?

You can think of an Analytics function as a small tool that performs a specific, useful task. For example, you can use a function to standardize upper and lowercase in a piece of inconsistently formatted text:

Function and input	Output
`PROPER("john SMITH")`	`John Smith`

Another way to think of a function is as an "opaque box". The input goes in one side, something happens inside to transform it, and the output comes out the other side.

The three basic parts of a function

The example `PROPER("john SMITH")` demonstrates the three basic parts of any function:

- **The function name** – in this case, `PROPER`
- **A set of parentheses** – an opening parenthesis `(` and a closing parenthesis `)`
- **The function input** – everything inside the parentheses: in this case, `"john SMITH"`, including the quotation marks

> **Note**
>
> Throughout Analytics documentation, function names are presented in uppercase, which is simply a formatting convention. Analytics does not require that functions are entered in uppercase.

Input and output

A slightly more technical description of a function is that it's a calculation or operation performed by a computer that accepts an **input** and returns an **output**.

In the example above, the input is `"john SMITH"`, the output is `John Smith`, and the operation performed by the `PROPER( )` function is to transform all words to proper case (initial capital letter followed by lowercase).

Functions never alter source data

A function never alters the source data used as input. It uses the input data to calculate the output results, and the results are stored in your computer's memory so you can make use of them.

In the example above, the physical data `john SMITH` remains unchanged on your computer. You can think of the output `John Smith` as "virtual data", which exists in memory, and can be used in subsequent operations.

We revisit this point in the discussion about using functions to created computed fields.

The scope of functions

The calculation or operation performed by any particular function is narrow in scope. The `PROPER( )` function does nothing more than convert the case of text. But as you'll see throughout this set of tutorials, even though the scope of each function is small, functions are powerful, and crucial to data analysis in Analytics.

The difference between a function and a command

Analytics functions and commands both perform calculations or operations on data, but functions are narrow in scope, whereas commands are often broad in scope. For example:

- **narrow scope** – the `PROPER( )` function converts the case of text
- **broad scope** – the `SUMMARIZE` command groups all the records in a table

Functions can provide input for commands. For example, you might use the `PROPER( )` function to convert the case of a Name field, and then use the `SUMMARIZE` command to group records by the now-standardized Name field.

The reverse is not true. Commands cannot be used as inputs for functions.

Some more examples

You can use a function to remove leading and trailing spaces from text, to remove hyphens from an ID number, or to find records with dates in a particular date range.

Example

Here are examples of what three different functions can do:

- the ALLTRIM( ) function
- the EXCLUDE( ) function
- the BETWEEN( ) function

Function and input	Output
`ALLTRIM(" Chicago ")`	`Chicago` Letters only, no leading or trailing spaces.
`EXCLUDE("VT-123-45", "-")`	`VT12345` Letters and numbers only, no hyphens.
`BETWEEN(`20170701`, `20170101`, `20171231`)` In the function input: ◦ the first date is the one being tested ◦ the second date is the start date of the range ◦ the third date is the end date of the range	`T` Returns `T` for True because 01 July 2017 is between 01 January and 31 December 2017.

Literal values versus fields as function input

The examples above use actual, or literal, input values so that you can see exactly what each function does. In Analytics, you typically use a field, or a variable, as the primary input for a function.

A field as function input is shown below. Variables as function input are explained in a subsequent tutorial.

Example

Here is the BETWEEN( ) example from above, but now with a date field as input rather than a literal date value.

```
BETWEEN(Invoice_Date, `20170101`, `20171231`)
```

The function returns `T` for True for every date in the Invoice_Date field that falls in the year 2017, and `F` for False for dates in other years.

> **Note**
> The two boundary values you specify for the BETWEEN( ) function are inclusive. Small details like this one are included in the Help topic for each function.

Where to next?

Learn how to quickly and easily familiarize with any Analytics function: "Familiarizing with different functions" on the next page

Familiarizing with different functions

The easiest way to learn what any function does is to play around with it in the Analytics command line. You can try now, using the examples from the previous tutorial:

`PROPER("john SMITH")`	`EXCLUDE("VT-123-45", "-")`
`ALLTRIM(" Chicago ")`	`BETWEEN(`20170701`, `20170101`, `20171231`)`

1. Open any Analytics project in Analytics.
2. Make sure the command line is open.

If it is not open, on the main menu, select **Window > Command Line**.

3. Copy and paste one of the function examples above into the command line.
4. Type `DISPLAY` and a space before the pasted example and press Enter.

The function output, also known as the **return value**, appears in the Analytics display screen.

> **Note**
> You can't do anything with the function output in the display screen. It's simply a read-only output that allows you to see what a particular function with a particular input returns.

5. To temporarily save the function output, pin the display screen.

> **Tip**
> You can click the linked function in the display screen to quickly reload it into the command line.

Now try changing some input values. . .

Enter or reload the function into the command line, and change one or more input values to see how the output changes.

> **Tip**
> If you start doing a lot of experimenting with functions in the command line, you can just type `disp` instead of `DISPLAY`.

EXCLUDE( ) example

In the EXCLUDE( ) example, if you add `VT` to the characters to exclude, you should have output that includes only numbers.

```
EXCLUDE("VT-123-45", "VT-")
```

BETWEEN( ) example

In the BETWEEN( ) example, what happens when you change the literal invoice date to 01 July 2016?

The invoice date is the first of the three input values.

```
BETWEEN(`20160701`, `20170101`, `20171231`)
```

You should find that the output has now changed from `T` to `F`, or from True to False, because 01 July 2016 is not between 01 January and 31 December 2017.

What other functions can I play around with?

You can use the `DISPLAY` method in the command line to experiment with any Analytics function. Analytics has over 130 functions, serving a wide range of purposes.

> **Note**
>
> Using `DISPLAY` with a function in the command line is only for testing or learning purposes. You do not use `DISPLAY` with functions anywhere else in Analytics.

To find other functions to try:

1. Hover over **Functions** in the menu at the top of this Help topic, and select a function category.
2. On the category page, click the name of a function that interests you.
3. Copy and paste an example from the individual function page into the command line.

   Make sure to choose an example that use literal values, not fields or generic placeholders.
4. Type `DISPLAY` and a space before the pasted example and verify that it returns the same output value as shown in the Help topic.
5. Change input values to create different output values and learn more about how the function works.

> **Tip**
>
> Consult the function Help topic if you need help understanding some of the function inputs.

I got an error message

If you get an error message while experimenting with a function, the most likely explanation is that you made a small error when entering the function in the command line.

Some of the error messages may sound serious, but often the error is minor and can be easily fixed if you know what it is.

The rules governing the way functions must be entered, in the command line and elsewhere in Analytics, are strict:

Function names	Function names must be spelled correctly.
Parentheses	The opening parenthesis must immediately follow the function name with no intervening space: `PROPER("john SMITH")`, not `PROPER ("john SMITH")` Function parentheses must open and close. When we come to nested functions, keeping tracking of opening and closing parentheses

	can be a little more challenging.
Text	Literal text values must be enclosed in "quotation marks". Quotation marks must be "straight". “Curly or slanted” quotation marks, which can occur when you copy and paste from some sources, cause an error.
Dates	Literal date values must be enclosed in `backquotes`, and use a YYYYMMDD format (or YYMMDD).
Fields and numbers	Field names and numbers use no punctuation: ◦ Invoice_Date ◦ 1000.00
DISPLAY	You must preface a function with `DISPLAY` in the command line (and nowhere else).
Data type	Functions require that input values are a specific data type. Some functions accept more than one data type, whereas others accept only the Character data type, or the Numeric data type, and so on. The function Help topics tell you what data type or types are valid for each function. For more information about data types, see "Data types in Analytics" on page 739.

> **Tip**
>
> Minor errors in function syntax can be difficult to spot. Check your syntax carefully if an error recurs.
>
> Function Help topics provide comprehensive information about the required syntax for each function.

Where to next?

Learn how to use functions to filter data in a variety of different ways: "Using functions to create filters" on the next page

Using functions to create filters

You can use an Analytics function to create a filter. Different functions allow you to create different kinds of filters, depending on your particular need.

What's the purpose of a filter?

Filters are a basic and critical component of data analysis. Filters allow you to exclude records that you are not currently interested in, and include only the records that you want to examine.

If you have a table with a million records, and you want to examine only a small portion of them, you need a filter of some kind.

How a filter works

A filter is an expression that evaluates the records in a table and returns a value of "T" (True) or "F" (False) for each record.

Example

You want to examine only those amounts in an accounts receivable table that you consider material. Your threshold for materiality is $1000.00, so you create the following filter:

```
Invoice_Amount >= 1000.00
```

This filter returns True for amounts greater than or equal to $1000.00, and False for amounts less than $1000.00. Records that evaluate to True are included by the filter, and records that evaluate to False are excluded.

Excluded records are hidden from view while the filter is applied, and they are excluded from any Analytics commands you run on the table.

More sophisticated filters

You can use operators such as the Greater Than `>` and Less Than `<` signs to create simple filters, but using functions you can create more sophisticated filters.

Filter by date

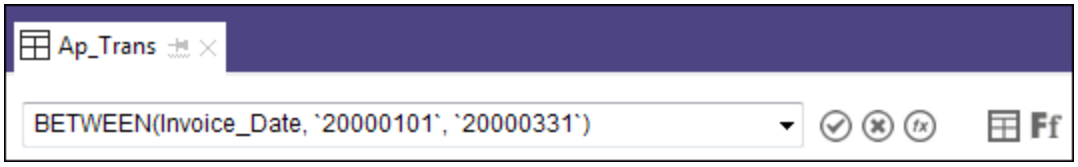
We can use a version of the BETWEEN( ) example from the previous tutorials to create a filter that includes only invoices from the first quarter.

1. In Analytics, open `Sample Project.ACL`, and open the **Ap_Trans** table (`Tables\Accounts_Payable\Ap_Trans`).

   If `Sample Project.ACL` is not available, open any table with a date field. To work with this example, the field must use the Datetime data type.

2. Copy and paste this version of the BETWEEN( ) example into the Filter text box at the top of the View tab, and press Enter:

   ```
   BETWEEN(Invoice_Date, `20000101`, `20000331`)
   ```

   Result: The table is filtered to display only invoices from the first quarter of the year.

   If you not using the **Ap_Trans** table, update the field name and boundary dates in the BETWEEN( ) function to match your data.

   The field name must be the physical field name, not the display name (alternate column title). Right-click the header for the date field and select **Properties** to see both the physical and the display field names.

   > **Note**
   > Do not use `DISPLAY` in the Filter text box.

3. Try changing one or both boundary dates to create a different date filter.

   When entering a literal date, you must use the format `` `YYYYMMDD` ``. If you are using the **Ap_Trans** table, all dates are in the year 2000.

> **Tip**
> You can also use BETWEEN( ) to filter numeric or text data. Enclose text inputs in `"quotation marks"`. Do not enclose field names or numeric inputs in any punctuation: `Invoice_Amount`, `1000.00`

Filter by multiple values

Now we'll use the MATCH( ) function to filter by multiple values simultaneously.

1. Copy and paste the MATCH( ) function with these inputs into the Filter text box, and press Enter:

```
MATCH(Vendor.Vendor_City, "Austin", "Chicago", "Salt Lake City")
```

Result: The filter on the **Ap_Trans** table updates to display only invoices from vendors in the three specified cities.

> **Note**
> The Vendor_City field is in the **Vendor** table, which is related to the **Ap_Trans** table in `Sample Project.ACL`. To reference related fields in functions, you use *table name.field name* syntax.
> To reference fields in the open table, you use just *field name*.

2. Try changing the field, and the three terms to match against, to create different sorts of filters.

> **Note**
> Search terms in the MATCH( ) function are case-sensitive.

Filter by fuzzy values

You should already be starting to see the power and usefulness of functions. This third filter uses the ISFUZZYDUP( ) function, which lets you filter by identical and nearly identical values.

Trying to perform a similar operation manually on a large table would be extremely time-consuming, if not impossible.

1. Copy and paste the ISFUZZYDUP( ) function with these inputs into the Filter text box, and press Enter:

```
ISFUZZYDUP(Vendor.Vendor_Name, "Miller Co", 4)
```

Result: The filter on the **Ap_Trans** table updates to display only invoices from vendors with names that are identical or nearly identical to "Miller Co". You should see two records for the vendor "Muller Corp."

2. Increase the degree of fuzziness from `4` to `8` and press Enter.

   An additional record for "MGMT Mfg." should now be included by the filter.

3. Click **Remove Filter**  , review the vendor names, and change "Miller Co" to something close to, but not exactly matching, one of the other vendor names.

   Experiment with different fuzziness settings. Valid settings are 1 to 10, inclusive.

The quick search in Analytics is really a filter

The quick search feature in Analytics is really a filter that uses the FIND( ) function.

1. In the **Ap_Trans** table, click **Remove Filter**  .
2. Type the search term `931` in the Filter text box, and press Enter.

   Result: The table is searched for the characters **931** and two records are included in the filtered results:

   - one has an invoice number ending with 931
   - one has a product number ending with 931

Note that in the Filter text box your search term has been converted to the FIND( ) function with the input "931": `FIND("931")`

The right tool for the job

FIND( ) is another Analytics function that you can use for filtering data, or searching for specific items. It has the benefit of letting you search across all fields in a table.

But as you have already learned, there are other functions that give you additional powerful and flexible ways of filtering and searching data.

As you become more familiar with the entire set of Analytics functions, you'll discover that the function you choose depends on what you're currently trying to achieve.

Key point

You can use functions to create filters throughout Analytics, including in scripts. Filters created with functions are a fundamental building block of data analysis in Analytics.

Where to next?

Learn how to use functions to perform data cleansing or data preparation tasks: "Using functions to clean data" on the next page

Using functions to clean data

You can use an Analytics function to clean data. Different functions allow you to perform different types of data cleansing, depending on your particular need.

Why do I need to clean data?

Frequently, data imported into Analytics is not clean – meaning perfectly formatted and standardized. Analytics commands do not work, or give inaccurate results, if you input badly formatted or non-standard data.

How cleaning data works

When you clean data you are not cleaning or modifying the actual source data. The source data always remains read-only.

Instead, you input the source data to a function that processes it and outputs appropriately formatted and standardized "virtual data". You then input the cleansed virtual data to an Analytics command, rather than the original source data.

Key point

Using one or more functions, you can perform a wide range of data cleansing or **data preparation** tasks that allow you to work effectively and accurately even if source data is inconsistent. Data preparation is a fundamental preliminary task for much data analysis.

Removing blank spaces

Inconsistent blank spaces in data are a common cause of inaccurate results. You can use the ALLTRIM( ) function to remove leading and trailing blank spaces and ensure accurate results.

Example: Blank spaces

You want to sort a vendor table by city, but leading spaces in some of the city names are causing inaccurate sorting.

Vendor_City sorted
[ ] [ ] [ ] [ ] Salt Lake City
[ ] [ ] Chicago
Ann Arbor
Austin
Englewood
[ ] = blank space

You can use the ALLTRIM( ) function to get rid of the leading spaces and ensure accurate sorting:

```
ALLTRIM(Vendor_City)
```

ALLTRIM(Vendor_City) sorted
Ann Arbor
Austin
Chicago
Englewood
Salt Lake City

> **Note**
> To apply the ALLTRIM( ) function to the Vendor_City field, you create a computed field that uses ALLTRIM( ). Computed fields are discussed in a subsequent tutorial.

Removing unwanted characters

Inconsistent characters, or non-critical characters, can often impede data analysis. You can use different functions to include only certain characters, or to exclude certain characters or strings of characters, prior to processing data with an Analytics command.

Trying things yourself

You can copy and paste any of the function examples below into the command line to verify the return value, or to experiment with different inputs.

In the command line, you must preface the example with `DISPLAY` and a space. Experimenting in the command line is explained in previous tutorials.

Example: Unwanted characters

You want to perform a duplicates test on a table, but inconsistent formatting of data is causing inaccurate results.

For example, running the duplicates command on an inconsistently formatted Phone Number field does not report these two phone numbers as duplicates, although they clearly are duplicates:

- (604) 555-1212
- Tel. No: 604-555-1212

To ensure all duplicates are found, you can use functions to standardize the data before performing the duplicates operation.

Task	Function example

Standardize telephone numbers	`INCLUDE("(604) 555-1212", "1234567890")`
	
	Returns **6045551212**
	
	`INCLUDE("Tel. No: 604-555-1212", "1234567890")`
	
	Returns **6045551212**
	
	The INCLUDE( ) function includes only the specified characters in the output - in this case, only the numbers 0 to 9
	
	Tip
	Use INCLUDE( ) if the set of characters you want to include is small, and the set you want to exclude is large.

Standardize addresses	`EXCLUDE("#1550-980 Howe St.", "#.")` Returns **1550-980 Howe St**

Task	Function example
	`EXCLUDE("1550-980 Howe St", "#.")` Returns **1550-980 Howe St** The EXCLUDE( ) function excludes the specified characters from the output - in this case, the hash sign (#) and the period (.) **Tip** Use EXCLUDE( ) if the set of characters you want to exclude is small, and the set you want to include is large.

Standardize addresses and remove street abbreviations	`OMIT("#1550-980 Howe St.", " Street, St.,#")`
	
	Returns **1550-980 Howe**
	
	`OMIT("1550-980 Howe Street", " Street, St.,#")`
	
	Returns **1550-980 Howe**
	
	The OMIT( ) function excludes the specified characters and strings of characters from the output - in this case, the hash sign (#), and the inconsistently formatted **St.** and **Street**
	
	Tip
	Use OMIT( ) if you want to exclude specific strings of characters, but not the individual characters that make up the string.
	For example, exclude **Street** when it occurs as a unit, but not **S**, **t**, **r**, **e**, or **t** when they occur in other words.

Where to next?

Learn how to use functions to perform multiple tasks simultaneously: "Cleaning and filtering data at the same time" on the next page

Cleaning and filtering data at the same time

The two previous tutorials showed you how to use functions to filter data and to clean data. Now we'll look at how you can **nest** functions to perform both tasks simultaneously.

Nested functions

You can nest one function inside another function to achieve results that you couldn't achieve with either function alone.

Basic structure

Here is the basic structure of a nested function with one level of nesting:

```
FUNCTION_2( FUNCTION_1(function_1 input) , function_2 input)
```

You can see that `FUNCTION_1( )` is completely contained inside `FUNCTION_2( )`.

Order of evaluation

Nested functions are evaluated starting with the innermost function and working outward to the outermost function. So, in the generic example above:

1. `FUNCTION_1(function_1 input)` is evaluated first.
2. The output of `FUNCTION_1( )` becomes one of the inputs for `FUNCTION_2( )`.
3. `FUNCTION_2( )` is evaluated second.

Think about inputs and outputs

Generally speaking, you can nest any Analytics function inside another function, and if required build multiple levels of nesting.

However, keep in mind that the output of a function must meet the input requirements of the function containing it. For example, if a function requires a date input, the function that it contains must output a value with a datetime data type.

Key point

Nesting functions is a powerful and flexible capability that can allow you to achieve a great range of useful results. You can perform multiple transformations of source data simultaneously in preparation for inputting the data to a command.

Standardize case, and filter by multiple values

In a previous tutorial we used the MATCH( ) function to filter by multiple values. MATCH( ) is case sensitive, so if the case of the input values varies, the filter produces inaccurate results.

You can produce accurate results by nesting the UPPER( ) function inside the MATCH( ) function.

Example

You want to use the Vendor_City field to filter records in a table, but the city names have been entered inconsistently. Some have an initial capital ("Austin"), and some are entirely uppercase ("AUSTIN").

You can nest the UPPER( ) function inside the MATCH( ) function to:

1. transform all values in the Vendor_City field to uppercase
2. filter the records by city

   Note that you have to adjust your filter terms to be all uppercase so that they match the uppercase values output by the UPPER( ) function.

   ```
   MATCH( UPPER(Vendor_City) , "AUSTIN", "CHICAGO")
   ```

The table below illustrates the difference between using the MATCH( ) function alone, and using the nested function.

With MATCH( ) alone, the filter is too restrictive and excludes records that should be included.

Returned by:	Returned by:
MATCH(Vendor_City, "Austin", "Chicago")	MATCH(UPPER(Vendor_City), "AUSTIN", "CHICAGO")
Austin	Austin
Chicago	Chicago

Returned by:	Returned by:
MATCH(Vendor_City, "Austin", "Chicago")	MATCH(UPPER(Vendor_City), "AUSTIN", "CHICAGO")
	AUSTIN
	CHICAGO

Tip

Instead of using a nested function, you could add variations to the filter terms: `MATCH(Vendor_City, "Austin", "AUSTIN", "Chicago", "CHICAGO")`. However, with additional filter terms this approach quickly becomes labor-intensive, and would fail to capture values with typos, such as "AUstin". Nesting UPPER( ) is the better approach.

Note

To apply the MATCH( ) or UPPER( ) functions to the Vendor_City field, you create a computed field that uses the function. Computed fields are discussed in a subsequent tutorial.

Standardize case, remove leading spaces, and filter by multiple values

You aren't limited to just one level of nesting. You can create multiple levels of nesting, based on your requirements.

Keep in mind:

- Nested functions are evaluate from the innermost function to the outermost function.
- The output of a function must meet the input requirements for the function containing it.

Example

In a second situation, the data in the Vendor_City field is even less consistent. Not only is case inconsistent, but some values are preceded by one or more blank spaces and some are not.

You can nest the UPPER( ) function inside the ALLTRIM( ) function, and the ALLTRIM ( ) function inside the MATCH( ) function to:

1. transform all values in the Vendor_City field to uppercase
2. remove all leading blank spaces

3. filter the records by city

```
MATCH( ALLTRIM( UPPER(Vendor_City) ), "AUSTIN", "CHICAGO")
```

> **Tip**
>
> It's easy to lose track of opening and closing parentheses when building nested functions. Missing or unmatched parentheses are a common cause of function errors.
>
> The number of opening parentheses `(` must always equal the number of closing parentheses `)`. In the example above, there are three opening parentheses and three closing parentheses.

The table below illustrates the difference between using the MATCH( ) function alone, and using the nested function.

With MATCH( ) alone, the filter is too restrictive and excludes records that should be included.

Returned by:	Returned by:
MATCH(Vendor_City, "Austin", "Chicago")	MATCH(ALLTRIM(UPPER(Vendor_City)), "AUSTIN", "CHICAGO")
Austin	Austin
Chicago	Chicago
	AUSTIN
	CHICAGO
	[ ] Austin
	[ ] [ ] [ ] Chicago
	[ ] [ ] AUSTIN
	[ ] CHICAGO
	[ ] = blank space

Where to next?

If you have completed all the tutorials in "How to use functions" on page 1424, you are ready to move on to "Advanced use of functions" on the next page.

The advanced tutorials teach you how to use functions with core Analytics features.

Advanced use of functions

Once you understand the basics of how Analytics functions work, you're ready to learn how to use them throughout Analytics.

The tutorials in this section show you how to use functions in computed fields, embedded in commands, and in scripts. The true usefulness of functions will become apparent as you start to use them with these core Analytics features.

Don't worry if you aren't familiar with computed fields or scripts. You will be introduced to them in the course of the tutorials.

> **Note**
>
> Using functions in these other situations is a little more involved than the usage presented in previous tutorials. However, the functions themselves **behave in exactly the same way**.
>
> Remember that you can quickly and easily test any function in the Analytics command line to see what it does: "Familiarizing with different functions" on page 1430.

Short tutorials

The tutorials are designed to be completed in sequence:

Learn how to . . .	Tutorial
Apply a function to all the values in a field	"Using a function to group records by month" on page 1448 ◦ Applying a function to all the values in a field: • by creating a computed field • by embedding a function in an Analytics command
Use variables with functions	"Using variables with a function to allow user input" on page 1455 ◦ Brief overview of variables ◦ Using variables as inputs for a function
Use functions in a script	"Putting it all together: using functions in a script" on page 1460 ◦ An interactive script that makes use of a number of functions

Basic use of functions

For an introduction to the basic use of functions, see "How to use functions" on page 1424.

Using a function to group records by month

To keep things simple in previous tutorials, a number of the examples of Analytics functions use literal input values - "john SMITH", "VT-123-45", and so on. But how do you apply a function to the entire set of values in a field in an Analytics table?

Applying a function to an entire field can help you perform useful tasks such as grouping the records in a table by month.

Computed field

One way to apply a function to all the values in a field is to create **a computed field**. A computed field is one that you create, often based on an actual physical field, but composed entirely of values computed by Analytics.

Similar to the output of a function, you can think of a computed field as virtual data, calculated by Analytics, that exists in memory. Once calculated, this virtual data can be used in subsequent operations.

Create a computed field to help group records by month

We can create a computed field called **Month** that uses the MONTH( ) function to extract the month portion from every date in an invoice date field. We can then group the records in the table by month.

Create the computed field

1. In Analytics, open `Sample Project.ACL`, and open the **Ap_Trans** table (`Tables\Accounts_Payable\Ap_Trans`).

   If `Sample Project.ACL` is not available, open any table with a date field. To work with this example, the field must use the Datetime data type.

2. Perform the following steps to create the **Month** computed field:
   a. At the top of the table view, click **Edit Table Layout**.
   b. In the **Table Layout** dialog box, click **Add a New Expression**.
   c. In the **Name** field type `Month`, and in the **Default Value** field copy and paste this version of the MONTH( ) function:

```
MONTH(Invoice_Date)
```

If you not using the **Ap_Trans** table, update the field name to match your data.

d. Click **Accept Entry**, and close the **Table Layout** dialog box.
e. In the table view, right-click the header of the **Invoice Date** column, select **Add Columns**, under **Available Fields** double-click **Month**, and click **OK**.

Result: The **Month** computed field is added to the view. It contains the month portion of each date in the **Invoice Date** column, displayed as a number from 1 to 12.

f. Click **Save the Open Project** to save your changes.

Group the records by month

Now that you've created the **Month** computed field, you can use it to group the records in the **Ap_Trans** table by month.

1. From the main menu, select **Analyze > Summarize**.
2. From the **Summarize On** list, select the **Month** field.
3. From the **Subtotal Fields** list, select the **Invoice_Amount** field.
4. Click the **Output** tab, select **File**, type `Ap_Trans_grouped` in the **Name** field, and click **OK**.

   Result: Analytics outputs the new table, which groups the records from the **Ap_Trans** table by month. For each month, there is an invoice amount subtotal, and a count of the number of records that occur in the month.

Month	Invoice Amount	Count
1	85,670.22	12
2	4,496.56	6
3	2,941.80	5
4	467.40	1
5	8,272.57	5
6	1,582.86	2
7	3,101.98	4
8	21,146.96	2
9	32,577.32	20
10	41,595.89	19

Month	Invoice Amount	Count
11	70,779.26	19
12	6,008.51	7

Suggested activity: display the names of the months

You can use the CMOY( ) function to create a second computed field if you want to display the names of the months. CMOY is an abbreviation for "Character Month of Year".

1. In the **Ap_Trans** table, follow the same steps you used to create the **Month** computed field and add it to the table view, but with these differences:
   - In the **Name** field type `Month_2`.
   - In the **Default Value** field copy and paste this version of the CMOY( ) function:

     ```
     CMOY(Invoice_Date, 9)
     ```

   Result: The **Month_2** computed field is added to the view with the name of each month.
2. Follow the same steps you used to group the records from the **Ap_Trans** table by month, but with these differences:
   - In the **Other Fields** list, select **Month_2**.
   - On the **Output** tab, specify the output file name `Ap_Trans_grouped_2`.

   Result: Analytics outputs the new table, which groups the records from the **Ap_Trans** table by month, but now the month names are included.

Month	Invoice Amount	Count	Month_2
1	85,670.22	12	January
2	4,496.56	6	February
3	2,941.80	5	March
4	467.40	1	April
5	8,272.57	5	May
6	1,582.86	2	June
7	3,101.98	4	July

Month	Invoice Amount	Count	Month_2
8	21,146.96	2	August
9	32,577.32	20	September
10	41,595.89	19	October
11	70,779.26	19	November
12	6,008.51	7	December

Skip creating the computed field

In many cases, creating a computed field as a way of applying a function to multiple values is useful. However, you can achieve the same result and streamline your work in Analytics by embedding functions directly inside Analytics commands.

Embed a function to help group records by month

We'll use the same example as above, but without creating the computed fields. Instead, we'll embed the functions directly in the summarize command.

1. Open the **Ap_Trans** table.
2. From the main menu, select **Analyze > Summarize**.

Embed the MONTH( ) function

1. Click **Summarize On**, and then click **Expr**.
2. In the **Expression Builder**, double-click **MONTH(date/datetime)** in the **Functions** list.

   > **Tip**
   > To make it easier to locate the MONTH( ) function, select **Date & Time** from the drop-down filter at the top of the **Functions** list.

3. In the **Expression** text box, select **date/datetime** and double-click **Invoice_Date** in the **Available Fields** list.

   You should have `MONTH( Invoice_Date )` in the **Expression** text box.

   > **Note**
   > The expression should look familiar. It's the same as the computed field from the previous example, only now it's embedded in the summarize command.

4. Click **OK** to exit the **Expression Builder**, and click **OK** to exit the **Select Fields** dialog box.

Embed the CMOY( ) function

1. Click **Other Fields**, and then click **Expr**.
2. In the **Expression Builder**, double-click **CMOY(date/datetime , length)** in the **Functions** list.
3. In the **Expression** text box, replace **date/datetime** with **Invoice_Date**, and **length** with `9`.

   You should have `CMOY( Invoice_Date , 9)` in the **Expression** text box.
4. Click **OK** to exit the **Expression Builder**, and click **OK** to exit the **Select Fields** dialog box.

Finalize the summarize operation

1. From the **Subtotal Fields** list, select the **Invoice_Amount** field.
2. Click the **Output** tab, select **File**, type `Ap_Trans_grouped_3` in the **Name** field, and click **OK**.

   Result: Analytics outputs the new table, which groups the records from the **Ap_Trans** table by month. You can see the two embedded functions.

MONTH(Invoice_Date)	Invoice Amount	Count	CMOY(Invoice_Date, 9)
1	85,670.22	12	January
2	4,496.56	6	February
3	2,941.80	5	March
4	467.40	1	April
5	8,272.57	5	May
6	1,582.86	2	June
7	3,101.98	4	July
8	21,146.96	2	August
9	32,577.32	20	September
10	41,595.89	19	October
11	70,779.26	19	November
12	6,008.51	7	December

Key point

You used two different methods to achieve exactly the same result:

- **Computed field** – Creating a computed field before using the field in a command is a more literal, step-by-step approach. It may be an appropriate approach if you want to use the

computed field for more than one purpose.
- **Embedded function** – Bypassing the creation of a computed field, and embedding a function in a command, is a more streamlined approach. It may be an appropriate approach in the context of scripting, producing more efficient scripts.

Suggested activity: group records by day of the week

If you want additional practice with using functions to create computed fields, or with embedding functions in commands, redo some or all of the activities above and substitute the DOW( ) and CDOW( ) functions.

DOW( ) and CDOW( ) are very similar to MONTH( ) and CMOY( ) except that they extract the day of the week from a date rather than the month of the year.

> **Tip**
> Using DOW( ) and CDOW( ), you could analyze how sales figures compare for different days of the week.

Another little trick for testing functions

You can use the `DISPLAY` method in the command line to get a sense of the output when you create a computed field, or embed a function in a command.

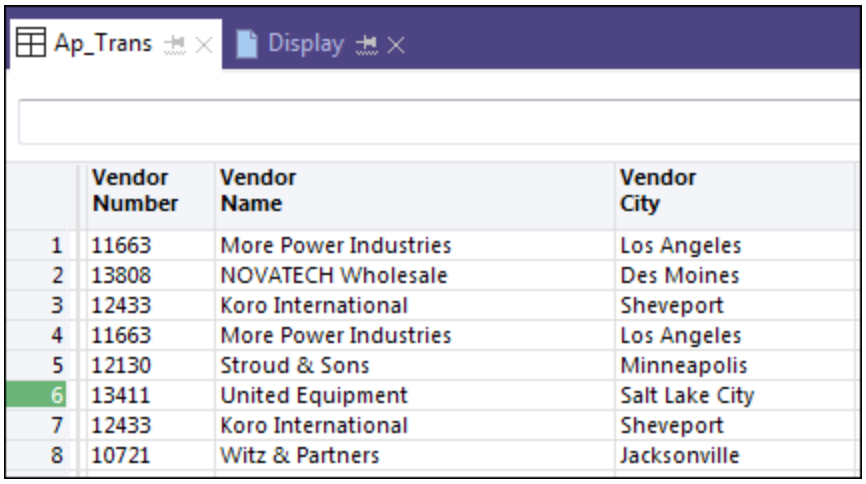
1. With the **Ap_Trans** table open, paste this version of the UPPER( ) function into the command line, type `DISPLAY` and a space before the pasted function, and press Enter.

   ```
   UPPER(Vendor.Vendor_Name)
   ```

   The UPPER( ) function converts all input text to uppercase.

   If the first record in the **Ap_Trans** table is selected, the function output is MORE POWER INDUSTRIES.
2. In the table, select record number 6.

   Click the record number to select the record. The selected record number is highlighted green.



	Vendor Number	Vendor Name	Vendor City
1	11663	More Power Industries	Los Angeles
2	13808	NOVATECH Wholesale	Des Moines
3	12433	Koro International	Sheveport
4	11663	More Power Industries	Los Angeles
5	12130	Stroud & Sons	Minneapolis
6	13411	United Equipment	Salt Lake City
7	12433	Koro International	Sheveport
8	10721	Witz & Partners	Jacksonville

3. Type or reload the function into the command line and press Enter.

   The function output is UNITED EQUIPMENT.

4. Select one or two other records and repeat the process.

   Key point: On a record-by-record basis you are seeing what a computed field, or an embedded function, would do to all the values in the Vendor.Vendor_Name field.

   You can use this testing method with any Analytics function that takes a field for input.

Where to next?

Learn how to use variables with a function to create interactivity: "Using variables with a function to allow user input" on the facing page

Using variables with a function to allow user input

So far, you've been shown how to use fields and literal values as inputs for functions. This approach is relatively straightforward and in many situations in Analytics it's all that you require.

In Analytics scripts, **variables** are typically used as inputs for functions, rather than fields or literal values.

In this tutorial, we'll look at using variables with a function. The tutorial concludes with a simple script that includes a function with variables as inputs. The variables allow a user to specify the actual input values interactively.

What are variables and how are they useful?

You can think of a variable as a named container in a computer's memory. You provide the name when you create the variable. The variable can temporarily, or more permanently, store whatever value a user selects or specifies.

Variables have two very useful qualities:

- **Flexibility** – Variables make a script much more flexible.

  For example, instead of requiring a specific field name, or set of dates, a script can use variables to allow users to select or specify whatever field name or dates they want.

- **Clarity** – Variables make a script easier to understand when you're reviewing it or updating it.

  It's much easier to understand a meaningfully named variable as a function input, than a piece of raw data. You can see the difference with the two examples of the BETWEEN( ) function below.

For more information, see "Variables" on page 1411.

BETWEEN( ) without variables

Consider the BETWEEN( ) example that we used to create a filter in an earlier tutorial:

```
BETWEEN(Invoice_Date, `20000101`, `20000331`)
```

The filter restricts the records in the Invoice_Date field to the first quarter of the year 2000.

This version of the BETWEEN( ) function is fine if we don't mind manually changing the field name, and the start and end dates, every time we want to use it in a different context.

But what if we want to include this filter in a script to be run by other users, against different data, and the other users don't understand how to update the function inputs?

Note, as well, that based solely on a visual inspection, there's no way of knowing conclusively the purpose of the raw data providing function inputs.

BETWEEN( ) with variables

Instead of specifying an actual field and literal date values as inputs for the BETWEEN( ) function, you can specify variables:

```
BETWEEN(v_date_field, v_start_date, v_end_date)
```

In conjunction with an interactive script, this version of the BETWEEN( ) function allows a user to pick any date field they want, and specify the two boundary dates.

Note, also, that just by looking at the function the purpose of each input is clear.

> **Note**
> By convention, scriptwriters preface their variable names with "v_" so that in a complex script it's easy to see what's a variable and what isn't.

Key point

By using variables, you can create a much broader and more flexible application of a function.

Test the BETWEEN( ) function with variables

You can test the BETWEEN( ) function in the Analytics command line to see exactly how the variables work.

Testing functions in the command line is fully explained in a previous tutorial: "Familiarizing with different functions" on page 1430.

Create the variables

1. In Analytics, at the bottom of the **Navigator**, click the **Variables** tab.

   The **Variables** tab displays all the variables that currently exist in an Analytics project, and the value each variable currently contains.

2. Create the three example variables by entering the following variable definitions in the command line, one at a time:
   - `v_date_field = `20170715``

     For this example, we'll just specify a single literal value for the `v_date_field` variable, rather than an actual field.

   - `v_start_date = `20170701``
   - `v_end_date = `20170731``

   In the **Variables** tab, you should see the three variables you just created, with the assigned values.

Test BETWEEN( )

1. Copy and paste the BETWEEN( ) example into the command line:

   ```
   BETWEEN(v_date_field, v_start_date, v_end_date)
   ```

2. Type `DISPLAY` and a space before the example, and press Enter.

   The result should be `T` for True, based on the values contained in the variables:

   15 July 2017 is between the specified start and end dates.

See the result of changing one of the variable values

1. Update the value in `v_start_date` by entering the following in the command line:

   ```
   v_start_date = `20170716`
   ```

   In the **Variables** tab, you should see the value in `v_start_date` has updated to 16 July 2017, which falls after the `v_date_field` value.

2. Re-run the BETWEEN( ) function in the command line.

   The result should be `F` for False, based on the values contained in the variables:

   15 July 2017 is not between the specified start and end dates.

Try out the BETWEEN( ) function in a script

The simple script below allows a user to apply a date filter to any Analytics table with a date field.

Don't worry if you can't understand all the script syntax. The main point is to see the BETWEEN ( ) function in action in a script.

A `COMMENT` before each piece of script syntax explains in simple terms what the syntax is doing. Functions are highlighted brown.

How to run the example script in Analytics

1. Open an Analytics project that contains one or more tables with date fields.

   `Sample Project.ACL` has several tables with date fields.

2. Create a new, empty script:
   a. In the **Navigator**, right-click a folder or the top-level project entry and select **New > Script**.
   b. Copy and paste the entirety of the script below into the new script in the Script Editor.
   c. Save the project.
3. Click **Run** ▶ to run the script.
4. Follow the dialog box prompts to select a table and a date field, and specify start and end dates.

   The script runs and filters the table you selected based on the field and dates you provided.

> **Tip**
>
> If you get an empty table, or a large number of records, check the dates in the unfiltered table, and rerun the script with boundary dates that you know will return a small number of records.

Things to note

- Note that in the filtered table the BETWEEN( ) function appears in the Filter text box with the actual input values you specified.
- Check the **Variables** tab. The values in the three example variables are updated with whatever values you selected and specified when you ran the script.

Example script: filter records by date

The example script filters the records in a table by date, using dates that you specify.

> **Note**
>
> You may notice that the CTOD( ) function is nested inside the BETWEEN( ) function. The CTOD( ) function converts character values to date values, which is required in this situation.
>
> If you want to know more, see "ACCEPT command" on page 1539.

```
COMMENT
This simple script allows you to apply a date filter to any Analytics table
with a date field.
END

COMMENT Prompts you to select a table in the Analytics project.
ACCEPT "Select a table with a date field:" FIELDS "xf" TO v_table_name

COMMENT Opens the selected table.
OPEN %v_table_name%

COMMENT Prompts you to select a field from the table.
ACCEPT "Select a date field:" FIELDS "D" TO v_date_field

COMMENT Prompts you to specify the start and end dates for the filter.
ACCEPT "Specify a start date (YYYYMMDD):" TO v_start_date, "Specify an end
date (YYYYMMDD):" TO v_end_date

COMMENT Applies the filter to the table and field you selected.
SET FILTER TO BETWEEN(%v_date_field%, CTOD(%v_start_date%), CTOD(%v_end_
date%))
```

Where to next?

Review and run a script that uses several functions to help perform a real-world task: "Putting it all together: using functions in a script" on the next page

Putting it all together: using functions in a script

In the final Analytics function tutorial, we'll put everything together by using variables with a number of functions in a script that performs a real-world task.

> **Note**
> You do not need to know anything about scripting to do this tutorial. You copy and paste the pre-written script at the bottom of the tutorial into Analytics.

What the script does

The example script allows anyone running the script to apply a date filter to any Analytics table with a date field, and then group by month the records included by the filter.

The script combines operations already explained in previous function tutorials.

How functions relate to a script

Within a single script, an Analytics scriptwriter might make use of multiple functions to perform various small but important helper tasks that contribute to the overall data analysis performed by the script.

Including a function in a script does not change the way the function works. Functions in scripts behave in exactly the same way they behave when you test them in isolation in the Analytics command line.

Suggested activities

- **Review the script**

  Review the example script at the bottom of the tutorial. Analytics scripts are executed in sequence, line by line. So you can proceed sequentially down the script and read each `COMMENT` to get a general idea of what the script logic is doing.

  `COMMENT` lines are not part of the script logic and are not executed.

- **Understand what the functions are doing**

  Pay special attention to the functions contained in the script. The functions are highlighted in brown. Refer to the table above the script for additional detail about the small task performed by each function.

If you've done the previous function tutorials, most of the functions in the script and the tasks they perform will already be familiar to you.

- **Run the script**

  Once you are familiar with the script and the functions it contains, copy and paste the script into Analytics and run it to see how the script interactivity works.

How to run the example script in Analytics

1. Open an Analytics project that contains one or more tables with date fields.

   `Sample Project.ACL` has several tables with date fields.

2. Create a new, empty script:
   a. In the **Navigator**, right-click a folder or the top-level project entry and select **New > Script**.
   b. Copy and paste the entirety of the script below into the new script in the Script Editor.
   c. Save the project.
3. Click **Run**  to run the script.
4. Follow the dialog box prompts to select a table and a date field, specify start and end dates, and select a numeric subtotal field.

Example script: filter and group records

The example script does two main things:

- filters the records in a table by date, using dates that you specify
- groups by month the records included in the filter

Don't worry if you can't understand all the script syntax. The main point is to see the various Analytics functions in action in a script.

A `COMMENT` before each piece of script syntax explains in simple terms what the syntax is doing.

The functions used in the example script

The purpose of each function used in the example script is described below.

In the script, the functions are highlighted brown.

Function in script	Purpose
DATE( )	Converts the `MIN1` and `MAX1` variables from the Datetime to the Character data type. The Character data type is required in order to display the contents of the variables in a text string in a dialog box. `MIN1` and `MAX1` are system variables automatically created by the `STATISTICS` command. They contain the oldest and the most recent dates in the date field you select.

Function in script	Purpose
ALLTRIM( )	Cleans up extra spaces around the oldest and the most recent dates when they are displayed in the dialog box.
CTOD( )	Converts the `v_start_date` and `v_end_date` variables from the Character data type to the Datetime data type. The Datetime data type is required for subtracting or comparing dates.
CTOD( )	Converts the `v_start_date` and `v_end_date` variables from the Character data type to the Datetime data type so that they are consistent with the `v_date_field` variable. All `BETWEEN` function parameters must be the same data type.
BETWEEN( )	Filters the date field based on the start and end dates you specified.
MONTH( )	Extracts the month portion from every date in the date field as a number.
CMOY( )	Extracts the month portion from every date in the date field as a character value.

Example script: filter records by date, and group filtered records by month

```
COMMENT
This script allows you to apply a date filter to any Analytics table with a
date field, and then group by month the records included by the filter.
END

COMMENT Prompts you to select a table in the Analytics project.
ACCEPT "Select a table with a date field:" FIELDS "xf" TO v_table_name

COMMENT Opens the selected table.
OPEN %v_table_name%

COMMENT Prompts you to select a date field from the table.
ACCEPT "Select a date field:" FIELDS "D" TO v_date_field

COMMENT Identifies the oldest and the most recent dates in the selected date
field.
STATISTICS ON %v_date_field%

COMMENT Assigns the oldest and the most recent dates to variables. The vari-
ables are used to display the existing date range in the dialog box where you
specify the start and end dates for the date filter. It's easier to specify
filter dates if you know what the existing date range is.
ASSIGN v_min_date = ALLTRIM(DATE(MIN1, "YYYYMMDD"))
ASSIGN v_max_date = ALLTRIM(DATE(MAX1, "YYYYMMDD"))
```

```
COMMENT Prompts you to specify the start and end dates for the date filter.
DIALOG (DIALOG TITLE "User Dialog" WIDTH 484 HEIGHT 153 ) (BUTTONSET TITLE
"&OK;&Cancel" AT 370 12 DEFAULT 1 ) (TEXT TITLE "Specify a start date:" AT 12
16 ) (EDIT TO "v_start_date" AT 156 12 DEFAULT "YYYYMMDD" ) (TEXT TITLE "Spe-
cify an end date:" AT 12 52 ) (EDIT TO "v_end_date" AT 156 48 DEFAULT
"YYYYMMDD" ) (TEXT TITLE "Date range in table:" AT 12 88 ) (TEXT TITLE "%v_
min_date%   to  %v_max_date%" AT 156 88 )

COMMENT Displays a warning if the user-specified date filter spans more than 1
year.
IF CTOD(v_end_date) - CTOD(v_start_date) > 365 OR CTOD(v_start_date) - CTOD(v_
end_date) > 365 DIALOG (DIALOG TITLE "User Dialog" WIDTH 469 HEIGHT 100 )
(BUTTONSET TITLE "&OK;&Cancel" AT 348 8 DEFAULT 1 ) (TEXT TITLE "Date range
exceeds 1 year. Monthly groupings may include records from more than 1 year."
AT 12 28 WIDTH 326 HEIGHT 33 ) (TEXT TITLE "Caution" AT 12 8 )

COMMENT Displays a warning if the user-specified start date is after the end
date.
IF CTOD(v_start_date) > CTOD(v_end_date) DIALOG (DIALOG TITLE "User Dialog"
WIDTH 469 HEIGHT 100 ) (BUTTONSET TITLE "&OK;&Cancel" AT 348 8 DEFAULT 1 )
(TEXT TITLE "Start date is after end date. Records between the two dates are
included." AT 12 28 WIDTH 326 HEIGHT 33 ) (TEXT TITLE "Caution" AT 12 8 )

COMMENT Applies the date filter to the table and field you selected.
SET FILTER TO BETWEEN(%v_date_field%, CTOD(%v_start_date%), CTOD(%v_end_
date%))

COMMENT Prompts you to select a subtotal field.
ACCEPT "Select a numeric field to subtotal for each month:" FIELDS "N" TO v_
subtotal_field

COMMENT Groups the table by month, and outputs the results to a new table.
SUMMARIZE ON MONTH(%v_date_field%) SUBTOTAL %v_subtotal_field% OTHER CMOY(%v_
date_field%, 9) TO "%v_table_name%_by_month.FIL" OPEN PRESORT
```

Where to next?

If you've completed all the tutorials in "How to use functions" on page 1424 and "Advanced use of functions" on page 1446, congratulations! You now have a solid grounding in how Analytics functions work throughout Analytics.

Here are some suggestions for continuing to increase your expertise with functions:

- **Continue to explore**
  - Check out "Top 30 Analytics functions" on page 1465 for a list of the most frequently used Analytics functions, with accompanying examples.

  - "Search and filter using Analytics functions" on page 1173 provides numerous examples of using Analytics functions to perform powerful and effective searching and filtering of data in tables.
  - Browse through the entire set of Analytics "Functions overview" on page 2016. Familiarize yourself at a high level with all the different things that functions can do.
- **Don't forget about functions**

  When you're presented with a data analysis challenge in your Analytics work, ask yourself, "Could a function help me out? Or several functions in combination?"

  With data analysis using Analytics commands, a large part of the challenge can be preparing the data for analysis. Functions, either singly, or in combination, are often critical in the preparation.

Top 30 Analytics functions

The top 30 functions in ACLScript are useful across a number of different tasks. Use these functions regularly to help you prepare, parse, convert, and harmonize data in your scripts.

Removing leading and trailing space

Character fields in Analytics tables often contain leading or trailing spaces because the field widths are fixed length. When you need to perform an operation using the data from a character field, you can remove these spaces so that the string only contains the actual data.

ALLTRIM( )

Returns a string with leading and trailing spaces removed from the input string.

> **Note**
> It is good practice to use `ALLTRIM()` on any character field that you are using as input for another function so that no leading or trailing spaces affect the returned value.

Example

The `Vendor_Number` field contains the value `" 1254"`. You need to remove this extra space from `Vendor_Number` so that you can harmonize the field with data in another table.

```
COMMENT returns "1254"
ALLTRIM(Vendor_Number)
```

Synchronizing alphabetic case

String comparison in Analytics is case-sensitive, therefore it is useful to synchronize the casing of all data in a field before you perform any comparisons, joins, or relations using the data.

UPPER( )

Returns a string with alphabetic characters converted to uppercase.

Example

The `Last_Name` field contains the value `"Smith"`. You need to make this value uppercase to compare with an uppercase value from another table.

```
COMMENT returns "SMITH"
UPPER(Last_Name)
```

LOWER( )

Returns a string with alphabetic characters converted to lowercase.

Example

The `Last_Name` field contains the value `"Smith"`. You need to make this value lowercase to compare with an lowercase value from another table.

```
COMMENT returns "smith"
LOWER(Last_Name)
```

PROPER( )

Returns a string with the first character of each word set to uppercase and the remaining characters set to lowercase.

Example

The `Last_Name` field contains the value `"smith"`. You need to display it as a proper noun in your output.

```
COMMENT returns "Smith"
PROPER(Last_Name)
```

Calculating and separating strings

When you need to extract a segment of data from a longer string, or test some information about the string such as its length or contents, use these functions.

SUBSTR( )

Returns a specified substring from a string.

Example

The `GL_Account_Code` field contains the value `"001-458-873-99"`. You need to extract the first three bytes, or characters, from the string.

```
COMMENT returns "001"
ASSIGN v_start_pos = 1
ASSIGN v_length = 3
SUBSTR(GL_Account_Code, v_start_pos, v_length)
```

LAST( )

Returns a specified number of characters from the end of a string.

Example

The `GL_Account_Code` field contains the value `"001-458-873-99"`. You need to extract the last two bytes, or characters, from the string.

```
COMMENT returns "99"
ASSIGN v_num_chars = 2
LAST(GL_Account_Code, v_num_chars)
```

SPLIT( )

Returns a specified segment from a string.

Example

The `GL_Account_Code` field contains the value `"001-458-873-99"`. You need to extract the second segment of the code from the string.

```
COMMENT returns "458"
ASSIGN v_delimiter = "-"
ASSIGN v_segment_num = 2
SPLIT(GL_Account_Code, v_delimiter, v_segment_num)
```

AT( )

Returns a number specifying the starting location of a particular occurrence of a substring within a character value.

Example

The `GL_Account_Code` field contains the value `"001-458-873-99"`. You need to determine the starting byte position of the value `"458"` to test whether the GL code's second segment is "458" (start position "5").

```
COMMENT returns "5"
ASSIGN v_occurrence = 1
ASSIGN v_substring = "458"
AT(v_occurrence, v_substring, GL_Account_Code)
```

OCCURS( )

Returns a count of the number of times a substring occurs in a specified character value.

Example

The `GL_Account_Code` field contains the value `"001-458-873-99"`. You need to determine that the GL code is correctly formatted by ensuring the data contains three hyphen characters.

```
COMMENT returns "3"
ASSIGN v_substring = "-"
OCCURS(GL_Account_Code, v_substring)
```

LENGTH( )

Returns the number of characters in a string.

Example

The `GL_Account_Code` field contains the value `"001-458-873-99"`. You need to determine that the GL code is correctly formatted by ensuring the data contains 14 characters.

```
COMMENT returns "14"
LENGTH(GL_Account_Code)
```

Converting data types

Depending on the data source and import statements that produced the Analytics table, you may need to convert values in a field from one data type to another so that an operation is possible. For example, to perform arithmetic on data that was imported as character (`"12345"`), you must convert it to numeric.

STRING( )

Converts a numeric value to a character string.

Example

The `Invoice_Amount` field contains the value `12345.67`. You need to convert this to character data.

```
COMMENT returns "12345.67"
ASSIGN v_str_length = 8
STRING(Invoice_Amount, v_str_length)
```

VALUE( )

Converts a character string to a numeric value.

> **Tip**
> VALUE( ) is often used with ZONED( ) to add leading zeros.

Example

The `Invoice_Amount` field contains the value `"12345.67"`. You need to convert this to numeric data.

```
COMMENT returns 12345.67
VALUE(Invoice_Amount, 2)
```

CTOD( )

Converts a character or numeric date value to a date. Can also extract the date from a character or numeric datetime value and return it as a date. Abbreviation for "Character to Date".

Example

The `Submission_Date` field contains the value `"April 25, 2016"`. You need to convert this to datetime data.

```
COMMENT returns `20160425`
ASSIGN v_date_format = "mmm dd, yyyy"
CTOD(Submission_Date, v_date_format)
```

DATE( )

Extracts the date from a specified date or datetime and returns it as a character string. Can also return the current operating system date.

Example

The `Submission_Date` field contains the value `` `20160425` ``. You need to convert this to character data.

```
COMMENT returns "04/25/2016"
ASSIGN v_date_format = "MM/DD/YYYY"
DATE(Submission_Date, v_date_format)
```

Adding leading zeros

Convert numeric data to character data and adds leading zeros to the output when you need to harmonize fields that require leading zeros.

ZONED( )

Converts numeric data to character data and adds leading zeros to the output.

Example

The `Employee_Number` field contains the value `"254879"`. You need to convert the value to a 10-digit string with leading zeros.

> **Tip**
> You must use the `VALUE()` function to convert the character to numeric data before using the numeric as input for `ZONED()`.

```
COMMENT returns "0000254879"
ASSIGN v_str_length = 10
```

```
ASSIGN v_num_decimals = 0
ZONED(VALUE(Employee_Number, v_num_decimals), v_str_length)
```

BINTOSTR( )

Returns Unicode character data converted from ZONED or EBCDIC character data. Abbreviation for "Binary to String".

> **Note**
> Unicode edition only. For non-Unicode editions, see `ZONED()` above.

Example

The `Employee_Number` field contains the value `"254879"`. You need to convert the value to a 10-digit string with leading zeros.

> **Tip**
> You must use the `VALUE()` function to convert the character to numeric data before using the numeric as input for `ZONED()`. You then use `BINTOSTR()` to convert the ASCII data returned from `ZONED()` to Unicode.

```
COMMENT returns "0000254879"
ASSIGN v_str_length = 10
ASSIGN v_num_decimals = 0
ASSIGN v_str_type = "A"
BINTOSTR(ZONED(VALUE(Employee_Number, v_num_decimals), v_str_length), v_
str_type)
```

Extracting datetime parts

Use these functions to isolate and extract specific components of a datetime value.

MONTH( )

Extracts the month from a specified date or datetime and returns it as a numeric value (1 to 12).

Example

The `Transaction_Date` field contains the value `` `20160815 100252` ``. You need to extract the month as character data with a leading zero.

```
COMMENT returns "08"
ASSIGN v_str_length = 2
ZONED(MONTH(Transaction_Date), v_str_length)
```

DAY( )

Extracts the day of the month from a specified date or datetime and returns it as a numeric value (1 to 31).

Example

The `Transaction_Date` field contains the value `` `20160815 100252` ``. You need to extract the day as character data.

```
COMMENT returns "15"
ASSIGN v_str_length = 2
STRING(DAY(Transaction_Date), v_str_length)
```

YEAR( )

Extracts the year from a specified date or datetime and returns it as a numeric value using the YYYY format.

Example

The `Transaction_Date` field contains the value `` `20160815 100252` ``. You need to extract the year as a numeric value.

```
COMMENT returns 2016
YEAR(Transaction_Date)
```

HOUR( )

Extracts the hour from a specified time or datetime and returns it as a numeric value using the 24-hour clock.

Example

The `Transaction_Date` field contains the value `` `20160815 100252` ``. You need to extract the hours as a numeric value.

```
COMMENT returns 10
HOUR(Transaction_Date)
```

MINUTE( )

Extracts the minutes from a specified time or datetime and returns it as a numeric value.

Example

The `Transaction_Date` field contains the value `` `20160815 100252` ``. You need to extract the minutes as a numeric value.

```
COMMENT returns 2
MINUTE(Transaction_Date)
```

SECOND( )

Extracts the seconds from a specified time or datetime and returns it as a numeric value.

Example

The `Transaction_Date` field contains the value `` `20160815 100252` ``. You need to extract the seconds as a numeric value.

```
COMMENT returns 52
SECOND(Transaction_Date)
```

CDOW( )

Returns the name of the day of the week for a specified date or datetime. Abbreviation for "Character Day of Week".

Example

The `Transaction_Date` field contains the value `` `20160815 100252` ``. You need to extract the name of the day as character data.

```
COMMENT returns "Mon"
CDOW(Transaction_Date, 3)
```

CMOY( )

Returns the name of the month of the year for a specified date or datetime. Abbreviation for "Character Month of Year".

Example

The `Transaction_Date` field contains the value `` `20160815 100252` ``. You need to extract the name of the month as character data.

```
COMMENT returns "Aug"
CMOY(Transaction_Date, 3)
```

Manipulating strings

Remove or replace segments of character fields using these functions.

INCLUDE( )

Returns a string that includes only the specified characters.

Example

The `Address` field contains the value `"12345 ABC Corporation"`. You need to extract the address number and exclude the name of the company.

```
COMMENT returns "12345"
ASSIGN v_chars_to_return = "0123456789"
INCLUDE(Address, v_chars_to_return)
```

EXCLUDE( )

Returns a string that excludes the specified characters.

Example

The `Address` field contains the value `"12345 ABC Corporation"`. You need to extract the name of the company and exclude the address number.

```
COMMENT returns "ABC Corporation"
ASSIGN v_chars_to_exclude = "0123456789"
EXCLUDE(Address, v_chars_to_exclude)
```

REPLACE( )

Replaces all instances of a specified character string with a new character string.

Example

The `Address` field contains the value `"12345 Acme&Sons"`. You need to replace the "&" character with the word " and ".

```
COMMENT returns "12345 Acme and Sons"
ASSIGN v_target_char = "&"
ASSIGN v_replacement_char = " and "
REPLACE(Address, v_target_char, v_replacement_char)
```

OMIT( )

Returns a string with one or more specified substrings removed.

Example

The `Address` field contains the value `"12345 Fake St"`. You need to extract the address without the street suffix.

```
COMMENT returns "12345 Fake"
ASSIGN v_chars_to_omit = "St"
OMIT(Address, v_chars_to_omit)
```

REVERSE( )

Returns a string with the characters in reverse order.

Example

The `Report_Line` field contains the value `"001 Correction 5874.39 CR "`. You need to reverse the value and omit any leading or trailing spaces.

```
COMMENT returns "RC 93.4785 noitcerroC 100"
REVERSE(ALLTRIM(Report_Line))
```

BLANKS( )

Returns a string containing a specified number of blank spaces.

Example

You need to create a computed field for a region name based on a value in the `region_code` field. You must ensure that the default value you specify at the end of the command is at least as long as the longest input value.

```
COMMENT BLANKS returns a string of 8 " " chars
ASSIGN v_length = 8
DEFINE FIELD region COMPUTED

"Southern" IF region_code = 1
"Northern" IF region_code = 2
"Eastern" IF region_code = 3
"Western" IF region_code = 4
BLANKS(v_length)
```

Working with scripts

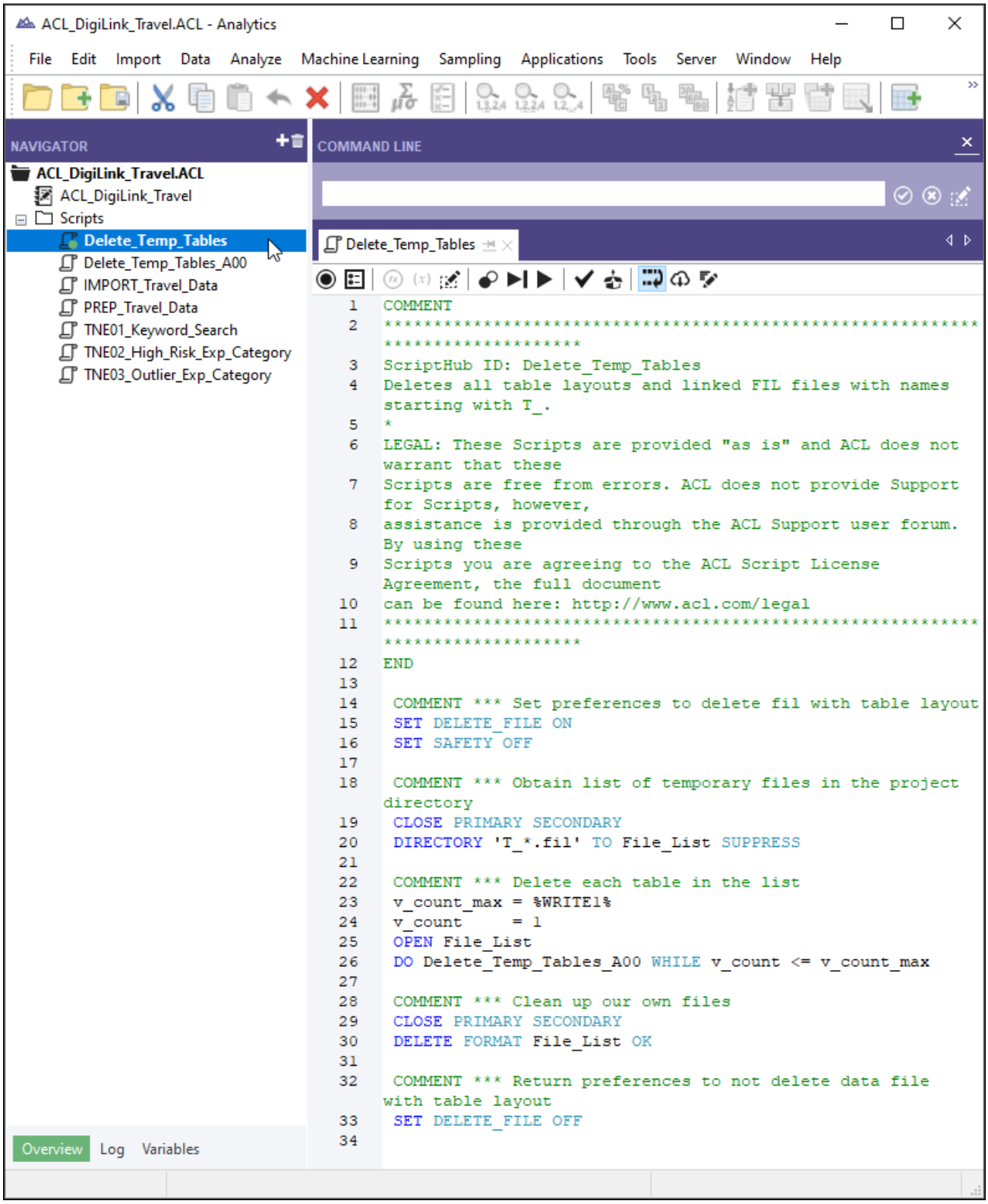
Analytics scripts are written in plain text in the Script Editor, which is part of Analytics. Like any plain text content, you can copy-and-paste text freely between the Script Editor and other plain text sources.

Scripts in the Analytics user interface

Scripts are visualized individually in the **Overview** tab of the **Navigator**. Although the scripts are visualized individually, all scripts in an Analytics project are contained within the single Analytics project file (`*.acl`).

If required, you can export an individual script as a separate `.aclscript` file saved outside the Analytics project. A script exported as a separate file can later be imported into any Analytics project.

The main Analytics user interface below shows a number of scripts in the **Navigator**, with the first script open in the Script Editor.


ACL_DigiLink_Travel.ACL - Analytics
File Edit Import Data Analyze Machine Learning Sampling Applications Tools Server Window Help
NAVIGATOR
ACL_DigiLink_Travel.ACL
ACL_DigiLink_Travel
Scripts
Delete_Temp_Tables
Delete_Temp_Tables_A00
IMPORT_Travel_Data
PREP_Travel_Data
TNE01_Keyword_Search
TNE02_High_Risk_Exp_Category
TNE03_Outlier_Exp_Category
Overview Log Variables
COMMAND LINE
Delete_Temp_Tables
1 COMMENT
2 ********** ScriptHub ID: Delete_Temp_Tables
3 Deletes all table layouts and linked FIL files with names starting with T_.
4 *
5 LEGAL: These Scripts are provided "as is" and ACL does not warrant that these
6 Scripts are free from errors. ACL does not provide Support for Scripts, however,
7 assistance is provided through the ACL Support user forum. By using these
8 Scripts you are agreeing to the ACL Script License Agreement, the full document
9 can be found here: http://www.acl.com/legal
10 ********** END
11
12 COMMENT *** Set preferences to delete fil with table layout
13 SET DELETE_FILE ON
14 SET SAFETY OFF
15
16 COMMENT *** Obtain list of temporary files in the project directory
17 CLOSE PRIMARY SECONDARY
18 DIRECTORY 'T_*.fil' TO File_List SUPPRESS
19
20 COMMENT *** Delete each table in the list
21 v_count_max = %WRITE1%
22 v_count = 1
23 OPEN File_List
24 DO Delete_Temp_Tables_A00 WHILE v_count <= v_count_max
25
26 COMMENT *** Clean up our own files
27 CLOSE PRIMARY SECONDARY
28 DELETE FORMAT File_List OK
29
30 COMMENT *** Return preferences to not delete data file with table layout
31 SET DELETE_FILE OFF
32
33
34

Scriptwriting tools

You have a number of tools to choose from when you create, edit, or debug scripts. Some of the tools allow you to automatically create ACLScript syntax without requiring that you know the syntax in advance.

You can use the tools individually, or in combination, to create new scripts or to modify existing scripts.

Tool	Description
Script Editor	Author or edit scripts by typing ACLScript syntax.
Command log	Automatically create ACLScript syntax by selecting entries in the command log, which retains a record of all the commands that have been executed in a project.
Script Recorder	Automatically create ACLScript syntax by recording any ACLScript commands that you execute while script recording is enabled.
Syntax capture	Automatically create ACLScript syntax by recording any ACLScript commands that you access in the Analytics user interface while syntax capture is enabled. The commands are not actually executed.
Table history	Automatically create ACLScript syntax from the table history of any Analytics table that has been created as the output of an ACLScript command or series of commands.
ScriptHub	Import scripts or snippets from ScriptHub, a web-based library of Analytics scripted items.
Debugging features	Set break points, or step through scripts one line at a time, to test or debug scripts.

Creating and editing scripts

You have several options for creating scripts:

- **Script Editor** – type script syntax in the **Script Editor** window
- **Command log** – copy script syntax from the command log
- **Script Recorder** – record the syntax for commands as you execute them
- **Syntax capture** – insert the syntax for commands as you select them in the user interface without executing them
- **Table history** – copy script syntax from the history of an output table

You can use these options in combination. For example, you could begin a script by copying syntax from the command log, and then add additional lines of syntax directly in the **Script Editor** window.

To edit an existing script in Analytics, you must use the **Script Editor**.

You also have the option of using a text editor of your choice, and copying and pasting syntax into an Analytics script.

Syntax auto-completion

As you type syntax in the **Script Editor**, Analytics provides auto-completion for ACLScript commands and keywords, and automatic on-screen help for function parameters.

You can turn off auto-completion by selecting **Disable auto complete in scripts** in the **Interface** tab in the **Options** dialog box (**Tools > Options**). On-screen help for function parameters cannot be disabled.

Import scripts or snippets from ScriptHub

Another option for creating scripts is to make use of the content in ScriptHub. ScriptHub is a web-based library of Analytics scripted items developed by Galvanize employees and the Galvanize user community. As part of your subscription, you can download and use any of the content in ScriptHub.

For more information, see "Importing from ScriptHub" on page 1503.

Create or edit a script in the Script Editor

You can create or edit scripts by typing the required ACLScript syntax directly into the **Script Editor**, or by using one of the other supported methods for entering syntax.

> **Note**
> When you create or edit a script you must ensure that each ACLScript command is entered on a separate line.

Show me how

Create or edit the script

1. Complete one of the following steps to open the script in the **Script Editor**:
   - To create a new script, select **File > New > Script**.

     The script is created with the name **New_Script**. Right-click the name and select **Rename** to rename the script.

     > **Note**
     > Script names are limited to 64 alphanumeric characters. The name can include the underscore character (_), but no other special characters, or any spaces. The name cannot start with a number.

   - To open an existing script, double-click the script in the **Overview** tab in the **Navigator**.
2. Add, modify, or delete ACLScript syntax in the **Script Editor**.

   > **Tip**
   > You can use these shortcut keys for common actions:
   > - **Ctrl+Z** – undo one or more actions
   > - **Ctrl+Y** – redo one or more actions
   > - **Ctrl+S** – save the Analytics project, including the open script

3. (Optional) Position the cursor at an appropriate place in the script and complete any of the following steps to insert one or more specific items:

Item	Steps
Project item name (table, script, view, workspace, or index)	a. Right-click and select **Insert > Project Item**. b. Select the type of item from the **Item Type** drop-down list. c. Select one or more item name(s), and click **OK**.
Field name	a. From the **Script Editor** toolbar, click **Insert Field**  . b. Select one or more field name(s) and click **OK**.
Expression	a. From the **Script Editor** toolbar, click **Insert Expression**  . b. Create an expression and click **OK**.
Dialog box	a. From the **Script Editor** toolbar, click **Build New Dialog**  . b. Create a custom dialog box, click **Close**, and click **OK**.

Item	Steps
	For more information, see "Creating custom dialog boxes" on page 1508.
Date and time	a. Right-click and select **Insert > Date & Time**. b. Enter or select a date, datetime, or time and click **OK**.
HighBond token	a. Right-click and select **Insert > HighBond Token** to insert a HighBond access token in the script. The **Manage API tokens** page opens in your browser. You may be required to first sign in to Launchpad. b. Do one of the following: • **Use an existing token** - In the **Token** column, click the partially masked token that you want to use and enter your HighBond account password. The unmasked token is displayed. • **Create a new token** - Click **Create token > HighBond API** and enter your HighBond account password. A new HighBond token is created. **Tip** Use an existing token unless you have a reason for creating a new one. If the existing token does not work, create a new one. Using an existing token cuts down on the number of tokens you need to manage. c. Click **Copy** to copy the token. **Tip** Do not close the dialog box containing the token until you have successfully pasted the token into the script. d. In Analytics, paste the token at the appropriate point in the script. e. In Launchpad, close the dialog box containing the token. If you created a new token, a partially masked version of the token is added to the top of your list of tokens. For more information, see Creating and managing access tokens. **Caution** Safeguard your access tokens like any account password. They contain information unique to your HighBond account. You should not share access tokens.

4. Select **File > Save Project**.
5. Click **Yes** in the confirmation dialog box.

Edit command syntax using a dialog box

Instead of manually editing commands in a script, you can edit them using the associated dialog box.

> **Note**
> This method is available only for commands that have dialog boxes.

1. Select an existing ACLScript command in the script.
2. From the **Script Editor** toolbar, click **Edit Command**  .

   The command dialog box opens.
3. In the dialog box, make the required changes to the command parameters and click **OK**.

   The script syntax is updated.

Test the script

If you want to test the script by running it, or by stepping through it, click **Run**  or **Step**  in the **Script Editor** toolbar.

> **Note**
> If you run or step through a script, all open scripts are automatically saved.

Open the project as an analysis app

If you are creating or editing analytics and want to open the project as an analysis app, click **Open as Analysis App**  in the **Script Editor** toolbar.

Create a script from the command log

You can copy log entries from the **Log** tab in the **Navigator** as the starting point for creating a new script, or to add to an existing script. The syntax of each command previously executed in Analytics is recorded in an individual log entry.

You can select the following types of log entries and copy them to a new or existing script:

- individual entries
- a series of entries associated with a table
- larger groups of entries associated with a session or a time period

Show me how

1. In the **Navigator**, click the **Log** tab to display the log.
2. Select the log entry, or group of entries, that you want to include in the script.

   If you select higher level items in the treeview, the sub-entries are automatically selected.
3. Do one of the following:

Create a new script	a. Right-click in the **Log** tab and select **Save Selected Items > Script**. b. In the **Save Script As** dialog box, enter a name for the new script and click **OK**. **Note** Script names are limited to 64 alphanumeric characters. The name can include the underscore character (_), but no other special characters, or any spaces. The name cannot start with a number. The new script is added to the **Overview** tab in the **Navigator**. The script is saved in the folder containing the active Analytics table, or in the root project folder if no table is open.
Copy syntax to an existing script	a. Right-click in the **Log** tab and select **Copy**. b. Open an existing script if one is not already open. c. In the **Script Editor**, position the cursor where you want to insert the copied syntax. d. Right-click and select **Paste**.

Create a script with the Script Recorder

The Analytics **Script Recorder** allows you to create a script by recording your actions as you work with tables and commands in the Analytics user interface. The advantage of using the **Script Recorder** to create scripts is that you do not need to manually enter the required syntax for each ACLScript command, or even know the syntax.

Only commands are captured by the **Script Recorder**. As a general rule, if the command appears in the command log, it can be captured by the **Script Recorder**.

> **Tip**
> The **Script Recorder** is also a useful tool for learning ACLScript. You can record a series of analysis steps using the **Script Recorder** and then view the resulting script to see the series of commands and syntax required to reproduce the behavior in a script.

Show me how

1. From the Analytics main menu, select **Tools > Set Script Recorder On**.

   The Script Recorder icon  is displayed in the status bar, and a checkbox is displayed to the left of the menu item, to indicate that the Script Recorder is on.

2. Perform the analysis steps or processing you want to record.

   Analytics records each processed command in a new script.

3. When you have finished analyzing or processing data, select **Tools > Set Script Recorder On** again to turn the **Script Recorder** off.

   Analytics prompts you to save the script.

4. Enter a meaningful name for the script in the text box and click **OK**.

> **Note**
>
> Script names are limited to 64 alphanumeric characters. The name can include the underscore character (_), but no other special characters, or any spaces. The name cannot start with a number.

Create a script with syntax capture

Syntax capture allows you to use Analytics menus and dialog boxes to automatically insert ACLScript syntax into a script.

Show me how

1. Open the script that you want to work with.
2. (Optional) Open the table that you want to work with.

> **Tip**
>
> If you start syntax capture before you open a table, the table does not physically open in the View tab because commands are not actually executed during syntax capture. You may find it difficult to visualize subsequent commands without an open table for guidance.

3. From the **Script Editor** toolbar, click **Start Syntax Capture**  .
4. Perform the analysis steps or processing that you want to record.

   The associated syntax is automatically inserted into the script. The commands themselves are not executed.

5. Click **End Syntax Capture**  to stop inserting command syntax in the script.

Create a script from table history

You can create a script based on the history associated with an Analytics output or results table.

For example, if you add a monthly inventory table to an Analytics project and extract relevant divisions and items to a new table, you could script this process based on the table history associated with the first output table you created.

Each table created as the output of an Analytics command keeps a record of the commands used to create the table, including commands that create any intermediate tables between the original Analytics table and the output table. You can copy this table history to a new script that you can then use to automate creation of subsequent output tables.

Show me how

1. Open an output table that is the result of a process you want to automate in a script.
2. Select **Tools > Create Script from Table History**.

   If Default_View is active, Analytics prompts you to rename the view to prevent you from overwriting it when you run the new script.
3. If Analytics prompts you to rename the view, click **Rename**, enter a new name, and click **OK**.
4. Enter a name for the new script in the **Save As** dialog box and click **OK**.

   > **Note**
   >
   > Script names are limited to 64 alphanumeric characters. The name can include the underscore character (_), but no other special characters, or any spaces. The name cannot start with a number.

5. (Optional) Open and edit the new script if you want to adjust any of the script behavior.

   For example, instead of overwriting the original table you could choose to save the output to a table with a different name.

Testing and debugging scripts

The Analytics **Script Editor** includes several features that help you test or debug Analytics scripts:

- Run scripts from the cursor position in a script
- Set break points to pause the execution of a script at a specific line
- Step through scripts by executing one line at a time
- Isolate script errors
- In the associated **Variables** tab, track the creation of variables and the assignment of values to variables in real time

These features are available whenever a script is open in the **Script Editor**. If the open script calls one or more subscripts, the subscripts are automatically opened when they are called.

While a script is running in step mode, or break point mode, it is read-only, and most other Analytics functionality is disabled, including the command line. If a script error occurs, the script becomes editable, allowing you to fix the error.

When you run a script, regardless of how you run it, all open scripts are automatically saved.

> **Note**
>
> If a table remains open at the completion of running or stepping through a script, the Analytics display area automatically switches from the **Script Editor** to displaying the open table in the View tab. If you want to keep the **Script Editor** continuously displayed while you are testing or debugging scripts, you can temporarily include the CLOSE command at the end of the script.

Running scripts from the cursor

If you do not want to run or step through a script from the beginning of the script you can position the cursor in the line where you want to start script execution and right-click and select **Run From Cursor**, or **Step From Cursor**. Running or stepping through scripts from the cursor allows you to test specific portions of a script and avoid the wasted time and effort of running entire scripts needlessly.

You cannot use **Run From Cursor** or **Step From Cursor** once a script is running. You can only use these options to start execution of a script, or restart a script after you encounter or fix a script error.

> **Note**
>
> If you use script execution from the cursor to bypass a section of a script that contains prerequisite operations required by a subsequent section of the script, the subsequent section is unlikely to run correctly.

Setting break points

You can set one or more break points in an Analytics script to pause the execution of a script at a specific line. Break points allow you to test a portion of a script without having to run the entire script. They also allow you to examine the state of an Analytics project at a specific point in a script. Break points can be a useful tool as you develop and test more complex or critical portions of scripts.

Restarting a script from a break point

When you restart a script from a break point, you have the following options:

- step through the script from the break point
- run the script to the next break point, if you have inserted one
- run the script to the end
- exit the script

Blank lines and comments

If you position a break point at a blank line, or at a comment line, the script pauses at the first line of the script after the blank line or lines, or after the comment.

Persistence of break points

- Break points persist in a script even if you close the script.
- At any time, you can remove all break points from all scripts in an Analytics project by right-clicking in the **Script Editor** and selecting **Clear All Breakpoints**.
- All break points are automatically removed from all scripts in a project when you close Analytics.

Steps

Show me how

Set one or more break points

1. Open the Analytics script in which you want to set one or more break points  .
2. Click the break point column immediately to the left of the target line in the Analytics script.

   The break point column is located between the line numbering column and the left margin of the script.

   You can also set a break point by positioning the cursor in the target line in the script and pressing **F9** or clicking **Toggle breakpoint**  in the **Script Editor** toolbar.

3. To remove a break point, click the break point, or position the cursor in the target line and press **F9** or click **Toggle breakpoint**  .

Run a script with a break point

1. Click **Run**  or press **F5** to run the script to the break point.

   The script starts running and executes to the break point. While the script is running in break point mode it is read-only, and most other Analytics functionality is disabled, including the command line.

2. To move beyond the break point, click **Run**  or press **F5**.

   The script runs to the next break point, or if there are no other break points, completes execution of the script.

3. If the step arrow turns red  and stops at a line, indicating an error, the script becomes editable and you can fix the error, and then do either of the following:
   - Continue running the script from the point of the error, or from any other line, by placing the cursor in the appropriate line, and right-clicking and selecting **Run From Cursor**.
   - Restart the script from the beginning by clicking **Run**  or pressing **F5**.

   If a table is open when the error occurs, the Analytics display area automatically switches from the **Script Editor** to displaying the open table in the View tab. Switch back to the **Script Editor** to fix the error.

4. If you want to exit the script before it completes, press **Esc** and click **Yes** in the confirmation prompt.

   You can also exit the script by closing Analytics.

5. After a break point, or after fixing an error, if you want to step through the remainder of the script, do one of the following:
   - After a break point, click **Step**  or press **F10**.
   - After fixing an error, place the cursor in the appropriate line, and right-click and select **Step From Cursor**.

Stepping through scripts

You can step through an Analytics script by executing one line at a time. Stepping through a script allows you to test the execution in a controlled manner, and discover any errors at the exact line where they occur.

The step arrow

Green arrow – As you step through a script, the green step arrow indicates the line in the script that is about to be executed. When the arrow progresses one step beyond the line, the line has been executed.

Red arrow – If the line contains invalid command syntax, or some other type of error, the script stops and the step arrow turns red and does not progress, highlighting the location of the error. The script is read-only while you are stepping through, but if an error occurs the script becomes editable, allowing you to fix the error.

Steps

Show me how

1. Open the Analytics script that you want to step through.
2. Click **Step** or press **F10** repeatedly.

   The script starts when you click **Step** or press **F10**. A single line is executed, in sequence, each additional time you click **Step** or press **F10**.

   While the script is running in step mode it is read-only, and most other Analytics functionality is disabled, including the command line.
3. If the step arrow turns red, indicating an error, the script becomes editable and you can fix the error, and then do either of the following:
   - Continue stepping through the script from the point of the error, or from any other line, by placing the cursor in the appropriate line, and right-clicking and selecting **Step From Cursor**.
   - Restart the script and begin stepping through from the beginning by clicking **Step** or pressing **F10**.

   If a table is open when the error occurs, the Analytics display area automatically switches from the **Script Editor** to displaying the open table in the View tab. Switch back to the **Script Editor** to fix the error.
4. If you want to exit the script before it completes, press **Esc** and click **Yes** in the confirmation prompt.

   You can also exit the script by closing Analytics.
5. At any point, if you want to run the remainder of the script without stepping through, click **Run** or press **F5**.

Isolating script errors

Whenever you run a script in Analytics that encounters an error that causes the script to fail, the line where the error occurs is automatically highlighted in the **Script Editor**. If the **Script Editor** is not

open, it opens automatically. This identification of script errors occurs regardless of whether you run a script directly in the **Script Editor**, from the **Tools** menu, from the command line, or by right-clicking a script in the **Navigator**.

This automated error identification is a powerful troubleshooting capability, especially for errors that occur deep in nested subscripts. Analytics users with scripting ability can fix errors as they encounter them. Users unfamiliar with scripting can record the name of the script and the line number where the error occurred, which makes it easier to get help with script problems.

Using the Variables tab

The **Variables** tab, in the **Navigator**, allows you to track the creation of variables and the assignment of values to variables in real time. The tab displays the names, values, and data categories of all variables in the Analytics project. Names are listed alphabetically.

If you step through a script, any user-defined or system-generated variables in the script, at the moment of their creation, appear in the **Variables** tab, or have their value updated if they already existed. Being able to watch exactly what changes are happening with script variables, as they happen, is an important diagnostic tool that can allow you to pinpoint script errors that might be hard to locate by examining script syntax alone.

If you run a script, all changes associated with variables are displayed when a break point is reached, or when a script completes.

Multiline commands

You cannot step through the content of multiline commands such as GROUP, LOOP, or DEFINE FIELD . . . COMPUTED. If you run a script in step mode and you encounter a multiline command, the entire content of the command is executed and the step arrow is positioned at the line immediately following the multiline command.

Break points are not recognized inside multiline commands. If you set a break point inside a multiline command, the script is paused at the line immediately following the multiline command.

> **Tip**
> You may be able to test portions of the content of a multiline command by copying the content, without the surrounding command syntax, into a separate script.

Testing an analytic script that includes a PASSWORD analytic tag

If you test an analytic script by running it in Analytics and the script has a PASSWORD tag in the analytic header, Analytics automatically generates a PASSWORD command and prompts you to

enter the appropriate password. This auto-generated command saves you the labor of inserting a PASSWORD command in the script portion of the analytic script for the purposes of testing, and then removing it again before uploading the analytic script to Robots or AX Server. The auto-generated PASSWORD command is saved in the log, without the password value.

The password value is not saved when you run the analytic script in Analytics, so you must specify the password each time you run the analytic script, including running or stepping through the script from the cursor position.

Analytic scripts are regular scripts with analytic headers that allow them to run in the Robots app on the HighBond platform, or in Analytics Exchange. You can also run analytic scripts in the Analysis App window, a freestanding component of Analytics.

Run scripts

When you run a script in Analytics, each command in the script is processed in sequence until the end of the script is reached.

You cannot continue working in Analytics while the script is running, and you can run only one script at a time. However, using the DO SCRIPT command, you can create scripts that call and run other scripts.

Script status

While a script is running, Analytics displays the processing status and the name of the script, or subscript, in the status bar.

When the script finishes running, an icon appears in the status bar indicating if the script ran successfully to completion  , or failed  . If a script fails, the line where the error occurs is automatically highlighted  in the **Script Editor**.

If necessary, you can stop the processing of a script by pressing the **Esc** key, or by closing Analytics.

Run a script from the main menu

There are two different ways to run a script from the main menu:

- **Applications menu** – choose a specific script name from a custom menu

  For more information, see "Adding custom items to the Analytics main menu" on page 153.

- **Tools option** – access a pick list of all the scripts in a project
  1. Select **Tools > Run Script**.
  2. In the **Do Script** dialog box, select the script to run from the list of available scripts in the project.
  3. If you want to specify a condition that must evaluate to true in order for the script to run, do one of the following:
     - enter a logical expression in the **If** text box
     - click **If** to create a logical expression using the **Expression Builder**

     The logical expression is evaluated just once to determine if the script should run. If the expression evaluates to false the script does not run.
  4. Click **OK**.

Run a script from the Overview tab

In the **Overview** tab in the **Navigator**, right-click the script in the treeview and select **Run**.

Run a script from the Script Editor

Open the script in the **Script Editor**, and click **Run**  in the **Script Editor** toolbar.

If you have made changes to a script in the **Script Editor**, the changes are automatically saved when you run the script.

Run a script from the Windows command line

You can run a script from the Windows command line, or from a batch file (`*.bat`), which allows you to schedule the script using a utility such as Windows Task Scheduler and run it unattended.

The command line syntax uses this basic form:

```
acl_executable_path_and_filename acl_project_path_and_filename
</vVarName=value> /bScript_name </min>
```

Example

The command line syntax below opens `Sample Project.ACL` and runs a script called Calculate_Median_Value.

```
"C:\Program Files (x86)\ACL Software\ACL for Windows 14\ACLWin.exe"
"C:\Users\username\Documents\ACL Data\Sample Data Files\Sample Project.ACL" /vv_
table_name="Ap_Trans" /vv_field_name="Invoice_Amount" /bCalculate_Median_Value
```

Command line syntax

> **Note**
> Specify the full paths to the Analytics executable and the Analytics project, including the file name and the file extension. Enclose the path in quotation marks if the path includes any spaces.

Parameter	Details	Example
"*ACL_exe_path_and_file-name*"	Specifies the path to the Analytics executable file, and the executable file name (`ACLWin.exe`).	`"C:\Program Files (x86)\ACL Software\ACL for Windows 14\ACLWin.exe"`
"*ACL_project_path_and_file-name*"	Specifies the path to the Analytics project file, and the file name of the project (`*.acl`) containing the script.	`"C:\Users\username\Documents\ACL Data\Sample Data Files\Sample Project.ACL"`

/v
optional
	Specifies variable names and assigns values. The variables are automatically initialized when the Analytics project opens.
	Do not enter a space between the `/v` switch and the variable name. For example, for the variable `v_table_name`:
	`/vv_table_name="Ap_Trans"`
	Note
	The data type of an assigned value must match the data type of the variable in the script. If the data types are mismatched, an "Expression type mismatch" error occurs and the script fails.
	Use quotation marks to qualify character values, and backquotes to qualify datetime values.
		Character variables
		`/vv_table_name="Ap_Trans" /vv_field_name="I-Invoice_Amount"`
		Numeric variable
		`/vv_materiality=10000`
		Datetime variables
		`/vv_start_date=`20180101` /vv_end_date=`20180331``

Para-meter	Details	Example
/b	Specifies the name of the script to run. Do not enter a space between the `/b` switch and the script name.	`/bCalculate_Median_Value`
/min optiona-l	Specifies that Analytics is minimized when it opens.	

Guidelines for creating a script that runs unattended

Avoid user interaction	Do not include any of the following user interaction commands: ◦ `DIALOG` ◦ `ACCEPT` ◦ `PASSWORD` ◦ `PAUSE` Instead, specify any required variables, and assign values, using the command line syntax explained above.
Suppress confirmation dialog boxes	Add the `SET SAFETY OFF` command at the beginning of the script so that files can be overwritten as necessary without displaying a confirmation dialog box. Add the `SET SAFETY ON` command at the end of the script to restore the default behavior. Add the `OK` parameter after any command, such as `DELETE` or `RENAME`, that normally displays a confirmation dialog box.
Exit Analytics	End the script with the `QUIT` command to exit Analytics.

Run a script from a Windows shortcut

You can run a script from a Windows shortcut.

1. Create a shortcut for Analytics.
2. Right-click the shortcut and select **Properties**.
3. In the **Target** field, enter the appropriate command line syntax (see above).
4. Click **OK**.
5. Double-click the shortcut to run the script.

Customizing the Script Editor

You can customize the **Script Editor** by enabling or disabling word wrap, by changing the text colors and fonts, and by changing the background color of the editing area. You can also disable keyword auto-completion. On-screen help for function parameters cannot be disabled.

The following types of text can be customized:

- **Default Style** – All script text that is not a comment or an ACLScript keyword
- **Comment Style** – Script comments
- **Command Style** – ACLScript command keywords
- **Parameter Style** – ACLScript parameter keywords
- **Function Style** – ACLScript function keywords

1. To enable or disable word wrapping, click **Word Wrap**  in the **Script Editor** toolbar.
2. To change the text styles or background color in the **Script Editor**, do the following:
   a. Select **Tools > Options**.
   b. Click the **Application Font** tab.

   > **Note**
   > The fixed-width and proportional font settings on the **Application Font** tab apply to all application fonts, not only to the fonts in the **Script Editor**.

   c. In the **Script Editor Settings** area, select a text style or **Background Color** and click **Change Color**.
   d. In the **Color** dialog box, select a color from the **Basic colors** area or from the color palette. Or, if you know the **Red**, **Green**, and **Blue** (RGB) values of the color you want to use, enter them in the appropriate text boxes.
   e. Click **OK**.
   f. To bold a selected style, select **Bold**.
   g. To italicize a selected style, select **Italic**.
   h. Click **OK**.

   If you want to revert to the text styles and background color used when Analytics was first installed, click **Factory** at the bottom of the **Options** dialog box. Clicking **Factory** sets all options on all **Options** tabs to their default settings, not just the text styles and background color in the **Script Editor**.
3. If you want to disable keyword auto-completion, do the following:
   a. Select **Tools > Options**.
   b. Click the **Interface** tab.
   c. Select **Disable auto complete in scripts**.
   d. Click **OK**.

Copy scripts

You can copy a script from one Analytics project to another. You can copy a single script, or multiple scripts simultaneously.

If you want to import a script that exists as a separate file outside an Analytics project, see "Import scripts" on the next page.

1. Open the project that will contain the copied script or scripts.
2. In the **Overview** tab of the **Navigator**, right-click the Analytics project entry, or a project folder, and select **Copy from another Project > Script**.

   The Analytics project is the top-level folder in the treeview.
3. In the **Locate Project File** dialog box, locate and select the Analytics project you want to copy the script or scripts from and click **Open**.
4. In the **Import** dialog box, complete any of the following tasks to add one or more scripts to the **To *project_name*** list:
   - Double-click a script.
   - **Ctrl+click** multiple scripts and then click the right-arrow button.
   - Click **Add All** to add all the scripts.

   You can remove scripts from the **To *project_name*** list by double-clicking an individual script, by using **Ctrl+click** to select multiple scripts and then clicking the left-arrow button, or by clicking **Clear All**.
5. Click **OK** to copy the script or scripts into the destination project.

   If a script with the same name already exists in the project, the copied script is given an incrementing numeric suffix.

Import scripts

You can import a script that exists as a separate `.aclscript` file outside an Analytics project. You can import only one script at a time.

If you want to import a script from another Analytics project, see "Copy scripts" on the previous page.

1. In the **Overview** tab of the **Navigator**, right-click the Analytics project entry, or a project folder, and select **Import Project Item > Script**.

   The Analytics project is the top-level folder in the treeview.

2. In the **Project** dialog box, locate and select a script file (`.aclscript`) and click **Open**.
3. Click **OK** in the confirmation dialog box.

   The script is imported into the project. If a script with the same name already exists in the project, the imported script is given an incrementing numeric suffix.

Importing from ScriptHub

ScriptHub is a web-based library of Analytics scripted items developed by Galvanize employees and the Galvanize user community. As part of your subscription, you can download and use any of the content in ScriptHub.

ScriptHub includes:

- Analytics
- Scripts to import, prepare, or analyze data
- Code snippets
- Analysis apps

You can access ScriptHub from the following locations:

- ACL for Windows
- Launchpad (www.highbond.com)
- The ScriptHub homepage (scripts.highbond.com)

Import content from ScriptHub to Analytics

You can use either of the methods outlined below to import content from ScriptHub to Analytics.

Access ScriptHub through Analytics

> **Note**
> Requires Analytics version 12 (or higher).

1. Open an Analytics project.
2. In the **Navigator**, right-click the top-level project item and select **Import from ScriptHub**.
3. If required, sign in to ScriptHub using your HighBond account.
4. In ScriptHub, find the item you want to import and click **View details**.

   > **Note**
   > Make sure that you read the contents in the **Script Details**, which includes important information about prerequisites, data requirements, and limitations.

5. At the top of the **Script Files** panel, click **Download All Script Files**  .

   The ScriptHub content downloads and appears in the **Navigator**.

Access ScriptHub directly

> **Note**
> Requires Analytics version 11.4 (or higher).

1. Sign in to ScriptHub (scripts.highbond.com) using your HighBond account.
2. In ScriptHub, find the item you want to import and click **View details**.

> **Note**
> Make sure that you read the contents in the **Script Details**, which includes important information about prerequisites, data requirements, and limitations.

3. Copy the ID that appears in the **ScriptHub ID** text box.
4. In an Analytics project, do one of the following:
   - If you are importing an analysis app, an analytic, or a script, create a new script.
   - If you are importing a code snippet, open an existing script you want to paste the snippet into.
5. Click **ScriptHub Access**  in the **Script Editor** toolbar to display the **Paste ScriptHub content link** dialog box.
6. Click **Paste** to paste the ScriptHub ID into the dialog box.
7. Click **Done**.

   The ScriptHub content downloads and appears in the **Navigator**.

> **Note**
> Code snippets are inserted into the open script. Analysis apps, analytics, and scripts appear as separate scripts in the **Navigator**. In this situation you can delete the empty script you used to import the items.

Export scripts

You can export a script as a separate `.aclscript` file saved outside the Analytics project. A script exported as a separate file can later be imported into any Analytics project. You can export only one script at a time.

1. Right-click the script in the **Overview** tab of the **Navigator** and select **Export Project Item**.
2. In the **Save As** dialog box, choose a location to save the script, rename the script if required, click **Save**, and click **OK** in the confirmation dialog box.

   The script is exported to the location you specified.

> **Note**
>
> Limit the script name to 64 alphanumeric characters, not including the `.aclscript` extension, to ensure that the name is not truncated when the script is imported back into Analytics.
>
> The name can include the underscore character (_), but do not use any other special characters, or any spaces, or start the name with a number. Special characters, spaces, and a leading number are all replaced by the underscore character when the script is imported.

Creating interactive scripts

You can create interactive scripts that prompt the user for input. Unlike standard scripts that run uninterrupted, interactive scripts pause their execution until required information is provided by the user.

The benefit of interactivity

Interactivity helps you write scripts that are flexible, with broader applicability. You do not need to specify all the input information in advance, which requires that you know information like table and field names, and typically results in scripts that are single-purpose, or narrow in focus.

Using script interactivity, you can gather input information using one or more dialog boxes when the user runs the script. For example, you could use interactivity to gather any of the following input:

- user name and password
- table and field names
- file names
- amount thresholds
- date ranges
- identifiers such as merchant codes, branch codes, and vendor and customer IDs
- command parameters

Sequencing interactivity

Whenever possible, you should place all interactive dialog boxes at the beginning of a script so that the remainder of the script can run without interruption.

If interactive dialog boxes occur mid-script, the user may no longer be attending the script execution at the point that input is required, and the script remains stalled until the input is provided.

Three methods for creating interactivity

Analytics provides three methods for creating interactivity in scripts. Each method is associated with an Analytics command.

The ACCEPT and PASSWORD commands can only be created using ACLScript syntax. The DIALOG command can be created using ACLScript syntax, or the syntax can be autogenerated using the **Dialog Builder**, a visual utility.

Command	Description
"ACCEPT	The ACCEPT command creates the default interactive dialog box, which supports two

Command	Description
command" on page 1539	methods of user input: ◦ **Text box** - gathers information that the user must type in, such as dates, or vendor or customer IDs ◦ **Project item list** - presents a list of Analytics project items, such as tables, fields, or variables, to the user The list of items is dynamically populated based on the contents of the Analytics project in which the script is run. You can create separate dialog boxes that prompt for one item at a time, or you can create one dialog box that prompts for multiple items.
"DIALOG command" on page 1652 Dialog Builder	The DIALOG command creates a custom interactive dialog box. Custom dialog boxes support more advanced layout options, and five methods of user input: ◦ **Text box** - gathers information that the user must type in, such as dates, or vendor or customer IDs ◦ **Check box** - presents a binary choice to the user – that is, the associated option can be either on or off ◦ **Radio buttons** - present mutually exclusive options to the user – that is, only one of the presented options can be selected at a time ◦ **Drop-down list** - presents a list of custom, text-based options to the user ◦ **Project item list** - presents a list of Analytics project items, such as tables, fields, or variables, to the user The list of items is dynamically populated based on the contents of the Analytics project in which the script is run. You can create separate dialog boxes that prompt for one item at a time, or you can create one dialog box that prompts for multiple items.
"PASSWORD command" on page 1893	The PASSWORD command creates a simple dialog box with a single field for entering a password. When a user enters a password, the characters are displayed as asterisks (*) in the dialog box. The password is retained in memory of the duration of the Analytics session, but it does not appear in either the script or the log.

Creating custom dialog boxes

The Analytics **Dialog Builder** allows you to create one or more custom dialog boxes to gather user input during execution of a script.

You can use a custom dialog box to perform various functions:

- prompt a user for input, such as a table name, a field name, or a date range
- allow a user to select from among several options
- display more information than a standard message box
- dynamically list Analytics project items

> **Note**
>
> Using a custom dialog box to enter passwords is not secure. You should use the "PASSWORD command" on page 1893 instead.

Dialog box controls

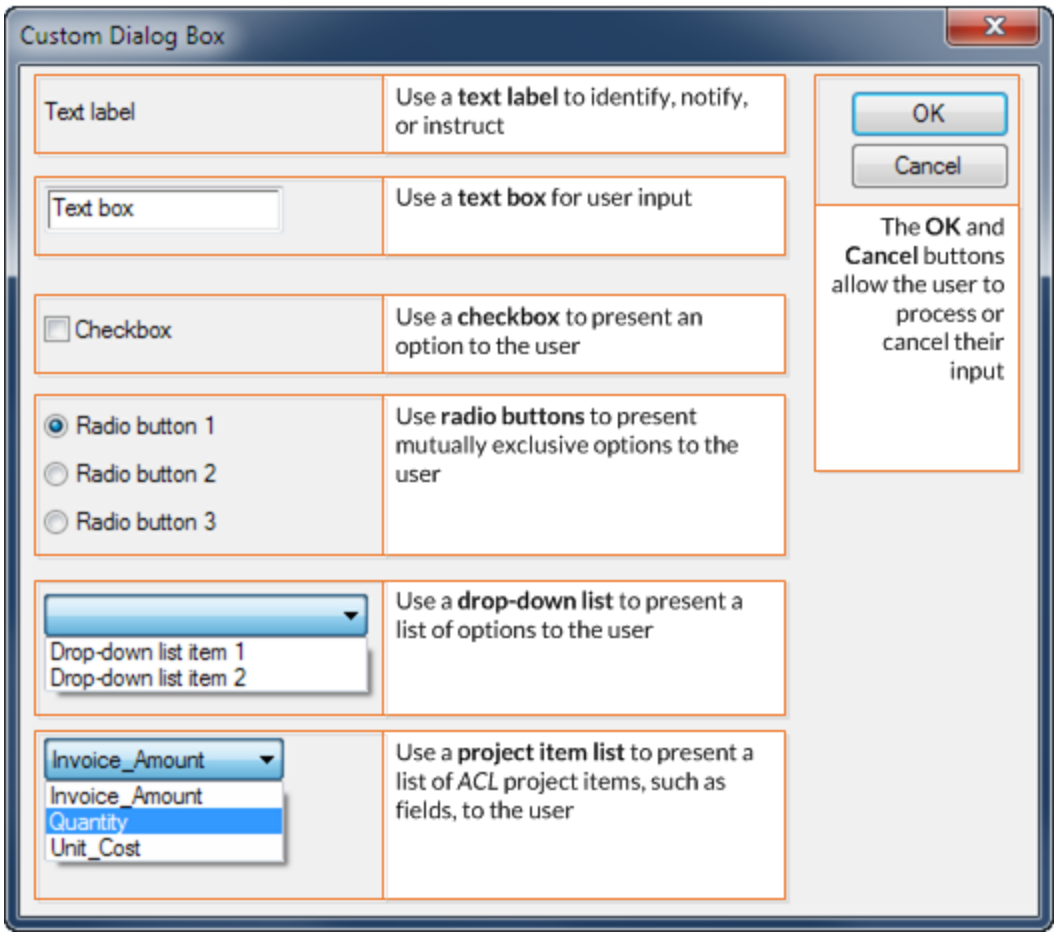
In the **Dialog Builder**, you design and build a custom dialog box by adding user input **controls** to the dialog box.

Controls are small, interactive software components that provide different ways of gathering user input required by a script. You add one or more controls to the basic dialog box and configure them to suit your needs.

The following controls are available:

- text label
- text box
- checkbox
- radio button
- drop-down list
- project item list

The sample custom dialog box below provides an example of each type of control.



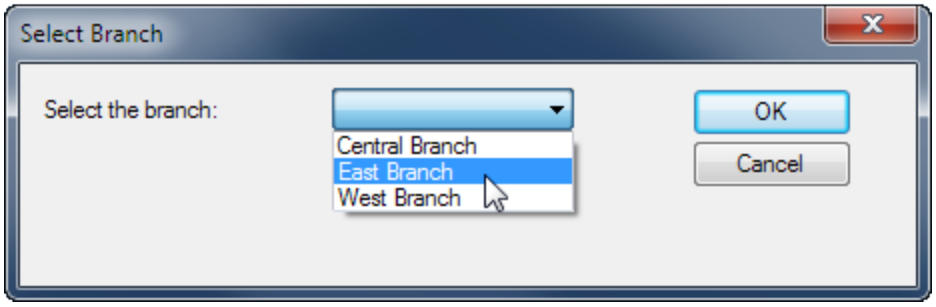
Dialog box automatically translated to a command

When you save a newly created custom dialog box, Analytics automatically translates the visual dialog box to a matching ACLScript `DIALOG` command.

The command is inserted at the line in the script where the cursor is positioned. When you run the script, the visual dialog box appears.

The example below shows a simple custom dialog box with one text label and one drop-down list, and the corresponding `DIALOG` command.

Custom dialog box



Corresponding DIALOG command

```
DIALOG (DIALOG TITLE "Select Branch" WIDTH 464 HEIGHT 116 ) (BUTTONSET TITLE
"&OK;&Cancel" AT 336 12 DEFAULT 1 ) (TEXT TITLE "Select the branch:" AT 12 16
) (DROPDOWN TITLE "Central Branch;East Branch;West Branch" TO "DROPDOWN1" AT
156 12 )
```

Create a custom dialog box – general steps

1. Open the Analytics script that you want to add the custom dialog box to.
2. Position the cursor in the line in the script where you want to insert the `DIALOG` command.

   > **Note**
   > Position the cursor in a blank line. Create a new blank line if necessary.

3. Click **Build New Dialog**  .

   Analytics displays the **Dialog Builder** with a default title of "User Dialog".

4. Complete any of the following steps to create the custom dialog box:
   - Double-click the **Dialog Builder** to modify the title or the size of the dialog box.

     You specify the **Width** and **Height** of the dialog box in pixels. You can also resize the dialog box by dragging the bottom right corner of the working area in the **Dialog Builder** (**Snap to Grid** must be on).

   - Click **Snap to Grid**  to turn the grid on or off in the **Dialog Builder**.

     Use the grid to align controls in the layout area. When the grid is turned on, the top-left corner of each control is aligned with the closest grid point.

   - On the left side of the **Dialog Builder**, click a control icon and then click in the layout area to add the control.

> **Note**
> The steps for adding and designing specific controls appear below.

5. Add as many controls as you need.
6. If you need to modify a control once you have add it, double-click the control.
7. If you need to delete a control from the **Dialog Builder**, select the control and click **Delete**  .

> **Note**
> You cannot delete the **OK** and **Cancel** buttons, but you can rename them (see below).

8. Click **Close** to exit the **Dialog Builder**.
9. Click **OK** in the confirmation dialog box to save your changes.

   Analytics displays the corresponding `DIALOG` command in the **Script Editor**. You can see all the controls in the custom dialog box by scrolling to the right.
10. Optional. In the `DIALOG` command in the script, edit the label text for the OK or the Cancel buttons.

    Typically you should not edit the OK and Cancel labels. If you do edit the labels, ensure that the positive value (for example, Yes) comes before the negative value (for example, No).

    Edit only the label text. For example: `"&Yes;&No"`

Modify a custom dialog box

If you need to modify a custom dialog box after you have created it, position the cursor in the corresponding `DIALOG` command and click **Edit Command**  .

Add a text label

Use the text control to add a text label to the custom dialog box.

A text label can be used for any of the following purposes:

- to identify another control
- to provide a notification
- to prompt or instruct users
- to provide any other text-based information required in the custom dialog box

Text labels are display-only and are not associated with any interactive functionality.

Steps

Show me how

1. In the **Dialog Builder**, click **Text**  and then click the layout area at the position where you want the top left corner of the control.

   The **Text** dialog box opens.
2. In the **Label** field, type the text that you want to display in the custom dialog box.

   You are limited to a maximum of 255 characters, including spaces.
3. Optional. If you want to specify the exact position of the control, modify the **x** (horizontal) and **y** (vertical) values, which are specified in pixels.

   > **Tip**
   > You can also position the control by dragging it in the **Dialog Builder**.

4. Optional. If you want to specify a specific size for the control, deselect **Auto** beside the **Width** or **Height** fields and modify the values, which are specified in pixels.
   - **Auto selected** – the text control automatically adjusts to the size of the text contained by the control
   - **Auto deselected** – the text control remains at the specified size, regardless of the size of the text contained by the control

   > **Tip**
   > You can also resize the control using the resize handles in the **Dialog Builder**.

5. Under **Alignment**, specify the alignment of the text in the control by selecting **Left**, **Right**, or **Center**.
6. Click **OK** to add the control to the **Dialog Builder**.

Add a text box

Use the edit box control to add a text box to the custom dialog box.

A text box gathers information that the user must type in, such as dates, or vendor or customer IDs.

Edit box variable

The edit box control creates a character variable for storing the user input.

Steps

Show me how

1. In the **Dialog Builder**, click **Edit Box**  and then click the layout area at the position where you want the top left corner of the control.

The **Editbox** dialog box opens.

2. Optional. In the **Variable** field, type the name of the variable that will store the value input by the user in the custom dialog box.

   You can choose to keep the default variable name of `EDITn`.

3. Optional. In the **Default Text** field, specify a default input value for the text box.

   If the user does not specify an input value, the default value is used.

4. Optional. If you want to specify the exact position of the control, modify the **x** (horizontal) and **y** (vertical) values, which are specified in pixels.

   > **Tip**
   > You can also position the control by dragging it in the **Dialog Builder**.

5. Optional. If you want to specify a specific size for the control, deselect **Auto** beside the **Width** or **Height** fields and modify the values, which are specified in pixels.
   - **Auto selected** – the edit box control automatically adjusts to the size of the text contained by the control
   - **Auto deselected** – the edit box control remains at the specified size, regardless of the size of the text contained by the control

   > **Tip**
   > You can also resize the control using the resize handles in the **Dialog Builder**.

6. Click **OK** to add the control to the **Dialog Builder**.

Add a checkbox

Use the checkbox control to add a checkbox to the custom dialog box.

A checkbox presents a binary choice to the user – that is, the associated option can be either on or off. For example, you could use a checkbox to allow a user to either include or exclude the Email Address field in a data extract from a personnel table.

Combinations of options

Use multiple checkboxes to allow a user to select any combination of options in a custom dialog box. If options are mutually exclusive, use radio buttons instead.

Checkbox variable

The checkbox control creates a logical variable for storing the user input. The variable stores a value of True if the checkbox is selected, and False if the checkbox is unselected.

Steps

Show me how

1. In the **Dialog Builder**, click **Check Box**  and then click the layout area at the position where you want the top left corner of the control.

   The **Checkbox** dialog box opens.
2. Optional. In the **Variable** field, type the name of the variable that will store the value input by the user in the custom dialog box.

   You can choose to keep the default variable name of `CHECKBOX`*n*.
3. In the **Label** field, type the text that you want to accompany the checkbox.

   You are limited to a maximum of 255 characters, including spaces.
4. Optional. If you want to specify the exact position of the control, modify the **x** (horizontal) and **y** (vertical) values, which are specified in pixels.

   > **Tip**
   > You can also position the control by dragging it in the **Dialog Builder**.

5. Optional. If you want to specify a specific size for the control, deselect **Auto** beside the **Width** or **Height** fields and modify the values, which are specified in pixels.
   - **Auto selected** – the checkbox control automatically adjusts to the size of the text contained by the control
   - **Auto deselected** – the checkbox control remains at the specified size, regardless of the size of the text contained by the control

   > **Tip**
   > You can also resize the control using the resize handles in the **Dialog Builder**.

6. Under **Initial State**, specify whether the checkbox is **Unchecked** or **Checked** when the custom dialog box first opens.
7. Click **OK** to add the control to the **Dialog Builder**.

Add radio buttons

Use the radio button control to add two or more radio buttons to the custom dialog box.

Radio buttons present mutually exclusive options to the user – that is, only one of the presented options can be selected at a time. For example, you could use two radio buttons to allow a user to select either:

- amounts less than $5000
- amounts greater than or equal to $5000

Mutually exclusive options

Use multiple radio buttons to allow a user to select only one from a number of options in a custom dialog box. If the options are not mutually exclusive, use checkboxes instead.

Radio button variable

The radio button control creates a numeric variable for storing the user input. The variable stores a value of 1 if the first radio button is selected, 2 if the second radio button is selected, and so on.

Steps

Show me how

1. In the **Dialog Builder**, click **Radio Button**  and then click the layout area at the position where you want the top left corner of the control.

   The **Radiobuttons** dialog box opens.
2. Optional. In the **Variable** field, type the name of the variable that will store the value input by the user in the custom dialog box.

   You can choose to keep the default variable name of `RADIO`*n*.
3. In the **Label** field, type the text that you want to accompany the first radio button and click **Add**.

   You are limited to a maximum of 255 characters, including spaces.

   The radio button is added to the **Label List**.
4. Add additional labels for each additional radio button that you want.

   Each additional radio button is added to the end of the **Label List**.

   > **Note**
   > Because the radio button control creates mutually exclusive options, you should have at least two radio buttons.

5. Optional. Instead of adding a radio button to the end of the **Label List**, you can use any of these other options:

Option	Description
Insert	Allows you to insert a radio button anywhere in the **Label List**. Before you click **Insert**, select the list item immediately below where you want to insert the new radio button.
Replace (Rename)	Allows you to replace a radio button in the **Label List**. **Replace** essentially renames the radio button.

Option	Description
	Before you click **Replace**, select the list item that you want to replace with the new radio button.
Delete	Allows you to delete a radio button from the **Label List**. Select the list item that you want to delete and click **Delete**.
Set Default	Allows you to specify which radio button is selected by default when the custom dialog box first opens. Select the list item that you want to specify as the default and click **Set Default**.

6. Optional. If you want to specify the exact position of the control, modify the **x** (horizontal) and **y** (vertical) values, which are specified in pixels.

> **Tip**
> You can also position the control by dragging it in the **Dialog Builder**.

7. Optional. If you want to specify a specific size for the control, deselect **Auto** beside the **Width** or **Height** fields and modify the values, which are specified in pixels.
   - **Auto selected** – the radio button control automatically adjusts to the size of the text contained by the control
   - **Auto deselected** – the radio button control remains at the specified size, regardless of the size of the text contained by the control

> **Tip**
> You can also resize the control using the resize handles in the **Dialog Builder**.

8. Under **Alignment**, specify whether the radio buttons have a **Horizontal** or **Vertical** alignment in the custom dialog box.
9. Click **OK** to add the control to the **Dialog Builder**.

Add a drop-down list

Use the drop-down list control to add a drop-down list to the custom dialog box.

A drop-down list presents a list of custom, text-based options to the user. The user can select only one of the options at a time. For example, you could use a drop-down list to allow a user to select:

- a month of the year
- a category
- a company department or branch

Mutually exclusive options

The options in a drop-down list are mutually exclusive. You could use radio buttons to achieve a similar result, but for lists of more than a few items, drop-down lists are more compact and easier to use.

If you want to allow a user to select more than one option at a time, use checkboxes instead.

Drop-down list variable

The drop-down list control creates a character variable for storing the user input.

Steps

Show me how

1. In the **Dialog Builder**, click **Drop-down List**  and then click the layout area at the position where you want the top left corner of the control.

   The **Dropdown list** dialog box opens.
2. Optional. In the **Variable** field, type the name of the variable that will store the value input by the user in the custom dialog box.

   You can choose to keep the default variable name of `DROPDOWN`*n*.
3. In the **Label** field, type the text that you want to accompany the first drop-down list item and click **Add**.

   You are limited to a maximum of 255 characters, including spaces.

   The list item is added to the **Label List**.
4. Add additional labels for each additional list item that you want.

   Each additional list item is added to the end of the **Label List**.

   > **Note**
   > Because the drop-down list control creates mutually exclusive options, you should have at least two list items.

5. Optional. Instead of adding a list item to the end of the **Label List**, you can use any of these other options:

Option	Description
Insert	Allows you to insert a list item anywhere in the **Label List**. Before you click **Insert**, select the list item immediately below where you want to insert the new item.

Option	Description
Replace (Rename)	Allows you to replace a list item in the **Label List**. **Replace** essentially renames the list item. Before you click **Replace**, select the list item that you want to replace with the new item.
Delete	Allows you to delete a list item from the **Label List**. Select the list item that you want to delete and click **Delete**.
Set Default	Allows you to specify which list item is selected by default when the custom dialog box first opens. Select the list item that you want to specify as the default and click **Set Default**.

6. Optional. If you want to specify the exact position of the control, modify the **x** (horizontal) and **y** (vertical) values, which are specified in pixels.

> **Tip**
> You can also position the control by dragging it in the **Dialog Builder**.

7. Optional. If you want to specify a specific size for the control, deselect **Auto** beside the **Width** or **Height** fields and modify the values, which are specified in pixels.
   - **Auto selected** – the drop-down list control automatically adjusts to the size of the text contained by the control
   - **Auto deselected** – the drop-down list control remains at the specified size, regardless of the size of the text contained by the control

> **Tip**
> You can also resize the control using the resize handles in the **Dialog Builder**.

8. Click **OK** to add the control to the **Dialog Builder**.

Add a project item list

Use the project item list control to add a project item list to the custom dialog box.

A project item list presents a list of Analytics project items, such as tables or fields, to the user. The list of items is dynamically populated based on the contents of the Analytics project in which the script is run.

The user can select only one of the options at a time. For example, you could use a project item list to allow a user to select:

- a table for a particular month, from all the tables for the year
- a particular numeric field, from all the numeric fields in a table
- a particular subscript, from among several possible subscripts

Categories of project items

When you create a project item list you do not specify actual project items, you specify a category of project item. For example, you specify the **Character Fields** category if you want to present the user with a list of all character fields in the open table when the script runs.

Based on the category or categories you specify, Analytics dynamically populates the project item list in the custom dialog box.

Available categories

The following categories are available:

◦ Character Fields ◦ Numeric Fields ◦ Datetime Fields ◦ Logical Fields	◦ Character Variables ◦ Numeric Variables ◦ Datetime Variables ◦ Logical Variables	◦ Tables ◦ Views ◦ Scripts ◦ Indexes ◦ Workspaces

Project item list variable

The project item list control creates a character variable for storing the user input.

Steps

Show me how

1. In the **Dialog Builder**, click **Project Item List**  and then click the layout area at the position where you want the top left corner of the control.

   The **Project item list** dialog box opens.
2. Optional. In the **Variable** field, type the name of the variable that will store the value input by the user in the custom dialog box.

   You can choose to keep the default variable name of `ITEM`*n*.
3. In the **Category** drop-down list, select the category of project item that you want in the project item list and click **Add**.

   For example, if you select **Numeric Fields**, the project item list contains all numeric fields in the open table when the script is run.

   The category is added to the **Category List**.
4. Optional. Add additional categories that you want.

   Each additional category is added to the end of the **Category List**.

> **Caution**
> Adding dissimilar categories, such as tables and fields, or scripts and variables, is potentially confusing for users. A best practice is to add only similar categories, such as character fields and numeric fields.

5. Optional. Instead of adding a category to the end of the **Category List**, you can use any of these other options:

Option	Description
Insert	Allows you to insert a category anywhere in the **Category List**. Before you click **Insert**, select the category immediately below where you want to insert the new category.
Replace	Allows you to replace a category in the **Category List**. Before you click **Replace**, select the category that you want to replace with the new category.
Delete	Allows you to delete a category from the **Category List**. Select the category that you want to delete and click **Delete**.

6. Optional. In the **Default** field, specify a project item that is selected by default when the custom dialog box first opens.

   For example, you could specify a particular table name, or field name.

> **Note**
> Do not specify a **Category** name.
> Make sure you exactly replicate the spelling of the project item, including any underscores (_).

7. Optional. If you want to specify the exact position of the control, modify the **x** (horizontal) and **y** (vertical) values, which are specified in pixels.

> **Tip**
> You can also position the control by dragging it in the **Dialog Builder**.

8. Optional. If you want to specify a specific size for the control, deselect **Auto** beside the **Width** or **Height** fields and modify the values, which are specified in pixels.
   - **Auto selected** – the project item list control automatically adjusts to the size of the text contained by the control
   - **Auto deselected** – the project item list control remains at the specified size, regardless of the size of the text contained by the control

> **Tip**
> You can also resize the control using the resize handles in the **Dialog Builder**.

9. Click **OK** to add the control to the **Dialog Builder**.

Find and replace text

You can search for, and optionally replace, strings or words in Analytics scripts.

1. Click the location in the **Script Editor** where you want to start searching the script.

   To search the whole script click the left corner on the first line.
2. Right-click and select **Find**.
3. In the **Replace** dialog box, enter the following information:
   - **Find What** – Specify the string or word to find.
   - **Replace With** – (Optional) If you want to replace the value you are searching for, specify the value to replace it with.
   - **Match whole word only** – Only return matches where the search value is an exact match for a word. For example, by default "int" would be found in "integer", but if this option is specified only "int" would be found and shorter or longer strings would be ignored.
   - **Match case** – Select this option to make the search case-sensitive. For example, by default searching for "Integer" would match "Integer" and "integer", but if this option is specified only "Integer" would be found.
4. Complete one of the following tasks:
   - Click **Find Next** to locate the first, or next, instance of the search string. You can click this button repeatedly to move from one match to the next.
   - Click **Replace** to replace a currently highlighted match with the **Replace With** value.
   - Click **Replace All** to replace all matched values. Analytics displays a count of the number of occurrences that have been replaced.

Display variables

You can display the current values of all system and user-defined variables in an Analytics project.

- Do one of the following:
  - Click **Display Variables**  in the toolbar.

    You may need to add the **Display Variables** button to the toolbar.
  - Type `DISPLAY VARIABLES` in the command line and press **Enter**.

  All variables in the project, and their current values, are displayed on screen.

Maintain variables

You can use the **Variables** dialog box to add, modify, duplicate, rename, or delete variables in an Analytics project.

1. Select **Edit > Variables**.

   Analytics displays a list of all system and user-defined variables in the project. System variables are those created automatically by commands.

2. If you want to add a new, user-defined variable, do the following:
   a. Click **New** to open the **Expression Builder**.
   b. Enter the expression or the value to assign to the variable in the **Expression** text box.
   c. Type the variable name in the **Save As** text box.

      If you want the variable to be permanently saved with the Analytics project, preface the name with an underscore – for example, `_varname`. Variables with names that are not prefaced with an underscore are retained for the duration of the current Analytics session only.

      > **Note**
      > Do not use non-English characters, such as é, in the names of variables. Variable names that contain non-English characters will cause scripts to fail.

   d. Click **OK**.
3. If you want to work with an existing variable, select it in the list and click one of the following:
   - **OK** – Modify the selected variable in the **Expression Builder**. The variable definition is updated when you click **OK** to close the **Expression Builder**.
   - **Duplicate** – Duplicate the selected variable, and click **Done** to create an exact copy of the variable, or click **OK** to modify the expression or value used by the variable.
   - **Rename** – Enter a new name in the highlighted text box, and click **OK**. Click **Done** to use the existing value of the variable, or click **OK** to modify the expression or value used by the variable.
   - **Delete** – Delete the variable, click **Delete** again in the confirmation dialog box, and click **Done** to close the dialog box. You can use **Shift+click** or **Ctrl+click** to select multiple variables for deletion.

Commands overview

ACLScript commands perform operations on data that are often broad in scope.

For example, the SUMMARIZE command groups records based on identical values in a field, and calculates subtotals and statistical values for each group.

A number of commands output results to a new Analytics table. Other commands perform various application tasks.

A full list of commands available in Analytics, organized by category, appears on subsequent pages:

Conventions and usage

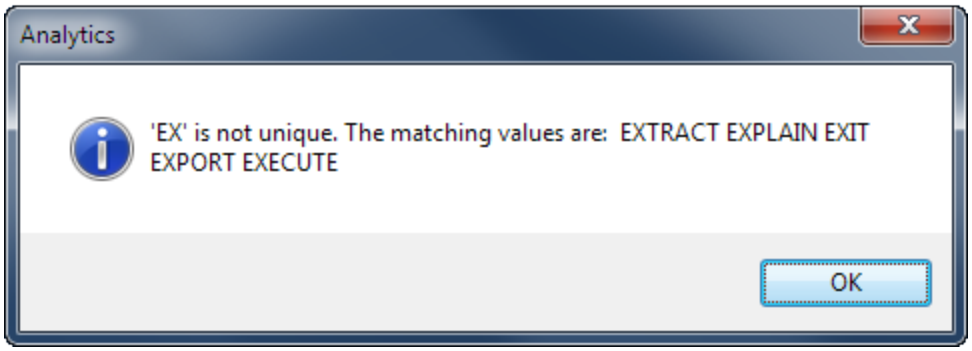
Abbreviating command names

> **Caution**
>
> ACL recommends that you do not abbreviate command names in scripts, and that you use the full version of each name.
>
> Abbreviation makes scripts harder to read and to understand. Without complete command names, searching commands in the online help becomes more difficult.
>
> Abbreviation is especially problematic if your scripts will be modified or inherited by someone else who may not be familiar with the abbreviations.

When specifying commands in scripts, you can abbreviate their names. You must include enough leading characters from a command name to uniquely identify the command among all Analytics commands.

For example:

- `EXT` uniquely identifies the `EXTRACT` command and therefore is a valid abbreviation.
- `EX` does not uniquely identify the `EXTRACT` command and generates an error message.



You can make an abbreviation as short as you want, provided that it still uniquely identifies the command.

For example, all the following abbreviations are valid for the `OPEN` command:

- `OPE`
- `OP`
- `O`

> **Note**
> As abbreviations get shorter they become harder for other users to recognize.

The order of parameters in commands

> **Note**
> As a scripting best practice, Analytics script writers should sequence parameters in exactly the same order that they appear in the command log when you run a command through the Analytics user interface.

Many Analytics commands allow some flexibility in the order of their parameters. For example, these three variations of the same CLASSIFY command all perform an identical operation, and all execute correctly:

```
CLASSIFY ON CUSTNO SUBTOTAL AMOUNT IF AMOUNT >= 100 TO "Classify_1.FIL" OPEN
APPEND KEY CODES STATISTICS
```

```
CLASSIFY ON CUSTNO SUBTOTAL AMOUNT KEY CODES IF AMOUNT >= 100 TO "Classify_
1.FIL" OPEN APPEND STATISTICS
```

```
CLASSIFY ON CUSTNO IF AMOUNT >= 100 SUBTOTAL AMOUNT STATISTICS KEY CODES TO
"Classify_1.FIL" APPEND OPEN
```

A few commands require that one or more parameters appear in a specific order. The required order is stated in the topics for those commands.

> **Note**
> The physical order of parameters in commands has no effect on the order that Analytics processes the parameters. For example, the scope parameters (ALL, FIRST, NEXT, WHILE) are applied before the IF parameter, regardless of the relative position of the parameters.

Command documentation conventions

Convention	Used for:
UPPERCASE	ACLScript keywords. In the generic syntax sections, keywords that are **not** enclosed in angled brackets `< >` are required syntax items. > **Note** > Throughout Analytics documentation, command and parameter keywords are presented in uppercase, which is simply a formatting convention. Analytics does not require that keywords are entered in uppercase.
italic	User-supplied command parameters.
| (vertical bar)	Separates syntax items enclosed in brackets or braces. You can use only one of the items.
`< >` (angled brackets)	Optional syntax items. Do not type the brackets.
`{ }` (braces)	Required syntax items. Do not type the braces.
`<,...n>`	Indicates the preceding item can be repeated *n* number of times. The occurrences are separated by commas.
`<...n>`	Indicates the preceding item can be repeated *n* number of times. The occurrences are separated by blanks.
[label] ::=	The name of a block of syntax. This convention is used to group and label sections of lengthy syntax or a unit of syntax that can be used in more than one location. Each location in which the block of syntax can be used is indicated with the label enclosed in square brackets. For example: [field_

Convention	Used for:
	syntax]

Import and export data

The import commands let you import data from a variety of different data sources.

Depending on the data source, you also define the source data as part of importing it. Defining data means specifying attributes such as field names, field lengths, and field data type.

The export command lets you export data to a variety of different file formats, or to Results in HighBond.

Data access by Analytics is read-only

When connecting to any data source, or importing from any data source, Analytics is strictly read-only. Analytics cannot add, update, or delete data in a data source, or modify a data source in any way. This restriction applies to all data sources accessible by Analytics: file-based data sources, databases, and cloud data services.

Analytics data files (.fil) created from imported data are also treated as read-only by Analytics. Analytics cannot alter .fil files, with the exception of refreshing the file from the data source.

.fil files are completely separate from the data source used to create them. Deleting a .fil file has no effect on the data source.

Command descriptions

Command	Description
ACCESSDATA	Imports data from a variety of ODBC-compliant data sources. The command takes the form ACCESSDATA64 or ACCESSDATA32 depending on whether you are using a 64-bit or 32-bit ODBC driver.
DEFINE TABLE DB	Defines an Analytics server table by connecting to a database table using AX Connector. You can connect to a Microsoft SQL Server, Oracle, or DB2 database.
EXPORT	Exports data from Analytics to the specified file format, or to HighBond Results.
IMPORT ACCESS	Creates an Analytics table by defining and importing a Microsoft Access database file.
IMPORT DELIMITED	Creates an Analytics table by defining and importing a delimited text file.
IMPORT EXCEL	Creates an Analytics table by defining and importing a Microsoft Excel worksheet or named range.

Command	Description
IMPORT GRCPROJECT	Creates an Analytics table by importing a HighBond Projects table.
IMPORT GRCRESULTS	Creates an Analytics table by importing a HighBond Results table or interpretation.
IMPORT MULTIDELIMITED	Creates multiple Analytics tables by defining and importing multiple delimited files.
IMPORT MULTIEXCEL	Creates multiple Analytics tables by defining and importing multiple Microsoft Excel worksheets or named ranges.
IMPORT ODBC	Creates an Analytics table by defining and importing data from an ODBC data source. ODBC stands for Open Database Connectivity, a standard method for accessing databases.
IMPORT PDF	Creates an Analytics table by defining and importing an Adobe PDF file.
IMPORT PRINT	Creates an Analytics table by defining and importing a Print Image (Report) file.
IMPORT SAP	Creates an Analytics table by importing data from an SAP system using Direct Link.
IMPORT XBRL	Creates an Analytics table by defining and importing an XBRL file.
IMPORT XML	Creates an Analytics table by defining and importing an XML file.
RETRIEVE	Retrieves the result of a Direct Link query submitted for background processing.

Profile and verify data

The profile commands let you count records, total numeric fields, and create a statistical profile of data.

The verify commands provide different ways to examine the integrity of a data set. For example, you can test for data validity, data sequence, gaps, and duplicates.

Command descriptions

Command	Description
BENFORD	Counts the number of times each leading digit (1-9) or leading digit combination occurs in a field, and compares the actual count to the expected count. The expected count is calculated using the Benford formula.
COUNT	Counts the total number of records in the current view, or only those records that meet the

Command	Description
	specified condition.
DUPLICATES	Detects whether duplicate values or entire duplicate records exist in an Analytics table.
FUZZYDUP	Detects nearly identical values (fuzzy duplicates) in a character field.
GAPS	Detects whether a numeric or datetime field in an Analytics table contains one or more gaps in sequential data.
OUTLIERS	Identifies statistical outliers in a numeric field. Outliers can be identified for the field as a whole, or for separate groups based on identical values in one or more character, numeric, or datetime key fields.
PROFILE	Generates summary statistics for one or more numeric fields, or numeric expressions, in an Analytics table.
SEQUENCE	Determines if one or more fields in an Analytics table are in sequential order, and identifies out-of-sequence items.
STATISTICS	Calculates statistics for one or more numeric or datetime fields in an Analytics table.
TOTAL	Calculates the total value of one or more fields in an Analytics table.
VERIFY	Checks for data validity errors in one or more fields in an Analytics table by verifying that the data is consistent with the field definitions in the table layout.

Sort data

The sort commands provide two different methods for sorting records in Analytics. The `INDEX` command temporarily reorders an existing table. The `SORT` command produces a new table with physically reordered records.

Command descriptions

Command	Description
INDEX	Creates an index for an Analytics table that allows access to the records in a sequential order rather than a physical order.
SORT	Sorts records in an Analytics table into an ascending or descending sequential order, based on a specified key field or fields. The results are output to a new, physically reordered Analytics table.

Group data

The group commands let you group records based on identical or similar values. Depending on the command, you can group text values, numbers, or dates, or a combination of these types.

Command descriptions

Command	Description
AGE	Groups records into aging periods based on values in a date or datetime field. Counts the number of records in each period, and also subtotals specified numeric fields for each period.
CLASSIFY	Groups records based on identical values in a character or numeric field. Counts the number of records in each group, and also subtotals specified numeric fields for each group.
CLUSTER	Groups records into clusters based on similar values in one or more numeric fields. Clusters can be uni-dimensional or multidimensional.
CROSSTAB	Groups records based on identical combinations of values in two or more character or numeric fields, and displays the resulting groups in a grid of rows and columns. Counts the number of records in each group, and also subtotals specified numeric fields for each group.
HISTOGRAM	Groups records based on values in a character or numeric field, counts the number of records in each group, and displays the groups and counts in a bar chart.
OUTLIERS	Identifies statistical outliers in a numeric field. Outliers can be identified for the field as a whole, or for separate groups based on identical values in one or more character, numeric, or datetime key fields.
STRATIFY	Groups records into numeric intervals based on values in a numeric field. Counts the number of records in each interval, and also subtotals specified numeric fields for each interval.
SUMMARIZE	Groups records based on identical values in one or more character, numeric, or datetime fields. Counts the number of records in each group, and also subtotals specified numeric fields for each group.

Combine data

The combine data commands provide several different ways for combining data inside Analytics.

For an overview of combing data in Analytics, see "Combining data" on page 840.

Command descriptions

Command	Description
APPEND	Combines records from two or more Analytics tables by appending them in a new Analytics table.
DEFINE RELATION	Defines a relation between two Analytics tables.
EXTRACT	Extracts data from an Analytics table and outputs it to a new Analytics table, or appends it to an existing Analytics table. You can extract entire records or selected fields.
FUZZYJOIN	Uses fuzzy matching to combine fields from two Analytics tables into a new, single Analytics table.
JOIN	Combines fields from two Analytics tables into a new, single Analytics table.
MERGE	Combines records from two sorted Analytics tables with an identical structure into a new Analytics table that uses the same sort order as the original tables.

Sample data

Analytics has three types of sampling:

- record sampling (attributes sampling)
- monetary unit sampling
- classical variables sampling

The type of sampling you choose depends on the nature of the analysis you are doing, and the nature of the data.

For guidance on which type of sampling to use, see "Sampling data" on page 949.

Sequence of sampling commands

The sampling commands are designed to be used in a specific sequence.

Sequence for classical variables sampling

1. `CVSPREPARE` - stratifies a population, and calculates the appropriate sample size for each stratum
2. `CVSSAMPLE` - draws the sample of records
3. `CVSEVALUATE` - projects errors found in the sample to the entire population of records

Sequence for record sampling, or monetary unit sampling

1. `SIZE` - calculates the appropriate sample size
2. `SAMPLE` - draws the sample of records
3. `EVALUATE` - projects errors found in the sample to the entire population of records

Command descriptions

Command	Description
CVSPREPARE	Stratifies a population, and calculates a statistically valid sample size for each stratum, for classical variables sampling.
CVSSAMPLE	Draws a sample of records using the classical variables sampling method.
CVSEVALUATE	For classical variables sampling, provides four different methods for projecting the results of sample analysis to the entire population.
SIZE	Calculates a statistically valid sample size, and sample interval, for record sampling or monetary unit sampling.
SAMPLE	Draws a sample of records using either the record sampling or monetary unit sampling method.
EVALUATE	For record sampling or monetary unit sampling, projects errors found in sampled data to the entire population, and calculates upper limits on deviation rate, or misstatement amount.

Machine learning

The machine learning commands let you predict classes or numeric values, or discover patterns, in unlabeled data.

Command descriptions

Command	Description
CLUSTER	Groups records into clusters based on similar values in one or more numeric fields. Clusters can be uni-dimensional or multidimensional.
TRAIN	Uses automated machine learning to create an optimum predictive model using a training data set.
PREDICT	Applies a predictive model to an unlabeled data set to predict classes or numeric values associated with individual records.

Field, record, and table

Commands in this group perform different operations on fields, records, or tables – the core items used to organize and display data in Analytics.

Command descriptions

Command	Description
ACTIVATE	Adds field definitions stored in an Analytics workspace to the existing set of field definitions in an Analytics table layout.
CREATE LAYOUT	Creates an empty Analytics table layout, which may be required in certain scripting situations.
DEFINE COLUMN	Creates and adds one or more columns to an existing view.
DEFINE FIELD	Defines a physical data field in an Analytics table layout.
DEFINE FIELD...COMPUTED	Defines a computed field in an Analytics table layout.
DEFINE REPORT	Creates a new view or opens an existing view.
DEFINE VIEW	Defines a new view or overwrites an existing view.
EXTRACT	Extracts data from an Analytics table and outputs it to a new Analytics table, or appends it to an existing Analytics table. You can extract entire records or selected fields.
FIELDSHIFT	Shifts the start position of a field definition in a table layout.
FIND	Searches an indexed character field for the first value that matches the specified character string.
IMPORT LAYOUT	Imports an external table layout file (.layout) to an Analytics project.
LIST	Outputs the data in one or more fields in an Analytics table to a display formatted in columns.
LOCATE	Searches for the first record that matches the specified value or condition, or moves to the specified record number.
NOTES	Creates, modifies, or removes a note associated with an individual record in an Analytics table.
OPEN	Opens an Analytics table and the associated data file.

Command	Description
REFRESH	Updates the data in an Analytics table from its associated data source.
SAVE	Copies an Analytics table and saves it with a different name, or saves an Analytics project.
SAVE LAYOUT	Saves an Analytics table layout to an external table layout file (.layout), or saves table layout metadata to an Analytics table.
SAVE TABLELIST	Saves a list of all tables in an Analytics project to an Analytics table or a CSV file.
SAVE WORKSPACE	Creates and saves a workspace.
SEEK	Searches an indexed character field for the first value that matches the specified character expression or character string.
TOP	Moves to the first record in an Analytics table.

User interaction and general scripting

The user interaction and general scripting commands provide Analytics scriptwriters with a set of commands for structuring and controlling the behavior of scripts.

Command descriptions

Command	Description
ACCEPT	Creates a dialog box that interactively prompts users for one or more script input values. Each input value is stored in a named character variable.
ASSIGN	Creates a variable and assigns a value to the variable.
CALCULATE	Calculates the value of one or more expressions.
CLOSE	Closes an Analytics table, index file, or log file, or ends a **Script Recorder** session.
COMMENT	Adds an explanatory note to a script without affecting processing.
DELETE	Deletes an Analytics project item, a field from a table layout, a variable, one or more table history entries, a relation between tables, or a file in a Windows folder. Also removes a column from a view.
DIALOG	Creates a custom dialog box that interactively prompts users for one or more script input values. Each input value is stored in a named variable.
DO SCRIPT	Executes a secondary script, or an external script, from within an Analytics script.

Command	Description
ESCAPE	Terminates the script being processed, or all scripts, without exiting Analytics.
EXECUTE	Executes an application or process external to Analytics. Emulates the Windows Run command. Can be used to interact with the Windows command prompt.
GROUP	Executes one or more ACLScript commands on a record before moving to the next record in the table, with only one pass through the table. Command execution can be controlled by conditions.
IF	Specifies a condition that must evaluate to true in order to execute a command.
LOOP	Executes a series of ACLScript commands repeatedly on a record while a specified condition evaluates to true.
NOTIFY	Sends an email notification message.
PASSWORD	Creates a password definition, without a password value, that prompts users for a password while a script is running.
PAUSE	Pauses a script, and displays information in a dialog box for users.
RCOMMAND	Passes an Analytics table to an external R script as a **data frame** and creates a new table in the Analytics project using output from the external R script.
RENAME	Renames an Analytics project item or a file.
SET	Sets a configurable Analytics option.

Report

The report commands let you format, generate, and print a basic Analytics report.

Command descriptions

Command	Description
DO REPORT	Generates the specified Analytics report.
PRINT	Prints a text file, an Analytics log file, or an Analytics project item that has been exported as an external file – a script (.aclscript), a table layout (.layout), or a workspace (.wsp). You can also print a graph that has been generated by a command.
REPORT	Formats and generates a report based on the open Analytics table.

File and system

The file and system commands perform different operations at the file, project, and operating system level.

Command descriptions

Command	Description
DIRECTORY	Generates a list of files and folders in the specified directory.
DISPLAY	Displays information about the specified Analytics item type. Can also display the result of an expression, or the output of a function.
DUMP	Displays the contents of a file, or the current record, in hexadecimal, ASCII, and EBCDIC character encodings.
HELP	Launches the Analytics Help Docs in a browser.
QUIT	Ends the current session and closes Analytics.
RANDOM	Generates a set of random numbers.
SAVE LOG	Saves the entire command log, or the log entries for the current Analytics session, to an external file.

ACCEPT command

Creates a dialog box that interactively prompts users for one or more script input values. Each input value is stored in a named character variable.

> **Note**
>
> Using the ACCEPT command to enter passwords is not secure. You should use the "PASSWORD command" on page 1893 instead.
>
> The ACCEPT command is not supported in AX Server analytics.
>
> You can create a more advanced interactive dialog box with the "DIALOG command" on page 1652.

Syntax

```
ACCEPT {message_text <FIELDS project_item_category> TO variable_name} <...n>
```

Parameters

Name	Description
message_text	The label displayed in the dialog box used to prompt for input. Must be a quoted string or a character variable. When entering multiple prompts, you can separate them with commas. Using commas improves script readability, but it is not required: `ACCEPT "Specify a start date:" TO v_start_date, "Specify an end date:" TO v_end_date`
FIELDS *project_item_category* optional	Creates a drop-down list of project items for user input instead of a text box. The user can select a single project item, field, or variable from the list. *project_item_category* specifies which item types to display in the list. For example, specifying `xf` displays all the project tables in the list. Enclose *project_item_category* in quotation marks: `FIELDS "xf"` For the codes used to specify categories, see "Codes for project item categories" on page 1542. You can specify more than one code in the same prompt, but you cannot mix project items, fields, or variables.

Name	Description
TO *variable_name*	The name of the character variable to use to store the user input. If the variable does not exist, it is created. If the variable already exists, its current value is displayed in the dialog box as the default value. **Note** You cannot use non-English characters, such as é, in the names of variables that will be used in variable substitution. Variable names that contain non-English characters will cause the script to fail. The ACCEPT command creates character variables only. If you need input of another data type, you must convert the character variable to the required type in subsequent processing in a script. For more information, see "Input data type" on page 1543.

Examples

Prompting the user to select the Analytics table to open

You require a dialog box that prompts the user to select the name of the table to open. The script then opens the table the user selects:

```
ACCEPT "Select the table to open:" FIELDS "xf" TO v_table_name
OPEN %v_table_name%
```

The percent signs are required because they indicate that the table name to open is stored in the *v_table_name* variable. If the percent signs are omitted, the script attempts to open a table called "v_table_name".

Using multiple dialog boxes to gather required input

You want to create a separate dialog box for each value that the script user must enter.

You use a single prompt string in each instance of the ACCEPT command. The script generates separate dialog boxes for specifying each of the following:

- a table name
- a field on which to sample

- a sampling interval
- a random start value

```
ACCEPT "Enter the name of the table to analyze" TO v_table_name
OPEN %v_table_name%
ACCEPT "Select the field to sample" FIELDS "N" to v_field_to_sample
ACCEPT "Enter the sampling interval" TO v_sampling_interval
ACCEPT "Enter the random start value" TO v_random_start_value
SAMPLE ON %v_field_to_sample% INTERVAL v_sampling_interval FIXED v_ran-
dom_start_value RECORD TO Sample_output OPEN
```

When the script runs

1. The first dialog box prompts for the table name.
2. The second dialog box, with `FIELDS "N"`, prompts for a field selection from a drop-down list of numeric fields.
3. The third dialog box prompts for the interval value.
4. The fourth dialog box prompts for the random start value.

Using a single dialog box with multiple prompts to gather required input

You want to create a single dialog box for all values that the script user must enter.

You use multiple prompts separated by commas in the ACCEPT command to ask the user for multiple input values. The same dialog box contains prompts for the start date and the end date of a date range:

```
ACCEPT "Specify a start date:" TO v_start_date, "Specify an end date:"
TO v_end_date
```

Remarks

Interactivity

Use ACCEPT to create an interactive script. When the ACCEPT command is processed, the script pauses and a dialog box is displayed that prompts the user for input that Analytics uses in subsequent processing.

You can create separate dialog boxes that prompt for one item at a time, or you can create one dialog box that prompts for multiple items.

DIALOG versus ACCEPT

The DIALOG command allows you to create a more advanced interactive dialog box that can have one or more of the following types of controls:

- text box
- check box
- radio buttons
- drop-down list of customized values
- project item list

You also have the flexibility to customize the layout of the dialog box. For more information, see "DIALOG command" on page 1652.

Codes for project item categories

Use the following codes to specify the category of project item to display in a drop-down list.

Project categories

Code	Category
xf	Tables
xb	Scripts
xi	Indexes
xr	Views and reports
xw	Workspaces

Field categories

Code	Category
C	Character fields
N	Numeric fields
D	Datetime fields
L	Logical fields

Variable categories

Code	Category
c	Character variables
n	Numeric variables
d	Datetime variables
l	Logical variables

Input data type

ACCEPT stores the user input in one or more character variables. If you need numeric or datetime input, you can use the VALUE( ) or CTOD( ) functions to convert the contents of a character variable to a numeric or datetime value:

```
SET FILTER TO BETWEEN(%v_date_field%, CTOD(%v_start_date%), CTOD(%v_end_
date%))
```

In the example, the start and end dates for this filter are stored as character values. They must be converted to date values in order to be used with a date field that uses a Datetime data type.

Enclosing the variable name in percent signs (%) substitutes the character value contained by the variable for the variable name. The CTOD( ) function then converts the character value to a date value.

Position of the ACCEPT command

It is good practice to place all ACCEPT commands at the beginning of a script, if possible. If you ask for all input at the beginning, the script can then run unimpeded once the user enters the necessary information.

> **Note**
> You cannot use the ACCEPT command inside the GROUP command.

ACCESSDATA command

Imports data from a variety of ODBC-compliant data sources.

The command takes the form ACCESSDATA64 or ACCESSDATA32 depending on whether you are using a 64-bit or 32-bit ODBC driver.

Syntax

```
{ACCESSDATA64 | ACCESSDATA32} {CONNECTOR | ODBC {"Driver"|"Dsn"|"File"}} NAME
value <USER user_id> <PASSWORD num | PROMPT_PASSWORD> <PASSWORD num AS pass-
word_keyname <...n>> TO table_name CHARMAX max_field_length MEMOMAX max_field_
length <ALLCHARACTER> SOURCE (connection_settings) <HASH(salt_value, fields)>
SQL_QUERY
(SQL_syntax)
END_QUERY
```

Parameters

Name	Description
CONNECTOR | ODBC {"Driver"|"Dsn"|"File"}	The type of ODBC connection you want to make: ◦ **CONNECTOR** - connect using a native Analytics data connector ◦ **ODBC "Driver"** - connect using a Windows ODBC driver installed on your computer ◦ **ODBC "Dsn"** - connect using a saved DSN (data source name) on your computer ◦ **ODBC "File"** - connect using a file DSN (a saved .dsn file)
NAME *value*	The name of the Analytics data connector, the ODBC driver, or the DSN. For example: ◦ `NAME "Amazon Redshift"` ◦ `NAME "Microsoft Access Driver (*.mdb, *.accdb)"` ◦ `NAME "My Excel DSN"` ◦ `NAME "excel.dsn"`
USER *user_id* optional	The user ID for data sources that require a user ID.
PASSWORD *num* |	For data sources that require a single password:

Name	Description
PROMPT_PASSWORD optional	◦ **PASSWORD *num*** - the password definition to use ◦ **PROMPT_PASSWORD** - display a password prompt For more information, see "Using password definitions with ACCESSDATA" on page 1550. Password value suppressed When you use the Data Access window in Analytics to run the ACCESSDATA command, and you provide a password, the password value is not written to the log. Instead, the PROMPT_PASSWORD parameter is substituted. Using the PASSWORD command in conjunction with PASSWORD *num* is similar to using PROMPT_PASSWORD. Both approaches prompt the user for a password. PROMPT_PASSWORD has the benefit of allowing updating of the *user_id*.
PASSWORD *num* AS *password_keyname <...n>* optional	For data sources that require multiple passwords, the password definitions to use. *password_keyname* must replicate exactly the password key name as it appears in the connection settings specified by SOURCE. For more information, see "Using password definitions with ACCESSDATA" on page 1550.
TO *table_name*	The location to send the results of the command to: ◦ ***table_name*** - saves the results to an Analytics table Specify *table_name* as a quoted string with a .FIL file extension. For example: `TO "Output.FIL"` By default, the table data file (.FIL) is saved to the folder containing the Analytics project. Use either an absolute or relative file path to save the data file to a different, existing folder: • `TO "C:\Output.FIL"` • `TO "Results\Output.FIL"` **Note** Table names are limited to 64 alphanumeric characters, not including the .FIL extension. The name can include the underscore character (_), but no other special characters, or any spaces. The name cannot start with a number.
CHARMAX *max_field_length*	The maximum length in characters for any field in the Analytics table that originates as character data in the source from which you are importing. The default value is 50. Data that exceeds the maximum field length is truncated when imported to Analytics.
MEMOMAX *max_field_length*	The maximum length in characters for text, note, or memo fields you are importing. The default value is 100. Data that exceeds the maximum field length is truncated when imported to Analytics.
ALLCHARACTER optional	Automatically assign the Character data type to all imported fields. Once the data is in Analytics, you can assign different data types, such as Numeric

Name	Description
	or Datetime, to the fields, and specify format details. **Tip** ALLCHARACTER is useful if you are importing a table that contains numeric ID values. You can use ALLCHARACTER to prevent Analytics automatically assigning the Numeric data type to values that should use the Character data type.
SOURCE *connection_settings*	The connection settings (connection string) required to connect to the data source.
HASH(*salt_value*, *fields*) optional	Imports the specified fields as cryptographic hash values. Hash values are one-way transformations and cannot be decoded after you import the fields: ◦ ***salt_value*** – an alphanumeric string that is concatenated with the source data values to strengthen the hashing of the values in the fields. Enter the hash value as a quoted string. The salt value is limited to 128 characters. Do not use any of the following characters: `(` `)` `"` ◦ ***fields*** – a list of one or more fields to hash. Enter the fields as a quoted string and separate each field with a comma. You must specify the field name you see in the Data Access window preview and staging area, not the physical field name in the data source. **Note** The field name that is shown in the Data Access window preview is the field alias value in the SQL query (`"field_name" AS "alias"`). You must use the alias value to reference fields. `HASH("QZ3x7", "SSN_NO, CC_NO, Last_Name")` For information about comparing values hashed during import to values hashed in ACLScript, see "Comparing data hashed with ACCESSDATA to data hashed with the ACLScript HASH( ) function" on page 1556.
SQL_QUERY (*SQL_syntax*) END_QUERY	The SQL import statement. Everything inside the parentheses is part of the SQL query and must be valid SQL. **Note** You cannot use ACLScript syntax (commands or functions) in the body of the SQL import statement. You must use valid SQL syntax only.

Examples

Importing data using a native Analytics data connector

You need to import data from the Amazon Redshift cloud data service. To do so, you use the Analytics Amazon Redshift data connector:

```
ACCESSDATA64 CONNECTOR NAME "Amazon Redshift" USER "ACL_user" PROMPT_
PASSWORD TO "Entitlement_History.FIL" CHARMAX 50 MEMOMAX 100
  SOURCE( boolsaschar-
=0;cache-
size-
=100;dat-
abase-
=usage-
;de-
clarefetch-
mod-
e=0;ma-
axbytea=255;maxlongvarchar=8190;maxvarchar=255;port=5439;servername=acl_
test.high-
bond.-
com;sin-
glerow-
mod-
e=1;ssl-
lmode=require;textaslongvarchar=0;usemultiplestatments=0;useunicode=1)
  SQL_QUERY(
    SELECT
      "entitlement_history"."organization" AS "organization",
      "entitlement_history"."user_email" AS "user_email",
      "entitlement_history"."plan_id" AS "plan_id",
      "entitlement_history"."date_from" AS "date_from",
      "entitlement_history"."date_to" AS "date_to"
    FROM
      "prm"."entitlement_history" "entitlement_history"
) END_QUERY
```

Importing data using a Windows ODBC driver

You need to import data from a Microsoft Access database. To do so, you use a Windows ODBC driver to connect to MS Access and complete the import:

```
ACCESSDATA32 ODBC "Driver" NAME "Microsoft Access Driver (*.mdb)" TO
"Invoices.FIL" CHARMAX 50 MEMOMAX 100
  SOURCE( dbq=C:\Users\lachlan_murray\Documents\ACL Data\Sample Data
Files\Sample.mdb;defaultdir=C:\Users\lachlan_murray\Documents\ACL
Data\Sample Data Files;driverid=281;fil=MS Access;maxbuf-
fer-
size-
=2048;ma-
xscan-
rows=8;pagetimeout=5;safetransactions=0;threads=3;usercommitsync=Yes)
  SQL_QUERY(
    SELECT
      `Customer`.`CustID` AS `CustID`,
      `Customer`.`Company` AS `Company`,
      `Customer`.`Address` AS `Address`,
      `Customer`.`City` AS `City`,
      `Customer`.`Region` AS `Region`,
      `Customer`.`PostalCode` AS `PostalCode`,
      `Customer`.`Country` AS `Country`,
      `Customer`.`Phone` AS `Phone`,
      `Orders`.`OrderID` AS `OrderID`,
      `Orders`.`CustID` AS `Orders_CustID`,
      `Orders`.`ProdID` AS `ProdID`,
      `Orders`.`OrderDate` AS `OrderDate`,
      `Orders`.`Quantity` AS `Quantity`,
      `Product`.`ProdID` AS `Product_ProdID`,
      `Product`.`ProdName` AS `ProdName`,
      `Product`.`UnitPrice` AS `UnitPrice`,
      `Product`.`Descript` AS `Descript`,
      `Product`.`ShipWt` AS `ShipWt`
    FROM
      (`Customer` `Customer`
    INNER JOIN
      `Orders` `Orders`
          ON `Customer`.`CustID` = `Orders`.`CustID`
      )
INNER JOIN
    `Product` `Product`
      ON `Orders`.`ProdID` = `Product`.`ProdID`
```

```
WHERE
    (
      `Customer`.`Region` = 'BC'
      OR `Customer`.`Region` = 'WA'
    )
  ) END_QUERY
```

Importing data using a Windows DSN (data source name)

You need to import data from a Microsoft Excel file. To do so, you use a Windows DSN to connect to Excel and complete the import:

```
ACCESSDATA32 ODBC "Dsn" NAME "Excel Files" TO "Trans_April_15_
cutoff.FIL" CHARMAX 50 MEMOMAX 100
  SOURCE( dbq=C:\Users\lachlan_murray\Documents\ACL Data\Sample Data
Files\Trans_April.xls;defaultdir=C:\Users\lachlan_murray\Documents\ACL
Data\Sample Data Files;driverid=1046;maxbuffersize=2048;pagetimeout=5)
  SQL_QUERY(
    SELECT
      `Trans_Apr_`.`CARDNUM` AS `CARDNUM`,
      `Trans_Apr_`.`AMOUNT` AS `AMOUNT`,
      `Trans_Apr_`.`TRANS_DATE` AS `TRANS_DATE`,
      `Trans_Apr_`.`CODES` AS `CODES`,
      `Trans_Apr_`.`CUSTNO` AS `CUSTNO`,
      `Trans_Apr_`.`DESCRIPTION` AS `DESCRIPTION`
    FROM
      `Trans_Apr$` `Trans_Apr_`
    WHERE
      (
          `Trans_Apr_`.`TRANS_DATE` <= {ts '2003-04-15 00:00:00'}
      )
  ) END_QUERY
```

Remarks

For more information about how this command works, see "Working with the Data Access window" on page 357.

Using password definitions with ACCESSDATA

The data sources that you connect to with the ACCESSDATA command often require authentication using a password, a token, or some other secret authentication value. For some data sources, more than one authentication value is required.

As part of the ACCESSDATA command, you can provide this authentication by specifying one or more **password definitions**. A password definition is not the password or authentication value itself. Rather, it is like a password variable that securely stores a previously provided password or authentication value. Specifying a password definition with the ACCESSDATA command allows you to avoid displaying an actual password in clear text in the connection settings specified by SOURCE.

Creating a password definition

In an import script, you must create a password definition first, before the definition can be used by the ACCESSDATA command.

For information about creating a password definition for use in Analytics, see "PASSWORD command" on page 1893.

For information about creating a password definition for use in Robots, Analytics Exchange, or the Analysis App window, see "PASSWORD tag" on page 2530.

Two options for specifying password definitions

You have two options when specifying password definitions with ACCESSDATA:

- **PASSWORD** ***num*** – specifies a single password definition for data sources that require a single password
- **PASSWORD** ***num*** **AS** ***password_keyname*** – can be used repeatedly to specify multiple password definitions for data sources that require multiple authentication values

> **Note**
> You can use the two options separately, or you can use them together.

How PASSWORD *num* works

Use the PASSWORD *num* parameter if a data source requires only a single password.

In the example below:

1. The `PASSWORD 1` command prompts a user to enter a password and creates a password definition that securely stores the entered password.

2. In the `ACCESSDATA` command, the `PASSWORD 1` parameter references the password definition and securely passes the stored password value into the connection setting specified by `SOURCE` (`auth_accesstoken=[$pwd]`).

```
PASSWORD 1
ACCESSDATA64 CONNECTOR NAME "Concur" PASSWORD 1 TO "Concur_data_
import.FIL" CHARMAX 50 MEMOMAX 100
 SOURCE( auth_accesstoken=[$pwd];auth_type=OAuth 2.0;en-
able-
double-
buf-
fer-
r=1;host=www.concursolutions.com;useencryptedendpoints=1;userparam=all)
 SQL_QUERY(
       SELECT
               "List_Items"."Level_7_Code" AS "Level_7_Code",
               "List_Items"."Name" AS "Name",
               "List_Items"."Level_10_Code" AS "Level_10_Code",
               "List_Items"."Level_8_Code" AS "Level_8_Code",
               "List_Items"."URI" AS "URI",
               "List_Items"."Id" AS "Id",
               "List_Items"."Level_3_Code" AS "Level_3_Code",
               "List_Items"."List_Id" AS "List_Id",
               "List_Items"."Level_4_Code" AS "Level_4_Code",
               "List_Items"."Level_1_Code" AS "Level_1_Code",
               "List_Items"."Parent_Id" AS "Parent_Id",
               "List_Items"."Level_2_Code" AS "Level_2_Code",
               "List_Items"."Level_5_Code" AS "Level_5_Code",
               "List_Items"."Level_6_Code" AS "Level_6_Code",
               "List_Items"."Level_9_Code" AS "Level_9_Code"
       FROM
               "Concur"."List_Items" "List_Items"
) END_QUERY
```

How PASSWORD *num* AS *password_keyname* works

Use the PASSWORD *num* AS *password_keyname* parameter if a data source requires multiple passwords or authentication values.

In the example below:

1. In Robots or Analytics Exchange, the `//PASSWORD` tags in an analytic header create four password parameters for which the user must enter authentication values. The four

parameters create four password definitions that securely store the entered values.
2. In the `ACCESSDATA` command, the four `PASSWORD` parameters reference the password definitions and securely pass the stored authentication values into the connection settings specified by `SOURCE`:
   - `oauthclientid=`
   - `oauthclientsecret=`
   - `oauthaccesstoken=`
   - `oauthaccesstokensecret=`

For more information, see "Configuring ACCESSDATA to work with multiple password definitions" on the facing page.

```
COMMENT
//ANALYTIC TYPE IMPORT Import Twitter data
//PASSWORD 1 Enter OAuth Client ID:
//PASSWORD 2 Enter OAuth Client Secret:
//PASSWORD 3 Enter OAuth Access Token:
//PASSWORD 4 Enter OAuth Access Token Secret:
//RESULT TABLE Twitter_user_data
END

ACCESSDATA64 CONNECTOR NAME "Twitter" PASSWORD 1 AS oauthclientid
PASSWORD 2 AS oauthclientsecret PASSWORD 3 AS oauthaccesstoken PASSWORD
4 AS oauthaccesstokensecret TO "Twitter_user_data.FIL" CHARMAX 50
MEMOMAX 100
 SOURCE( oau-
thcli-
entid-
=;oa-
uth-
cli-
entsecret-
=;oa-
authaccesstoken=;oauthaccesstokensecret=;readonly=true;drivertype=ACL
Connector for Twit-
ter-
;con-
nec-
tonopen-
n=true;convertdatetimetogmt=true;limitkeysize=255;maptolongvarchar=-
1;maptowvarchar-
=true;u-
pper-
case-
iden-
```

```
tifi-
ers-
=false;su-
pportenhancedsql=true;proxyauthscheme=BASIC;proxyautodetect=true;_
persist_=encrypted-dp{AQA ... kX3E8yyh05HoG1rH4bm1lhwudUQ==}})
 SQL_QUERY(
       SELECT
               "Users"."ID" AS "ID",
               "Users"."Name" AS "Name",
               "Users"."Screen_Name" AS "Screen_Name",
               "Users"."Location" AS "Location",
               "Users"."Profile_URL" AS "Profile_URL",
               "Users"."Lang" AS "Lang",
               "Users"."Created_At" AS "Created_At",
               "Users"."Friends_Count" AS "Friends_Count",
               "Users"."Followers_Count" AS "Followers_Count",
               "Users"."Favourites_Count" AS "Favourites_Count",
               "Users"."Statuses_Count" AS "Statuses_Count",
               "Users"."Time_Zone" AS "Time_Zone",
               "Users"."Following" AS "Following",
               "Users"."Contributors_Enabled" AS "Contributors_Enabled",
               "Users"."Follow_Request_Sent" AS "Follow_Request_Sent",
               "Users"."Listed_Count" AS "Listed_Count",
               "Users"."Description" AS "Description",
               "Users"."Default_Profile" AS "Default_Profile"
       FROM
               "Twitter"."Users" "Users"
) END_QUERY
```

Configuring ACCESSDATA to work with multiple password definitions

To configure the ACCESSDATA command to work with multiple password definitions, you insert PASSWORD parameters in the command and copy password key names from the SOURCE parameter to the PASSWORD parameters.

1. In Analytics, use the Data Access window to import data from a data source that requires more than one authentication value.
2. Copy the `ACCESSDATA` command from the log to an open script in the Script Editor.

   Typically, only one authentication value is masked (`[$pwd]`) in the `SOURCE` parameter, and additional values appear in clear text. For example:

   ```
   SOURCE( oauthclientid=cXQ ... dR4;oauthclientsecret=QUt ... beo;oau-
   thaccesstoken=913 ... cPn;oauthaccesstokensecret=[$pwd]; ... )
   ```

3. Delete the clear text authentication values from the `SOURCE` parameter and leave only the password key names and the equals sign.

   For example:

   ```
   SOURCE( oauthcli-
   entid=;oauthclientsecret=;oauthaccesstoken=;oauthaccesstokensecret=
   [$pwd]; ... )
   ```

   > **Note**
   > Authentication values must now be supplied by users via password definitions created with the PASSWORD command or the PASSWORD analytic tag. For more information, see "Creating a password definition" on page 1550.

4. Optional. Delete `[$pwd]` from the one password key name with a masked authentication value.

   With this password key name you can use either of the two methods for specifying a password definition in the ACCESSDATA command. For more information, see "Two options for specifying password definitions" on page 1550.

5. Delete the `PROMPT_PASSWORD` parameter from the `ACCESSDATA` command.
6. Insert numbered `PASSWORD` parameters at the location where you deleted `PROMPT_PASSWORD` and copy and paste the password key names from the `SOURCE` parameter to the `PASSWORD` parameters.

   For example:

   ```
   ACCESSDATA64 CONNECTOR NAME "Twitter" PASSWORD 1 AS oauthclientid
   PASSWORD 2 AS oauthclientsecret PASSWORD 3 AS oauthaccesstoken PASSWORD
   4 AS oauthaccesstokensecret ...
    SOURCE( oauthcli-
   entid=;oauthclientsecret=;oauthaccesstoken=;oauthaccesstokensecret=; ...
   )
   ```

   > **Important**
   > Password key names must be exactly identical between the SOURCE parameter and the PASSWORD parameters. If they are not, the ACCESSDATA command fails.

7. If you did not remove `[$pwd]` from the password key name with a masked authentication value, use the method for specifying a single password definition.

   For example:

```
ACCESSDATA64 CONNECTOR NAME "Twitter" PASSWORD 1 AS oauthclientid
PASSWORD 2 AS oauthclientsecret PASSWORD 3 AS oauthaccesstoken PASSWORD
4 ...
 SOURCE( oauthcli-
entid=;oauthclientsecret=;oauthaccesstoken=;oauthaccesstokensecret=
[$pwd]; ... )
```

Result – The ACCESSDATA command, in conjunction with separately created password definitions, can now be used in an import script without displaying clear text authentication values in the SOURCE connection settings.

Multi-factor authentication is not suitable

You cannot use the ACCESSDATA command to access data sources that require multi-factor authentication (MFA) since scripts cannot authenticate this way. If you need to access data protected with MFA, see if your organization will permit the use of a generic worker account that does not require MFA.

Data connector updates

When you upgrade Analytics, the Robots Agent, or AX Server, you should test any of your scripts that import data using one of the Analytics data connectors (ACCESSDATA command).

The possibility exists that changes made by third-party data sources or ODBC driver vendors required updates to one or more of the data connectors. Scripted data connections may need to be updated in order to continue working correctly.

- **Re-run the import** – The easiest way to update a connection is to manually perform an import using the Data Access window in the upgraded version of Analytics. Copy the ACCESSDATA command from the log and use it to update your script.

  > **Note**
  > Before connecting to a data source and re-running the import, clear the connector cache to flush the existing set of table names.
  > In the **Existing Connections** tab in the Data Access window, beside the connector name, select ··· **> Clear cache**.

- **Update field specifications** – You may also need to update field specifications in the script body to align with table schema changes in the data source or ODBC driver. Possible changes include field names, field data types, and field and record lengths.
- **Check the results of any filtering** – You should also check the results of any filtering that you apply as part of the data import. Confirm that the import filtering is including and excluding records correctly.

Creating ODBC connection settings and SQL import statements

ODBC connection settings, and SQL import statements, are often quite lengthy and involved, as shown in the examples.

The easiest way to create these portions of the ACCESSDATA command is to first use the Data Access window in Analytics to connect to the target data source, and import data. You can then copy the entire ACCESSDATA command from the log, including the connection settings and import statement, and customize the command in any way you require.

ACCESSDATA log files

Two log files record the transactions associated with the ACCESSDATA command, and can be used for troubleshooting if a data connection fails:

- **ServerDataAccess.log** - records all activities and errors prior to importing the data

  Location: `C:\Users\<user account>\AppData\Local\ACL\ACL for Windows\Data Access\ServerDataAccess.log`

  > **Note**
  > The "Server" in `ServerDataAccess.log` refers to the data access component of Analytics running locally on the computer where Analytics is installed.

- **DataAccess.log** - records information about the import operation and the Analytics project that you are importing data to

  Location: `..\<Analytics project folder>\DataAccess.log`

Comparing data hashed with ACCESSDATA to data hashed with the ACLScript HASH( ) function

Even though you cannot read the raw values of hashed data, it is still useful when combining or analyzing data.

If you want to compare values that are hashed by ACCESSDATA during import with values that are hashed using ACLScript's HASH( ) function, you must convert any numeric or datetime Analytics fields to character values and trim all leading and trailing spaces before hashing the data.

Datetime fields must use the following formats when converted to character:

- **Datetime** - "YYYY-MM-DD hh:mm:ss"
- **Date** - "YYYY-MM-DD"
- **Time** - "hh:mm:ss"

The following example uses the STRING( ) and ALLTRIM( ) functions to convert a numeric credit card number field to character data before hashing the value using ACLScript's HASH( ) function:

```
COMMENT ACL HASH function used after importing data
HASH(ALLTRIM(STRING(CC_No, 16)), "QZ3x7")
```

Once you hash the Analytics values, you can compare them with values hashed as part of the ACCESSDATA command import.

ACTIVATE command

Adds field definitions stored in an Analytics workspace to the existing set of field definitions in an Analytics table layout.

Syntax

```
ACTIVATE <WORKSPACE> workspace_name <OK>
```

Parameters

Name	Description
WORKSPACE *workspace_name*	The name of the workspace to activate.
OK optional	Deletes or overwrites items without asking you to confirm the action. If there is a field in the table with an identical name to one in the activated workspace, it will be overwritten without confirmation. You cannot replace any field that is referenced by a computed field.

Examples

Activating a workspace in your Analytics project

You activate the **ComplexFormulas** workspace:

```
ACTIVATE WORKSPACE ComplexFormulas OK
```

Activating a workspace saved as a file (.wsp) in the same folder as your Analytics project

You activate the **ComplexFormulas** workspace that was saved to a .wsp file:

```
ACTIVATE WORKSPACE ComplexFormulas.WSP OK
```

Remarks

How it works

ACTIVATE makes workspace field definitions available to the active table. Once you activate a workspace, its fields remain available for use with the active table until you close the table.

Editing table layouts

The workspace fields are permanently added to the table layout if:

- you edit the table layout after you activate a workspace
- you make a change that causes the table layout to be saved

Once the workspace fields are saved in the table layout, you can:

1. Use the DEFINE COLUMN command to add the fields to a view.
2. Use the SAVE command to save your changes.

AGE command

Groups records into aging periods based on values in a date or datetime field. Counts the number of records in each period, and also subtotals specified numeric fields for each period.

Syntax

```
AGE <ON> date_field <CUTOFF cutoff_date> <INTERVAL days <,...n>> <SUPPRESS>
<SUBTOTAL numeric_field <...n>|SUBTOTAL ALL <EXCLUDE numeric_field <...n>>>
<IF test> <WHILE test> <FIRST range|NEXT range> <TO
{SCREEN|filename|GRAPH|PRINT}> <KEY break_field> <HEADER header_text>
<FOOTER footer_text> <APPEND> <STATISTICS>
```

Parameters

Name	Description
ON *date_field*	The name of the date or datetime field or the expression to be aged. Although you can age on a datetime field, only the date portion of datetime values is considered. The time portion is ignored. You cannot age on time data alone.
CUTOFF *cutoff_date* optional	The date that values in *date_field* are compared against. You must specify *cutoff_date* as an unquoted string in YYMMDD or YYYYMMDD format, regardless of the format of the date field. For example: `CUTOFF 20141231` If you omit CUTOFF, the current system date is used as the cutoff date.
INTERVAL *days <,...n>* optional	The date intervals (that is, number of days) to use in calculating the aging periods. *days* represents the beginning of each aging period measured backward in time from *cutoff_date*: ◦ the first *days* value identifies the beginning of the first aging period ◦ a first *days* value of '0' specifies that the first aging period begins on the specified *cutoff_date* ◦ the last *days* value identifies the end of the final aging period You must specify the intervals as an unquoted string with comma separated values: `INTERVAL 0,90,180,270,365`

Name	Description
	The default aging periods are 0, 30, 60, 90, 120, and 10,000 days. An interval of 10,000 days is used to isolate records with dates that are probably invalid. If required, date intervals can be customized to mirror other internal aging reports.
SUPPRESS optional	Suppresses dates that fall outside the aging period from the command output.
SUBTOTAL *numeric_field <...n>* | SUBTOTAL ALL optional	One or more numeric fields or expressions to subtotal for each group. Multiple fields must be separated by spaces. Specify ALL to subtotal all the numeric fields in the table.
EXCLUDE *numeric_field* optional	Only valid when using SUBTOTAL ALL. The field or fields to exclude from the command. EXCLUDE allows you to fine-tune SUBTOTAL ALL, by excluding the specified fields. EXCLUDE must immediately follow SUBTOTAL ALL. For example: `SUBTOTAL ALL EXCLUDE field_1 field_2`
IF *test* optional	A conditional expression that must be true in order to process each record. The command is executed on only those records that satisfy the condition. **Note** The IF parameter is evaluated against only the records remaining in a table after any scope parameters have been applied (WHILE, FIRST, NEXT).
WHILE *test* optional	A conditional expression that must be true in order to process each record. The command is executed until the condition evaluates as false, or the end of the table is reached. **Note** If you use WHILE in conjunction with FIRST or NEXT, record processing stops as soon as one limit is reached.
FIRST *range* | NEXT *range* optional	The number of records to process: ◦ **FIRST** - start processing from the first record until the specified number of records is reached ◦ **NEXT** - start processing from the currently selected record until the specified number of records is reached Use *range* to specify the number of records to process. If you omit FIRST and NEXT, all records are processed by default.
TO SCREEN | *filename* | GRAPH | PRINT	The location to send the results of the command to: ◦ **SCREEN** - displays the results in the Analytics display area

Name	Description
	Tip You can click any linked result value in the display area to drill down to the associated record or records in the source table. ◦ ***filename*** - saves the results to a file Specify *filename* as a quoted string with the appropriate file extension. For example: `TO "Output.TXT"` By default, the file is saved to the folder containing the Analytics project. Use either an absolute or relative file path to save the file to a different, existing folder: • `TO "C:\Output.TXT"` • `TO "Results\Output.TXT"` ◦ **GRAPH** - displays the results in a graph in the Analytics display area ◦ **PRINT** - sends the results to the default printer
KEY *break_field* optional	The field or expression that groups subtotal calculations. A subtotal is calculated each time the value of *break_field* changes. *break_field* must be a character field or expression. You can specify only one field, but you can use an expression that contains more than one field.
HEADER *header_text* optional	The text to insert at the top of each page of a report. *header_text* must be specified as a quoted string. The value overrides the Analytics HEADER system variable.
FOOTER *footer_text* optional	The text to insert at the bottom of each page of a report. *footer_text* must be specified as a quoted string. The value overrides the Analytics FOOTER system variable.
APPEND optional	Appends the command output to the end of an existing file instead of overwriting it. **Note** You must ensure that the structure of the command output and the existing file are identical: • the same fields • the same field order • matching fields are the same length • matching fields are the same data type Analytics appends output to an existing file regardless of its structure. If the structure of the output and the existing file do not match, jumbled, missing, or inaccurate data can result.
STATISTICS optional	**Note** Cannot be used unless SUBTOTAL is also specified. Calculates average, minimum, and maximum values for all SUBTOTAL fields.

Examples

Age invoices with subtotaled amounts

You want to age an accounts receivable table on the **Invoice_Date** field and subtotal the **Invoice_Amount** field.

Invoices are grouped into 30-day periods:

- from the cutoff date to 29 days previous
- from 30 days previous to 59 days previous
- so on

The results include the total outstanding invoice amount for each period:

```
OPEN Ar
AGE ON Invoice_Date CUTOFF 20141231 INTERVAL 0,30,60,90,120,10000
SUBTOTAL Invoice_Amount TO SCREEN
```

Remarks

For more information about how this command works, see "Aging data" on page 1253.

Aging periods

The AGE command groups records into aging periods based on values in a date or datetime field. The output results contain a single record for each period, with a count of the number of records in the source table that fall into each period.

Interval measurement

Aging periods are based on date intervals (that is, number of days) measured backward in time from the current system date, or from a cutoff date you specify such as a fiscal period end date.

Future periods

You can create aging periods more recent than the cutoff date by entering negative values for date intervals. For example, the following creates aging periods running forward and backward from the cutoff date:

```
INTERVAL -60,-30,0,30,60,90
```

This approach creates a date profile of all the records in a table using different points in time.

Common use cases

Common uses of aging include evaluating sales trends, looking at transaction volumes, and grouping invoices by the number of days outstanding.

Analytics automatically creates one or two additional aging periods for any dates that fall outside the specified aging periods, assuming you are not using SUPPRESS.

APPEND command

Combines records from two or more Analytics tables by appending them in a new Analytics table.

Syntax

```
APPEND table_1, table_2, <...n> TO table_name <COMMONFIELDS> <OPEN> <ASCHAR>
<ALLCHAR> <SOURCETABLE>
```

Parameters

Name	Description
table_1, *table_2*, *<...n>*	The tables to append. The records from each table are appended in the order in which you specify the tables. The output table contains the records from *table_1*, followed by the records from *table_2*, and so on. The source tables can have different or identical record structures, and can be sorted or unsorted.
TO *table_name*	The location to send the results of the command to: ◦ ***table_name*** - saves the results to an Analytics table Specify *table_name* as a quoted string with a .FIL file extension. For example: `TO "Output.FIL"` By default, the table data file (.FIL) is saved to the folder containing the Analytics project. Use either an absolute or relative file path to save the data file to a different, existing folder: • `TO "C:\Output.FIL"` • `TO "Results\Output.FIL"` **Note** Table names are limited to 64 alphanumeric characters, not including the .FIL extension. The name can include the underscore character (_), but no other special characters, or any spaces. The name cannot start with a number.
COMMONFIELDS	Only those fields that are common to all tables being appended are included in the output table.

Name	Description
optional	If you omit COMMONFIELDS, all fields from all tables are included in the output table. Blank values appear in the output table where no fields exist in the source tables. **Tip** For diagrams and screen captures illustrating the two options, see "Appending tables" on page 858. **Note** The APPEND command does not support appending computed fields. For more information, see "Computed fields not supported" on page 1570. ## What makes fields common? For fields to be considered common they must: ◦ occur in every source table ◦ have an identical physical name ◦ belong to the same data category: • Character • Numeric • Datetime • Logical ## Identical name, different data category If two fields have an identical name but belong to different data categories, an error message appears and the APPEND command is not executed. The error message contains all data category conflicts in the set of tables specified by APPEND. The message is saved to the command log. **Note** You can avoid this situation by using either ASCHAR or ALLCHAR to harmonize data categories.
OPEN optional	Opens the table created by the command after the command executes. Only valid if the command creates an output table.
ASCHAR optional	Harmonizes fields with identical names but different data categories by converting non-character fields to the character data category. For example, you append two tables in which the Employee_ID field is character data in one table, and numeric data in the other. The numeric Employee_ID field is converted to character data and the two fields are appended without an error. ASCHAR is ignored if ALLCHAR is also specified.
ALLCHAR optional	Converts all non-character fields in all tables being appended to the character data category. This global conversion to character data ensures that all identically named fields

Name	Description
	are appended without error. **Note** After appending, you can change the data category of an entire appended field if appropriate for the data contained by the field.
SOURCETABLE optional	Include the **Source Table** field (`Source_Table`) in the output table. For each record in the output table, the **Source Table** field identifies the table from which the record originated. **Tip** Including the names of the source tables you are appending may provide useful information when you analyze data in the output table.

Examples

Append three monthly transaction tables

The example below appends three monthly transaction tables and outputs a quarterly transaction table that includes all fields from the three source tables:

```
APPEND Trans_Jan, Trans_Feb, Trans_Mar TO Trans_Q1
```

Append three employee tables and include only common fields

The example below appends three divisional employee tables and outputs a master employee table that includes only common fields from the three source tables:

```
APPEND Employees_central, Employees_east, Employees_west TO Employees_
master COMMONFIELDS
```

Append three employee tables and harmonize fields with different data categories

The examples below append three divisional employee tables in which some identically named fields use different data categories.

The first example converts non-character fields to the character data category only when required for harmonization:

```
APPEND Employees_central, Employees_east, Employees_west TO Employees_
master ASCHAR
```

The second example converts all non-character fields to the character data category whether required for harmonization or not:

```
APPEND Employees_central, Employees_east, Employees_west TO Employees_
master ALLCHAR
```

Remarks

For more information about how this command works, see "Appending tables" on page 858.

How it works

The APPEND command combines records from two or more tables by appending them and creating a new table. Appending means to add one group of records to the bottom of another group of records.

Source table fields with identical physical names and identical data categories are directly appended to one another.

Fields with physical names that are unique across all the source tables are added to the output table but not directly appended to any other field.

> **Tip**
> If you want to directly append inconsistently named fields, standardize the physical names of the fields in the table layouts before appending. (Assumes that the fields belong to the same data category, or that you use ASCHAR or ALLCHAR to harmonize the data category of the fields.)

When to use APPEND

Use APPEND when you want to combine data from multiple tables with an identical or similar structure. For example, APPEND is a good choice for combining monthly or quarterly tables into an annual table.

> **Tip**
> A single execution of the APPEND command can replace multiple executions of the EXTRACT command with the APPEND option.

Not a substitute for JOIN or DEFINE RELATION

APPEND is generally not a substitute for the JOIN or DEFINE RELATION commands because it does not allow you to include or exclude records based on matched or unmatched values in a common key field. With APPEND, all records from each source table are included in the output table.

Appending completely dissimilar tables

You can append completely dissimilar tables – that is, two or more tables that do not have any fields in common. While not the primary intended use of the APPEND command, there may be instances in which appending dissimilar tables serves an analytical purpose.

Appending datetime fields

For two or more datetime fields to be appended, the following conditions must be met:

- identical physical names
- identical data category (Datetime)
- identical data subtypes (date, datetime, or time)
- identical use of time zone indicator – either used, or not used, by all fields being appended

If two datetime fields have an identical name but fail to meet one of the other conditions an error message appears and the APPEND command is not executed.

The error message contains all failed conditions in the set of tables specified by APPEND. The message is saved to the command log.

> **Note**
> You can harmonize dissimilar datetime fields by converting them to the character data category, and then append them. This approach allows you to combine the data in a single table. However, depending on the nature of the source data, you may not be able to convert the combined data back to datetime data.

Automatic harmonization

In some situations the APPEND command automatically harmonizes fields in order to append them:

Data category of fields	Harmonization performed
Character	◦ Different field lengths are harmonized. ◦ Different character data types such as Custom, PCASCII, and EBCDIC are harmonized by converting the fields to the ASCII or UNICODE data type.
Numeric	◦ Different field lengths are harmonized. The fields are converted to the ACL data type. ◦ A different number of defined decimal places are harmonized. Decimal places are standardized on the greatest number of places, with trailing zeros added to numeric values where necessary. The fields are converted to the ACL data type. ◦ Different numeric data types such as Print, Float, EBCDIC, and Micro are harmonized by converting the fields to the ACL data type.
Datetime	◦ Different date, datetime, or time formats in the source data are harmonized by converting the fields to the Analytics default formats: • YYYYMMDD • YYYYMMDD hh:mm:ss • hh:mm:ss

When automatic harmonization is not performed

Analytics does not automatically harmonize fields in the following situations. An error message appears and the append operation is not executed.

- Two fields with an identical name belong to different data categories.
- Two datetime fields with an identical name belong to different datetime subtypes (date, datetime, or time).
- Two datetime fields with an identical name are inconsistent in their use of the time zone indicator.

> **Note**
>
> User-specified harmonization of fields with identical names but different data categories is explained above. For more information, see "ASCHAR" on page 1566 and "ALLCHAR" on page 1566.

Computed fields not supported

The APPEND command does not support appending computed fields. When you append tables, any computed fields in the source tables are automatically excluded from the output table.

If a computed field in a source table has the same name as a physical field in another source table, an error message appears and the APPEND command is not executed.

> **Tip**
> You can append a computed field by first extracting it to convert the field to a physical field. (For more information, see "EXTRACT command" on page 1712.) You then use the extracted table in the append operation.
>
> Another approach is to recreate the computed field in the appended output table.

Record Note fields not supported

The APPEND command does not support appending Record Note fields. When you append tables, any Record Note fields in the source tables are automatically excluded from the output table.

If a Record Note field in a source table has the same name as a physical field in another source table, an error message appears and the APPEND command is not executed.

A Record Note field is automatically generated by Analytics when you add a note to a record.

Record length

If you include all fields from all source tables when appending, the record length in the output table can be longer than the longest record in the source tables.

An error message appears if the output record length exceeds the Analytics maximum of 32 KB.

Appending and decimal places

Specific behavior governs the appending of numeric fields that include decimal places.

The Decimal setting

The APPEND command uses the number of decimal places defined in the **Dec** setting in the field definition in the table layout.

> **Note**
> The **Dec** setting may not be the same as the actual number of decimal places in the source data. Decimal places that exceed the **Dec** setting are undefined, and are rounded in calculations.

Inconsistent Decimal settings

If appended numeric fields have inconsistent **Dec** settings, the fields are converted to the ACL data type and automatically harmonized on the longest **Dec** setting.

Any decimal places in source data files that exceed the longest **Dec** setting are **excluded** from the output table generated by APPEND.

Consistent Decimal setting

If appended numeric fields have a consistent **Dec** setting, no data type conversion or harmonization occurs.

Any decimal places in source data files that exceed the **Dec** setting are **included** in the output table generated by APPEND.

Sorting

Any existing sort orders in the source tables are separately maintained in the respective record sets in the output table.

Even if the records in all source tables are sorted, the output table is considered unsorted because the source records are appended as groups, without consideration of any existing sort order in other source tables.

For example, if you append monthly or quarterly tables to create an annual table, any internal sorting of the monthly or quarterly data is retained. If required, you can sort the output table after performing the append operation.

How field order works

Common fields

Common fields in source tables do not have to be in the same order to be appended.

For example, these fields are correctly appended even though they are in a different order:

Table	Fields
table_1	Last_name | First_name | Middle_name
table_2	First_name | Middle_name | Last_name

The first table specified in the APPEND command dictates the order of the fields in the output table. So in the example above, the order in the output table is:

- Last_name | First_name | Middle_name

Non-common fields

Non-common fields in source tables appear in the output table in the order that they appear in the selected group of source tables.

For example, when appending these two tables:

Table	Fields
table_1	Title | Last_name | First_name | Middle_name
table_2	First_name | Middle_name | Last_name | Date_of_birth

the order in the output table is:

- Title | Last_name | First_name | Middle_name | Date_of_birth

Alternate Column Title

Alternate column titles in source tables appear in the output table. If more than one source table has an alternate column title for the same field, the title from the first selected table takes precedence.

ASSIGN command

Creates a variable and assigns a value to the variable.

Syntax

```
ASSIGN variable_name = value <IF test>
```

> **Tip**
> You can omit the ASSIGN keyword because Analytics automatically interprets the following syntax as an assignment operation:
>
> ```
> variable_name = value
> ```

Parameters

Name	Description
variable_name	The name of the variable to assign the value to. If the variable does not exist, it is created. If the variable already exists, it is updated with the new value. Do not use non-English characters, such as é, in the names of variables. Variable names that contain non-English characters will cause scripts to fail. **Note** Variable names are limited to 31 alphanumeric characters. The name can include the underscore character (_), but no other special characters, or any spaces. The name cannot start with a number.
value	The value to assign to the variable. If a new variable is created, the variable type is based on the data type in *value*.
IF *test* Optional	A conditional expression that must be true to create the variable or assign the value to the variable.

Examples

Assigning a value to a variable

You assign the value of the **Amount** field in the current record to a variable named *v_current_amount*. Because *v_current_amount* is a variable, its value does not change unless explicitly changed by another ASSIGN command:

```
ASSIGN v_current_amount = Amount
```

Conditionally assigning a value to a variable

You want to update the value of a variable called *v_quantity* to 1, but only if the value in another variable called *v_counter* is less than 10.

If *v_counter* is greater than or equal to 10, no assignment is made and the value of *v_quantity* remains unchanged.

Note that the optional ASSIGN keyword is omitted:

```
v_quantity = 1 IF v_counter < 10
```

Remarks

Duration of variables

Variables with names that are not prefaced with an underscore are retained for the duration of the current Analytics session only.

If you want a variable to be permanently saved with an Analytics project, preface the variable name with an underscore:

```
ASSIGN value = _variable_name
```

Reassigning variables used in a computed field or GROUP

If you assign a value to an existing variable in the following situations, then the new value is assigned but the previous value's length and decimal count are retained:

- variables used in computed fields
- variables reassigned inside a GROUP

The length of the new value is padded or truncated and the decimals are adjusted if required.

If you reassign a variable in any other context, then the previous value as well as its length and decimal specifications are overwritten.

Variables created by Analytics commands

When you execute certain commands, either by entering information in dialog boxes in Analytics or by running scripts, system variables are automatically created by Analytics. You can use these variables, and the values they contain, when processing subsequent Analytics commands.

The value in a system variable is replaced with an updated value if you execute the same command again.

For more information, see "Variables created by Analytics commands" on page 1366.

BENFORD command

Counts the number of times each leading digit (1-9) or leading digit combination occurs in a field, and compares the actual count to the expected count. The expected count is calculated using the Benford formula.

Syntax

```
BENFORD <ON> numeric_field <LEADING n> <IF test> <BOUNDS> <TO {SCREEN|table_
name|GRAPH|PRINT}> <LOCAL> <HEADER header_text> <FOOTER footer_text>
<WHILE test> <FIRST range|NEXT range> <APPEND> <OPEN>
```

Parameters

Name	Description
ON *numeric_field*	The numeric field to analyze. **Note** Select a field that contains "naturally occurring numbers", such as transaction amounts. Benford analysis is not suitable for numeric data that is constrained in any way. For more information, see "What data can I test using Benford analysis?" on page 1580
LEADING *n* optional	The number of leading digits to analyze. The value of *n* must be from 1 to 6. If LEADING is omitted, the default value of 1 is used.
IF *test* optional	A conditional expression that must be true in order to process each record. The command is executed on only those records that satisfy the condition. **Note** The IF parameter is evaluated against only the records remaining in a table after any scope parameters have been applied (WHILE, FIRST, NEXT).
BOUNDS optional	Includes computed upper and lower bound values in the output results. If two or more counts in the output results exceed either of the bounds, the data may have been manipulated and should be investigated.

Name	Description
TO SCREEN | *table_name* | GRAPH | PRINT optional	The location to send the results of the command to: ◦ **SCREEN** - displays the results in the Analytics display area **Tip** You can click any linked result value in the display area to drill down to the associated record or records in the source table. ◦ ***table_name*** - saves the results to an Analytics table Specify *table_name* as a quoted string with a .FIL file extension. For example: `TO "Output.FIL"` By default, the table data file (.FIL) is saved to the folder containing the Analytics project. Use either an absolute or relative file path to save the data file to a different, existing folder: • `TO "C:\Output.FIL"` • `TO "Results\Output.FIL"` **Note** Table names are limited to 64 alphanumeric characters, not including the .FIL extension. The name can include the underscore character (_), but no other special characters, or any spaces. The name cannot start with a number. ◦ **GRAPH** - displays the results in a graph in the Analytics display area ◦ **PRINT** - sends the results to the default printer
LOCAL optional	Saves the output file in the same location as the Analytics project. **Note** Applicable only when running the command against a server table with an output file that is an Analytics table. The LOCAL parameter must immediately follow the TO parameter.
HEADER *header_text* optional	The text to insert at the top of each page of a report. *header_text* must be specified as a quoted string. The value overrides the Analytics HEADER system variable.
FOOTER *footer_text* optional	The text to insert at the bottom of each page of a report. *footer_text* must be specified as a quoted string. The value overrides the Analytics FOOTER system variable.
WHILE *test* optional	A conditional expression that must be true in order to process each record. The command is executed until the condition evaluates as false, or the end of the table is reached.

Name	Description
	Note If you use WHILE in conjunction with FIRST or NEXT, record processing stops as soon as one limit is reached.
FIRST *range* | NEXT *range* optional	The number of records to process: ◦ **FIRST** - start processing from the first record until the specified number of records is reached ◦ **NEXT** - start processing from the currently selected record until the specified number of records is reached Use *range* to specify the number of records to process. If you omit FIRST and NEXT, all records are processed by default.
APPEND optional	Appends the command output to the end of an existing file instead of overwriting it. **Note** You must ensure that the structure of the command output and the existing file are identical: • the same fields • the same field order • matching fields are the same length • matching fields are the same data type Analytics appends output to an existing file regardless of its structure. If the structure of the output and the existing file do not match, jumbled, missing, or inaccurate data can result.
OPEN optional	Opens the table created by the command after the command executes. Only valid if the command creates an output table.

Examples

Outputting results to graph

You run the BENFORD command against the **Amount** field and output the results to a graph:

```
BENFORD ON Amount LEADING 2 BOUNDS TO GRAPH
```

Remarks

For more information about how this command works, see "Performing Benford analysis" on page 1316.

What data can I test using Benford analysis?

You should only use Benford analysis for testing numeric data composed of "naturally occurring numbers", such as accounting amounts, transaction amounts, expenses, or address numbers. Benford analysis is not suitable for numeric data that is constrained in any way.

Follow these guidelines for identifying numeric data that is suitable for Benford analysis:

- **Size of the data set** – The data set must be large enough to support a valid distribution. Benford analysis may not give reliable results for fewer than 500 records.
- **Leading digit requirement** – All numbers from 1 to 9 must have the possibility of occurring as the leading digit.
- **Leading digit combination requirement** – All numbers from 0 to 9 must have the possibility of occurring as the second leading digit, and as any additional digits being analyzed.
- **Constrained data** – Numeric data that is assigned or generated according to a pre-ordained pattern is not suitable for Benford analysis. For example, do not use Benford to analyze:
  - sequential check or invoice numbers
  - social security numbers or telephone numbers that map to a specific pattern
  - any numbering scheme with a range that prevents certain numbers from appearing
- **Random numbers** – Numbers generated by a random number generator are not suitable for Benford analysis.

CALCULATE command

Calculates the value of one or more expressions.

Syntax

```
CALCULATE expression <AS result_label> <,...n>
```

Parameters

Name	Description
expression	The expression to calculate. The expression can be any of the four types: ◦ character ◦ numeric ◦ datetime ◦ logical Separate multiple expressions with a comma: `CALCULATE 4.7 * 18.5, 1 + 2, "a" + "b"`
AS *result_label* optional	The name of the result when displayed on screen and in the Analytics command log. *result_label* must be a quoted string or a valid character expression. If omitted, the expression being calculated is used as the result name.

Examples

Performing a simple calculation

You use CALCULATE to multiply 4.70 by 18.50, returning the result 86.95:

```
CALCULATE 4.70 * 18.50
```

Naming the results of a calculation

You use CALCULATE to derive the gross margin for the currently selected record using previously defined fields for sale price and unit cost:

```
CALCULATE Sale_price - Unit_cost AS "Margin"
```

The result is identified on screen, and in the log, as "Margin".

Remarks

How it works

CALCULATE provides the functionality of a calculator combined with access to Analytics functions, variables, and the data in the currently selected record.

Command output

Depending on where you run CALCULATE, the results are output to different locations:

- **From the command line** – the result is displayed on screen
- **From within a script** – the result is recorded in the log

The *result_label* value is not a variable that you can use in a script. It is only used to identify the calculation on screen or in the log.

Number of decimal places in output

In a numeric calculation, the result has as many decimal places as the expression component with the greatest number of decimal places.

Returns 1:

```
CALCULATE 365/52/7
```

Returns 1.0027:

```
CALCULATE 365.0000/52/7
```

Working with table input

If the expression contains a field value, the table the field belongs to must be open. You can use the FIND, SEEK, or LOCATE commands to move to the record to be analyzed by CALCULATE.

CLASSIFY command

Groups records based on identical values in a character or numeric field. Counts the number of records in each group, and also subtotals specified numeric fields for each group.

Syntax

```
CLASSIFY <ON> key_field <SUBTOTAL numeric_field <...n>|SUBTOTAL ALL <EXCLUDE
numeric_field <...n>>> <INTERVALS number> <SUPPRESS> <TO {SCREEN|table_
name|GRAPH|PRINT}> <LOCAL> <IF test> <WHILE test> <FIRST range|NEXT range>
<HEADER header_text> <FOOTER footer_text> <KEY break_field> <OPEN> <APPEND>
<STATISTICS>
```

Parameters

Name	Description
ON *key_field*	The character or numeric field to classify. Maximum key field length is 2048 characters. If you want to classify a table using a key field longer than 2048 characters, use the SUMMARIZE command. It does not restrict key field length.
SUBTOTAL *numeric_field <...n>* | SUBTOTAL ALL optional	One or more numeric fields or expressions to subtotal for each group. Multiple fields must be separated by spaces. Specify ALL to subtotal all the numeric fields in the table.
EXCLUDE *numeric_field* optional	Only valid when using SUBTOTAL ALL. The field or fields to exclude from the command. EXCLUDE allows you to fine-tune SUBTOTAL ALL, by excluding the specified fields. EXCLUDE must immediately follow SUBTOTAL ALL. For example: `SUBTOTAL ALL EXCLUDE field_1 field_2`
INTERVALS *number* optional	The maximum number of groups in the output result. If the number of sets of identical values in the field being classified exceeds the specified maximum, sets are used starting from the top of the column.

Name	Description
	Sets exceeding the maximum are grouped together in a group called "OTHER". If INTERVALS is omitted, a group is created for each set of identical values in the field being classified. **Note** This parameter is not available in the Analytics user interface and can only be used as part of ACLScript syntax in a script or the command line.
SUPPRESS optional	**Note** Cannot be used unless INTERVALS is also specified. SUPPRESS is not available in the Analytics user interface and can only be used as part of ACLScript syntax in a script or the command line. Excludes sets of identical values exceeding the maximum specified by INTERVALS from the command output.
TO SCREEN | *table_name* | GRAPH | PRINT	The location to send the results of the command to: ◦ **SCREEN** - displays the results in the Analytics display area **Tip** You can click any linked result value in the display area to drill down to the associated record or records in the source table. ◦ ***table_name*** - saves the results to an Analytics table Specify *table_name* as a quoted string with a .FIL file extension. For example: `TO "Output.FIL"` By default, the table data file (.FIL) is saved to the folder containing the Analytics project. Use either an absolute or relative file path to save the data file to a different, existing folder: • `TO "C:\Output.FIL"` • `TO "Results\Output.FIL"` **Note** Table names are limited to 64 alphanumeric characters, not including the .FIL extension. The name can include the underscore character (_), but no other special characters, or any spaces. The name cannot start with a number. ◦ **GRAPH** - displays the results in a graph in the Analytics display area ◦ **PRINT** - sends the results to the default printer
LOCAL optional	Saves the output file in the same location as the Analytics project.

Name	Description
	Note Applicable only when running the command against a server table with an output file that is an Analytics table. The LOCAL parameter must immediately follow the TO parameter.
IF *test* optional	A conditional expression that must be true in order to process each record. The command is executed on only those records that satisfy the condition. **Note** The IF parameter is evaluated against only the records remaining in a table after any scope parameters have been applied (WHILE, FIRST, NEXT).
WHILE *test* optional	A conditional expression that must be true in order to process each record. The command is executed until the condition evaluates as false, or the end of the table is reached. **Note** If you use WHILE in conjunction with FIRST or NEXT, record processing stops as soon as one limit is reached.
FIRST *range* | NEXT *range* optional	The number of records to process: ◦ **FIRST** - start processing from the first record until the specified number of records is reached ◦ **NEXT** - start processing from the currently selected record until the specified number of records is reached Use *range* to specify the number of records to process. If you omit FIRST and NEXT, all records are processed by default.
HEADER *header_text* optional	The text to insert at the top of each page of a report. *header_text* must be specified as a quoted string. The value overrides the Analytics HEADER system variable.
FOOTER *footer_text* optional	The text to insert at the bottom of each page of a report. *footer_text* must be specified as a quoted string. The value overrides the Analytics FOOTER system variable.
KEY *break_field* optional	The field or expression that groups subtotal calculations. A subtotal is calculated each time the value of *break_field* changes. *break_field* must be a character field or expression. You can specify only one field, but you can use an expression that contains more than one field.
OPEN optional	Opens the table created by the command after the command executes. Only valid if the command creates an output table.
APPEND optional	Appends the command output to the end of an existing file instead of overwriting it.

Name	Description
	Note You must ensure that the structure of the command output and the existing file are identical: • the same fields • the same field order • matching fields are the same length • matching fields are the same data type Analytics appends output to an existing file regardless of its structure. If the structure of the output and the existing file do not match, jumbled, missing, or inaccurate data can result.
STATISTICS optional	**Note** Cannot be used unless SUBTOTAL is also specified. Calculates average, minimum, and maximum values for all SUBTOTAL fields.

Examples

Total transaction amount per customer

You want to classify an accounts receivable table on the **Customer_Number** field and subtotal the **Trans_Amount** field. The output results are grouped by customer, and include the total transaction amount for each customer:

```
OPEN Ar
CLASSIFY ON Customer_Number SUBTOTAL Trans_Amount TO "Customer_
total.FIL"
```

Total, average, minimum, and maximum transaction amounts per customer

As with the previous example, you classify an accounts receivable table on the **Customer_Number** field and subtotal the **Trans_Amount** field.

Now you include STATISTICS to calculate the average, minimum, and maximum transaction amounts for each customer:

```
OPEN Ar
CLASSIFY ON Customer_Number SUBTOTAL Trans_Amount TO "Customer_stat-
s.FIL" STATISTICS
```

Identical invoice amounts

You need to identify invoice amounts that appear more than once in the **Ap_Trans** table.

To do this, you classify the table on the **Invoice_Amount** field. The output results are grouped by invoice amount with an associated count that you can use to identify any invoice amounts that occur more than once:

```
OPEN Ap_Trans
CLASSIFY ON Invoice_Amount TO "Grouped_invoice_amounts.FIL" OPEN
SET FILTER TO COUNT > 1
```

Remarks

For more information about how this command works, see "Classifying data" on page 1261.

How it works

CLASSIFY groups records that have the same value in a character or numeric field.

Output contains a single record for each group, with a count of the number of records in the source table that belong to the group.

Sorting and CLASSIFY

CLASSIFY can process either sorted or unsorted data. Output is automatically sorted in ascending order.

Names of auto-generated subtotal and statistics fields

If you use STATISTICS to perform statistical calculations on one or more SUBTOTAL fields, and output the results to an Analytics table, the fields auto-generated by the parameters have the following names:

Description of auto-generated field	Field name in output table	Alternate column title (display name) in output table
Subtotal field	*subtotaled field name in source table*	**Total** + *subtotaled alternate column title in source table*
Average field	**a_***subtotaled field name in source table*	**Average** + *subtotaled alternate column title in source table*
Minimum field	**m_***subtotaled field name in source table*	**Minimum** + *subtotaled alternate column title in source table*
Maximum field	**x_***subtotaled field name in source table*	**Maximum** + *subtotaled alternate column title in source table*

CLOSE command

Closes an Analytics table, index file, or log file, or ends a **Script Recorder** session.

Syntax

```
CLOSE <table_name|PRIMARY|SECONDARY|INDEX|LOG|LEARN>
```

Parameters

Name	Description
table_name|PRIMARY|SECONDARY|INDEX|LOG|LEARN optional	The item to close: ◦ ***table_name*** - the name of the Analytics table to close ◦ **PRIMARY** - closes the primary Analytics table Using CLOSE without any parameters also closes the primary table. ◦ **SECONDARY** - closes the secondary Analytics table ◦ **INDEX** - closes the current index applied to the Analytics table ◦ **LOG** - returns the log file to the default command log, after the SET LOG command has been used to specify another log file ◦ **LEARN** - ends the active **Script Recorder** session and prompts you to save the script file the session was recorded to LEARN can be used in scripts, but its intended use is in the command line. The **Script Recorder** records the ACLScript syntax for commands that are executed using dialog boxes in the Analytics user interface.

Examples

Closing a table by name

You want to close a table called **Inventory**:

```
CLOSE Inventory
```

Closing a table by type

You want to close the current secondary table:

```
CLOSE SECONDARY
```

Restoring the default Analytics command log

You want to restore the default command log after using a separate log file to capture the data verification phase of a script:

```
SET LOG TO "DataVerificationPhase.log"
COMMENT Execute data verification commands
CLOSE LOG
```

Remarks

When to use CLOSE

You typically do not need to close Analytics tables. The active Analytics table automatically closes when you open another table. The primary table also closes automatically before the OPEN or QUIT commands execute.

You cannot use CLOSE to close an Analytics project. Use QUIT instead.

Related fields and tables

When you close a primary or secondary table, all related field definitions are removed from memory. Any changes to the table layout are saved before the table is closed.

If you have defined table relations in an Analytics project, the CLOSE command closes both the primary and any secondary tables. It also closes the related tables.

CLUSTER command

Groups records into clusters based on similar values in one or more numeric fields. Clusters can be uni-dimensional or multidimensional.

> **Note**
> The CLUSTER command is not supported if you are running Analytics on a 32-bit computer. The computation required by the command is processor-intensive and better suited to 64-bit computers.

Syntax

```
CLUSTER ON key_field <...n> KVALUE number_of_clusters ITERATIONS number_of_
iterations INITIALIZATIONS number_of_initializations <SEED seed_value> <OTHER
field < ...n>|OTHER ALL> TO table_name <IF test> <WHILE test> <FIRST
range|NEXT range> OPEN {no_keyword|NOCENTER|NOSCALE}
```

Parameters

Name	Description
ON *key_field* <...*n*>	One or more numeric fields to cluster. Multiple fields must be separated by spaces.
KVALUE *number_of_clusters*	The number of clusters generated in the output results. For more information, see "Choosing the number of clusters (K value)" on page 1311.
ITERATIONS *number_of_iterations*	The maximum number of times the cluster calculation is re-performed.
INITIALIZATIONS *number_of_initializations*	The number of times to generate an initial set of random centroids.
SEED *seed_value* optional	The seed value to use to initialize the random number generator in Analytics. If you omit SEED, Analytics randomly selects the seed value.
OTHER *field* <...*n*> | OTHER ALL	One or more additional fields to include in the output. ◦ **OTHER *field* <...*n*>** - include the specified field or fields

Name	Description
optional	Fields are included in the order that you list them. ◦ **OTHER ALL** – include all fields in the table Fields are included in the order that they appear in the table layout. **Note** Key fields are automatically included in the output table, although the values are scaled unless you specify NOSCALE. You can use OTHER to include a second, unscaled instance of a key field or fields.
TO *table_name*	The location to send the results of the command to: ◦ ***table_name*** – saves the results to an Analytics table Specify *table_name* as a quoted string with a .FIL file extension. For example: `TO "Output.FIL"` By default, the table data file (.FIL) is saved to the folder containing the Analytics project. Use either an absolute or relative file path to save the data file to a different, existing folder: • `TO "C:\Output.FIL"` • `TO "Results\Output.FIL"` **Note** Table names are limited to 64 alphanumeric characters, not including the .FIL extension. The name can include the underscore character (_), but no other special characters, or any spaces. The name cannot start with a number.
IF *test* optional	A conditional expression that must be true in order to process each record. The command is executed on only those records that satisfy the condition. **Note** The IF parameter is evaluated against only the records remaining in a table after any scope parameters have been applied (WHILE, FIRST, NEXT).
WHILE *test* optional	A conditional expression that must be true in order to process each record. The command is executed until the condition evaluates as false, or the end of the table is reached. **Note** If you use WHILE in conjunction with FIRST or NEXT, record processing stops as soon as one limit is reached.
FIRST *range* | NEXT *range* optional	The number of records to process: ◦ **FIRST** – start processing from the first record until the specified number of records is reached ◦ **NEXT** – start processing from the currently selected record until the specified

Name	Description
	number of records is reached Use *range* to specify the number of records to process. If you omit FIRST and NEXT, all records are processed by default.
OPEN optional	Opens the table created by the command after the command executes. Only valid if the command creates an output table.
no_keyword | NOCENTER | NOSCALE	The method for preprocessing key field numeric values before calculating the clusters. ◦ ***no_keyword*** - center key field values on a mean of zero (0), and scale them by dividing by their standard deviation, a process that converts the values to their z-score equivalent (standard score) ◦ **NOCENTER** - scale key field values by dividing by their standard deviation, but do not center them on a mean of zero (0) ◦ **NOSCALE** - use the raw key field values, uncentered and unscaled For more information, see "Specify a data preprocessing method" on page 1313.

Examples

Clustering on invoice amount

In addition to stratifying an accounts receivable table on the **Invoice_Amount** field, you also decide to cluster on the same field.

- Stratifying groups the amounts into strata with predefined numeric boundaries – for example, $1000 intervals.
- Clustering discovers any organic groupings of amounts that exist in the data without requiring that you decide on numeric boundaries in advance.

```
Open Ar
CLUSTER ON Invoice_Amount KVALUE 8 ITERATIONS 30 INITIALIZATIONS 10
OTHER No Due Date Ref Type TO "Clustered_invoices" NOSCALE
```

As a quick way of discovering how many records are contained in each output cluster, you classify the Clustered_invoices output table on the **Cluster** field.

```
OPEN Clustered_invoices
CLASSIFY ON Cluster TO SCREEN
```

Remarks

For more information about how this command works, see "Clustering data" on page 1307.

COMMENT command

Adds an explanatory note to a script without affecting processing.

Syntax

Single-line comments

```
COMMENT comment_text
```

Multiline comments

```
COMMENT
  comment_text
  <...n>
<END>
```

> **Note**
> Do not use a caret character ^ to preface lines of comment text. The caret has a special use in the .acl project file, and comment text is not saved if you preface it with a caret.

Parameters

Name	Description
comment_text	The comment you are adding. ◦ **single-line comment** - enter the entire comment text without a line break ◦ **multiline comment** - enter as many lines of comment text as necessary starting on the line immediately following the COMMENT command Terminate a multiline comment with the END keyword on a separate line, or with a blank line.
END optional	The end of a multiline COMMENT command. If you use END, it must be entered on the line immediately following the last comment line. If you omit END, a blank line must follow the last comment line.

Examples

Single-line comments

You use single-line comments before commands to add documentation for future users who will maintain the script:

```
COMMENT Generate the standard deviation and average.
STATISTICS ON %v_amt% STD TO SCREEN NUMBER 5
COMMENT Create fields for storing standard deviation and average.
DEFINE FIELD Standard_Dev COMPUTED STDDEV1
DEFINE FIELD Average COMPUTED AVERAGE1
```

Multiline comment

You begin each script you write with a multiline comment that explains the purpose of the script:

```
COMMENT
  This analytic identifies multiple records having common
  transaction originator IDs (like vendor ID or merchant ID)
  where the transaction date values are either equal or one day apart.
  This analytic can be used for split invoices, split purchase orders,
  split requisitions, and split corporate card transactions.
END
```

Remarks

When to use COMMENT

Use COMMENT to include information about the purpose of a script, the logic used, and other information such as the required inputs for the script and the purpose of each variable you define.

The comments in a script are written to the Analytics command log each time the script runs.

COUNT command

Counts the total number of records in the current view, or only those records that meet the specified condition.

Syntax

```
COUNT <IF test> <WHILE test> <FIRST range|NEXT range>
```

Parameters

Name	Description
IF *test* optional	A conditional expression that must be true in order to process each record. The command is executed on only those records that satisfy the condition. **Note** The IF parameter is evaluated against only the records remaining in a table after any scope parameters have been applied (WHILE, FIRST, NEXT).
WHILE *test* optional	A conditional expression that must be true in order to process each record. The command is executed until the condition evaluates as false, or the end of the table is reached. **Note** If you use WHILE in conjunction with FIRST or NEXT, record processing stops as soon as one limit is reached.
FIRST *range* | NEXT *range* optional	The number of records to process: ◦ **FIRST** - start processing from the first record until the specified number of records is reached ◦ **NEXT** - start processing from the currently selected record until the specified number of records is reached Use *range* to specify the number of records to process. If you omit FIRST and NEXT, all records are processed by default.

Analytics output variables

Name	Contains
COUNT*n*	The record count calculated by the command. ◦ If the variable name is COUNT1, it is storing the record count for the most recent command executed. ◦ If the variable name is COUNT*n*, where *n* is greater than 1, the variable is storing the record count for a command executed within a GROUP command. The value of *n* is assigned based on the line number of the command in the GROUP. For example, if the command is one line below the GROUP command it is assigned the value COUNT2. If the command is four lines below the GROUP command, it is assigned the value COUNT5.

Examples

Storing COUNT1

The result of the COUNT command is stored in the COUNT1 output variable. You can retrieve and store this value in a user-defined variable.

The COUNT command overwrites the COUNT1 variable each time it is executed, so the value needs to be stored in a user-defined variable before the command is executed for the second time after the filter is applied to the table:

```
OPEN CustomerAddress
COUNT
TotalRec = COUNT1
SET FILTER TO ModifiedDate > '20100101'
COUNT
TotalFilteredRec = COUNT1
```

Remarks

When to use COUNT

Use the COUNT command to count the number of records in an Analytics table, or to count the number of records that meet a particular test condition. If no test is specified, the total number of

records in the Analytics table is displayed.

How filters affect COUNT

If a filter has been applied to a view, the command counts the number of records remaining in the view after the filtering condition has been applied.

CREATE LAYOUT command

Creates an empty Analytics table layout, which may be required in certain scripting situations.

Syntax

```
CREATE LAYOUT layout_name WIDTH characters <RECORD 0|RECORD 1>
```

Parameters

Name	Description
layout_name	The name of the layout.
WIDTH *characters*	The record length in characters.
RECORD 0 | RECORD 1 optional	◦ If you specify `RECORD 0`, or omit this parameter, the table layout is created without any records or a source data file. ◦ If you specify `RECORD 1` the table layout is created with a single empty record and a source data file named *layout_name*.fil.

Examples

Creating an empty table layout without any records

You create an empty table layout with a record length of 100 characters:

```
CREATE LAYOUT empty_table WIDTH 100
```

Creating an empty table layout with one record

You create:

- an empty table layout with one empty record
- a record length of 50 characters
- an associated Analytics data file called `empty_table.fil`

```
CREATE LAYOUT empty_table WIDTH 50 RECORD 1
```

Remarks

The empty table layout is created with a single character field called **Field_1**. The field length is the same as the record length you specify with WIDTH.

> **Note**
> This command is not supported for use in Analytics scripts run on AX Server.

CROSSTAB command

Groups records based on identical combinations of values in two or more character or numeric fields, and displays the resulting groups in a grid of rows and columns. Counts the number of records in each group, and also subtotals specified numeric fields for each group.

Syntax

```
CROSSTAB {ON row_field <...n>|ON ALL <EXCLUDE field_name <...n>>} COLUMNS
column_field <SUBTOTAL numeric_field <...n>|SUBTOTAL ALL <EXCLUDE numeric_
field <...n>>> TO {SCREEN|table_name|filename|GRAPH|PRINT} <LOCAL> <IF test>
<WHILE test> <FIRST range|NEXT range> <APPEND> <COUNT> <OPEN> <HEADER header_
text> <FOOTER footer_text>
```

Parameters

Name	Description
ON *row_field <...n>* | ON ALL	One or more character or numeric fields or expressions to use for rows in the resulting grid of rows and columns. ◦ **ON *row_field <...n>*** - use the specified field or fields Multiple fields must be separated by spaces, and can be different data types. If you use more than one field, fields are included in the order that you list them. ◦ **ON ALL** - use all fields in the table If you use all fields, fields are included in the order that they appear in the table layout.
EXCLUDE *field_name* optional	Only valid when using ON ALL. The field or fields to exclude from the command. EXCLUDE allows you to fine-tune ON ALL, by excluding the specified fields. EXCLUDE must immediately follow ON ALL. For example: `ON ALL EXCLUDE field_1 field_2`
COLUMNS *column_field*	The character or numeric field or expression to use for columns in the resulting grid of rows and columns. You can specify only one field or expression for the columns.

Name	Description
SUBTOTAL *numeric_field <...n>* | SUBTOTAL ALL optional	One or more numeric fields or expressions to subtotal for each group. Multiple fields must be separated by spaces. Specify ALL to subtotal all the numeric fields in the table.
EXCLUDE *numeric_field* optional	Only valid when using SUBTOTAL ALL. The field or fields to exclude from the command. EXCLUDE allows you to fine-tune SUBTOTAL ALL, by excluding the specified fields. EXCLUDE must immediately follow SUBTOTAL ALL. For example: `SUBTOTAL ALL EXCLUDE field_1 field_2`
TO SCREEN | *table_name* | *filename* | GRAPH | PRINT	The location to send the results of the command to: ◦ **SCREEN** - displays the results in the Analytics display area **Tip** You can click any linked result value in the display area to drill down to the associated record or records in the source table. ◦ ***table_name*** - saves the results to an Analytics table Specify *table_name* as a quoted string with a .FIL file extension. For example: `TO "Output.FIL"` By default, the table data file (.FIL) is saved to the folder containing the Analytics project. Use either an absolute or relative file path to save the data file to a different, existing folder: • `TO "C:\Output.FIL"` • `TO "Results\Output.FIL"` **Note** Table names are limited to 64 alphanumeric characters, not including the .FIL extension. The name can include the underscore character (_), but no other special characters, or any spaces. The name cannot start with a number. ◦ ***filename*** - saves the results to a file Specify *filename* as a quoted string with the appropriate file extension. For example: `TO "Output.TXT"` By default, the file is saved to the folder containing the Analytics project. Use either an absolute or relative file path to save the file to a different, existing folder: • `TO "C:\Output.TXT"` • `TO "Results\Output.TXT"` ◦ **GRAPH** - displays the results in a graph in the Analytics display area ◦ **PRINT** - sends the results to the default printer

Name	Description
LOCAL optional	Saves the output file in the same location as the Analytics project. **Note** Applicable only when running the command against a server table with an output file that is an Analytics table. The LOCAL parameter must immediately follow the TO parameter.
IF *test* optional	A conditional expression that must be true in order to process each record. The command is executed on only those records that satisfy the condition. **Note** The IF parameter is evaluated against only the records remaining in a table after any scope parameters have been applied (WHILE, FIRST, NEXT).
WHILE *test* optional	A conditional expression that must be true in order to process each record. The command is executed until the condition evaluates as false, or the end of the table is reached. **Note** If you use WHILE in conjunction with FIRST or NEXT, record processing stops as soon as one limit is reached.
FIRST *range* | NEXT *range* optional	The number of records to process: ◦ **FIRST** - start processing from the first record until the specified number of records is reached ◦ **NEXT** - start processing from the currently selected record until the specified number of records is reached Use *range* to specify the number of records to process. If you omit FIRST and NEXT, all records are processed by default.
APPEND optional	Appends the command output to the end of an existing file instead of overwriting it. **Note** You must ensure that the structure of the command output and the existing file are identical: • the same fields • the same field order • matching fields are the same length • matching fields are the same data type Analytics appends output to an existing file regardless of its structure. If the structure of the output and the existing file do not match, jumbled, missing, or inaccurate data can result.
COUNT optional	Includes record counts as columns. Counts are useful when you use SUBTOTAL. Counts are automatically included if you do not select any subtotal fields.
OPEN	Opens the table created by the command after the command executes. Only valid if

Name	Description
optional	the command creates an output table.
HEADER *header_text* optional	The text to insert at the top of each page of a report. *header_text* must be specified as a quoted string. The value overrides the Analytics HEADER system variable.
FOOTER *footer_text* optional	The text to insert at the bottom of each page of a report. *footer_text* must be specified as a quoted string. The value overrides the Analytics FOOTER system variable.

Examples

Cross-tabulating an accounts receivable table with SUBTOTAL

You want to cross-tabulate an accounts receivable table on the **Customer Number** and **Transaction Type** fields. You also want to subtotal the **Transaction Amount** field.

The output is grouped by customer, and within each customer by transaction type. It includes the total transaction amount for each customer for each transaction type:

```
OPEN Ar
CROSSTAB ON Customer_Number COLUMNS Trans_Type SUBTOTAL Trans_Amount
COUNT TO SCREEN
```

Cross-tabulating an accounts receivable table to find duplicate transactions

You need to find evidence of duplicate transactions in an accounts receivable table.

To do this, you cross-tabulate an accounts receivable table on the **Transaction Amount** and **Transaction Type** fields. The output groups and counts identical transaction amounts for each transaction type:

```
OPEN Ar
CROSSTAB ON Trans_Amount COLUMNS Trans_Type TO SCREEN
```

Remarks

For more information about how this command works, see "Cross-tabulating data" on page 1279.

How it works

CROSSTAB groups records that have the same combination of values in two or more character or numeric fields.

The output contains a grid of rows and columns similar to a pivot table. It includes a single row-column intersection for each group, with a count of the number of records in the source table that belong to the group.

Sorting and CROSSTAB

CROSSTAB can process either sorted or unsorted data. Both the *row_field* and the *column_field* in the output are automatically sorted in ascending order.

If you specify more than one *row_field*, the fields use a nested sort. The order of the nesting follows the field order you specify, or the order of the fields in the table layout if you use ON ALL.

CVSEVALUATE command

For classical variables sampling, provides four different methods for projecting the results of sample analysis to the entire population.

Syntax

```
CVSEVALUATE BOOKED book_value_field AUDITED audit_value_field ETYPE
{MPU|DIFFERENCE|RATIO SEPARATE|RATIO COMBINED|ALL} STRATA boundary_value
<,...n> POPULATION stratum_count,stratum_book_value <,...n> CONFIDENCE con-
fidence_level CUTOFF value,certainty_stratum_count,certainty_stratum_book_
value ERRORLIMIT number PLIMIT {BOTH|UPPER|LOWER} <BCUTOFF value,certainty_
stratum_count,certainty_stratum_book_value> <TO {SCREEN|filename}>
```

Parameters

> **Note**
>
> If you are using the output results of the CVSPREPARE and CVSSAMPLE commands as input for the CVSEVALUATE command, a number of the parameter values are already specified and stored in variables. For more information, see "CVSPREPARE command" on page 1613 and "CVSSAMPLE command" on page 1618.
>
> Do not include thousands separators, or percentage signs, when you specify values.

Name	Description
BOOKED *book_value_field*	The numeric book value field to use in the evaluation.
AUDITED *audit_value_field*	The numeric audit value field to use in the evaluation.
ETYPE MPU | DIFFERENCE | RATIO SEPARATE | RATIO COMBINED | ALL	The estimation type to use: • MPU (Mean-per-unit) • Difference • Ratio Separate • Ratio Combined • All For more information, see "Which estimation type should I use?" on page 1610

Name	Description
STRATA *boundary_value* <,...*n*>	The upper boundary values to use for stratifying the *book_value_field*.
POPULATION *stratum_count*, *stratum_value* <,...*n*>	The number of records and the total value for each stratum in the *book_value_field*.
CONFIDENCE *confidence_level*	The confidence level used during the preparation stage of the classical variables sample.
CUTOFF *value*, *certainty_stratum_count*, *certainty_stratum_book_value*	◦ ***value*** - the top certainty stratum cutoff value used during the preparation and sampling stage of the classical variables sample ◦ ***certainty_stratum_count*** - the number of records in the top certainty stratum ◦ ***certainty_stratum_book_value*** - the total book value of the records in the top certainty stratum
ERRORLIMIT *number*	The minimum number of errors you expect in the sample. **Note** If the actual number of errors you found when you analyzed the sample is less than the ERRORLIMIT *number*, the only evaluation method available is mean-per-unit.
PLIMIT BOTH | UPPER | LOWER	The type of precision limit to use: ◦ Both ◦ Upper ◦ Lower For more information, see "CVSPREPARE command" on page 1613.
BCUTOFF *value*, *certainty_stratum_count*, *certainty_stratum_book_value* optional	◦ ***value*** - the bottom certainty stratum cutoff value used during the preparation and sampling stage of the classical variables sample ◦ ***certainty_stratum_count*** - the number of records in the bottom certainty stratum ◦ ***certainty_stratum_book_value*** - the total book value of the records in the bottom certainty stratum
TO SCREEN | *filename*	The location to send the results of the command to: ◦ **SCREEN** - displays the results in the Analytics display area **Tip** You can click any linked result value in the display area to drill down to the associated record or records in the source table. ◦ ***filename*** - saves the results to a file Specify *filename* as a quoted string with the appropriate file extension. For example: `TO "Output.TXT"` By default, the file is saved to the folder containing the Analytics project. Use either an absolute or relative file path to save the file to a different, existing folder:

Name	Description
	• `TO "C:\Output.TXT"` • `TO "Results\Output.TXT"`

Examples

Project errors found in the sampled data to the entire population

You have completed your testing of the sampled data and recorded the misstatements you found. You can now project the errors you found to the entire population.

The example below uses the Difference estimation type to project the results of sample analysis to the entire population:

```
CVSEVALUATE BOOKED invoice_amount AUDITED AUDIT_VALUE ETYPE DIFFERENCE
STRATA 4376.88,9248.74,16904.52,23864.32 POPULATION
1279,3382131.93,898,5693215.11,763,9987014.57,627,12657163.59,479,133463-
54.63 CONFIDENCE 95.00 CUTOFF 35000.00,36,1334318.88 ERRORLIMIT 6 PLIMIT
BOTH TO SCREEN
```

Remarks

For more information about how this command works, see "Evaluating errors in a classical variables sample" on page 1077.

Which estimation type should I use?

The estimation type that you should use depends on the nature of the data: the sample book values, the sample audit values, and the relation between them.

Guidelines

The guidelines below help you select an estimation type.

> **Tip**
> If you want to compare the results produced by different estimation types, you can specify `ETYPE ALL` to include all estimation types in the evaluation output.

Estimation type	Presence of misstatements	Size of misstatements	Sign of book values	Comparison of strata ratios
Mean-per-unit	**No misstatements, or very few misstatements** The only valid estimation type if there are no misstatements, or very few misstatements, in the audited sample population.	n/a	n/a	n/a
Difference	**Misstatements required** Requires a number of misstatements in the audited sample population. For example, 5% or more of the samples contain misstatements.	**Misstatements are non-proportional** More suitable when misstatements are non-proportional: the size of a misstatement is not related to the size of the associated book value. In other words, small and large book values can have either small or large misstatements.	n/a	n/a
Ratio Separate		**Misstatements are proportional** More suitable when misstatements are proportional: the size of a misstatement is related to the size of the associated book value. In other words, small book values have small misstatements, and large book values have large misstate-	**Book values have the same sign** All sample book values must have the same sign: either all positive, or all negative.	**Ratios vary** More suitable when the ratio of average sample audit value to average sample book value varies widely between strata.

Estimation type	Presence of mis-statements	Size of mis-statements	Sign of book val-ues	Comparison of strata ratios
		ments.		
Ratio Combined				**Ratios are consistent** More suitable when the ratio of average sample audit value to average sample book value is relatively consistent between strata.

CVSPREPARE command

Stratifies a population, and calculates a statistically valid sample size for each stratum, for classical variables sampling.

Syntax

```
CVSPREPARE ON book_value_field NUMSTRATA number MINIMUM minimum_strata_sample_
size PRECISION value CONFIDENCE confidence_level <CUTOFF value> <BCUTOFF
value> NCELLS number PLIMIT {BOTH|UPPER|LOWER} ERRORLIMIT number <IF test>
<MINSAMPSIZE minimum_sample_size> TO {SCREEN|filename}
```

Parameters

> **Note**
> Do not include thousands separators, or percentage signs, when you specify values.

Name	Description
ON *book_value_field*	The numeric book value field to use as the basis for preparing the classical variables sample.
NUMSTRATA *number*	The number of strata to use for numerically stratifying the *book_value_field*. The minimum number of strata is 1, and the maximum is 256. If you specify `NUMSTRATA 1`, and do not specify a `CUTOFF`, the population is unstratified prior to drawing a sample. **Note** The number of strata cannot exceed 50% of the number of cells specified for `NCELLS`.
MINIMUM *minimum_strata_sample_size*	The minimum number of records to sample from each stratum. Leave the default of zero (0) if you do not have a specific reason for specifying a minimum number.
PRECISION *value*	The monetary amount that is the difference between the tolerable misstatement and the expected misstatement in the account. ◦ **Tolerable misstatement** - the maximum total amount of misstatement that can

Name	Description
	occur in the sample field without being considered a material misstatement ◦ **Expected misstatement** - the total amount of misstatement that you expect the sample field to contain The precision establishes the range of acceptability for an account to be considered fairly stated. Reducing the precision decreases the range of acceptability (the margin of error) which requires an increased sample size.
CONFIDENCE *confidence_level*	The desired confidence level that the resulting sample is representative of the entire population. For example, specifying 95 means that you want to be confident that 95% of the time the sample will in fact be representative. Confidence is the complement of "sampling risk". A 95% confidence level is the same as a 5% sampling risk. ◦ If `PLIMIT` is `BOTH`, the minimum confidence level is 10%, and the maximum is 99.5%. ◦ If `PLIMIT` is `UPPER` or `LOWER`, the minimum confidence level is 55%, and the maximum is 99.5%.
CUTOFF *value* optional	A top certainty stratum cutoff value. Amounts in the *book_value_field* greater than or equal to the cutoff value are automatically selected and included in the sample. If you omit CUTOFF, a default cutoff value equal to the maximum amount in the *book_value_field* is used, and no records are included in the top certainty stratum.
BCUTOFF *value* optional	A bottom certainty stratum cutoff value. Amounts in the *book_value_field* less than or equal to the cutoff value are automatically selected and included in the sample. If you omit BCUTOFF, a default cutoff value equal to the minimum amount in the *book_value_field* is used, and no records are included in the bottom certainty stratum.
NCELLS *number*	The number of cells to use for pre-stratifying the *book_value_field*. Cells are narrower numeric divisions than strata. Pre-stratification is part of an internal process that optimizes the position of strata boundaries. Cells are not retained in the final stratified output. The minimum number of cells is 2, and the maximum is 999. **Note** The number of cells must be at least twice (2 x) the number of strata specified for `NUMSTRATA`.
PLIMIT BOTH | UPPER | LOWER	The type of precision limit to use. ◦ `BOTH` - specify this option if: • the account as a whole could be either overstated or understated • you are interested in estimating whether misstatement in either direction exceeds the specified `PRECISION` ◦ `UPPER` - specify this option if:

Name	Description
	• the account as a whole is likely to be understated • you are only interested in estimating whether the total amount of understatement exceeds the specified `PRECISION` ◦ `LOWER` - specify this option if: • the account as a whole is likely to be overstated • you are only interested in estimating whether the total amount of overstatement exceeds the specified `PRECISION` **Caution** Specify `BOTH` if you are not sure which option to specify.
ERRORLIMIT *number*	The minimum number of errors you expect in the sample. **Note** If the actual number of errors you find when you analyze the sample is less than the ERRORLIMIT *number*, the only evaluation method available is mean-per-unit.
IF *test* optional	A conditional expression that must be true in order to process each record. The command is executed on only those records that satisfy the condition. **Caution** If you specify a conditional expression, an identical conditional expression must be used during both the calculation of the sample size, and the drawing of the sample. If you use a condition at one stage and not the other, or if the two conditions are not identical, the sampling results will probably not be statistically valid.
MINSAMPSIZE *minimum_sample_size* optional	The minimum number of records to sample from the entire population. Leave the default of zero (0) if you do not have a specific reason for specifying a minimum number.
TO SCREEN | *filename*	The location to send the results of the command to: ◦ **SCREEN** - displays the results in the Analytics display area **Tip** You can click any linked result value in the display area to drill down to the associated record or records in the source table. ◦ ***filename*** - saves the results to a file Specify *filename* as a quoted string with the appropriate file extension. For example: `TO "Output.TXT"` By default, the file is saved to the folder containing the Analytics project. Use either an absolute or relative file path to save the file to a different, existing folder: • `TO "C:\Output.TXT"` • `TO "Results\Output.TXT"`

Analytics output variables

Name	Contains
CONFIDENCE	The confidence level specified by the user.
ERRLIMIT	The minimum number of errors specified by the user.
NSTRATA	The number of strata specified by the user.
PLIMIT	The type of precision limit specified by the user.
S_IF	A conditional expression specified by the user
S_TOP	The top certainty stratum cutoff value specified by the user, or if none was specified, the upper boundary of the top stratum calculated by the command.
SAMPLEFIELD	The book value field specified by the user.
SBOTTOM	The bottom certainty stratum cutoff value specified by the user, or if none was specified, the lower boundary of the bottom stratum calculated by the command.
SBOUNDARY	All strata upper boundaries calculated by the command. Does not include top or bottom certainty strata.
SPOPULATION	The count of the number of records in each stratum, and the total monetary value for each stratum. Does not include top or bottom certainty strata.
SSAMPLE	The sample size for each stratum calculated by the command. Does not include top or bottom certainty strata.

Examples

Prepare a classical variables sample

You have decided to use classical variables sampling to estimate the total amount of monetary misstatement in an account containing invoices.

Before drawing the sample, you must first stratify the population, and calculate a statistically valid sample size for each stratum.

You want to be confident that 95% of the time the sample drawn by Analytics will be representative of the population as a whole.

Using your specified confidence level, the example below stratifies a table based on the **invoice_amount** field, and calculates the sample size for each stratum and the top certainty stratum:

```
CVSPREPARE ON invoice_amount NUMSTRATA 5 MINIMUM 0 PRECISION 928003.97
CONFIDENCE 95.00 CUTOFF 35000 NCELLS 50 PLIMIT BOTH ERRORLIMIT 6
MINSAMPSIZE 0 TO SCREEN
```

Remarks

For more information about how this command works, see "Preparing a classical variables sample" on page 1058.

Numeric length limitation

Several internal calculations occur during the preparation stage of classical variables sampling. These calculations support numbers with a maximum length of 17 digits. If the result of any calculation exceeds 17 digits, the result is not included in the output, and you cannot continue with the sampling process.

Note that source data numbers of less than 17 digits can produce internal calculation results that exceed 17 digits.

CVSSAMPLE command

Draws a sample of records using the classical variables sampling method.

Syntax

```
CVSSAMPLE ON book_value_field NUMSTRATA number <SEED seed_value> CUTOFF value
<BCUTOFF value> STRATA boundary_value <,...n> SAMPLESIZE number <,...n>
POPULATION stratum_count,stratum_value <,...n> <IF test> TO table_name
```

Parameters

> **Note**
>
> If you are using the output results of the CVSPREPARE command as input for the CVSSAMPLE command, a number of the parameter values are already specified and stored in variables. For more information, see "CVSPREPARE command" on page 1613.
>
> Do not include thousands separators, or percentage signs, when you specify values.

Name	Description
ON *book_value_field*	The numeric book value field to use as the basis for the sample.
NUMSTRATA *number*	The number of strata to use for stratifying the *book_value_field*.
SEED *seed_value* optional	The seed value to use to initialize the random number generator in Analytics. If you omit SEED, Analytics randomly selects the seed value.
CUTOFF *value*	A top certainty stratum cutoff value. Amounts in the *book_value_field* greater than or equal to the cutoff *value* are automatically selected and included in the sample.
BCUTOFF *value* optional	A bottom certainty stratum cutoff value. Amounts in the *book_value_field* less than or equal to the cutoff *value* are automatically selected and included in the sample.
STRATA *boundary_value* <,...*n*>	The upper boundary values to use for stratifying the *book_value_field*.

Name	Description
SAMPLESIZE *number* <,...*n*>	The number of records to sample from each stratum.
POPULATION *stratum_count*, *stratum_value* <,...*n*>	The number of records in each stratum, and the total value for each stratum.
IF *test* optional	A conditional expression that must be true in order to process each record. The command is executed on only those records that satisfy the condition. **Caution** If you specify a conditional expression, an identical conditional expression must be used during both the calculation of the sample size, and the drawing of the sample. If you use a condition at one stage and not the other, or if the two conditions are not identical, the sampling results will probably not be statistically valid.
TO *table_name*	The location to send the results of the command to: ◦ ***table_name*** - saves the results to an Analytics table Specify *table_name* as a quoted string with a .FIL file extension. For example: `TO "Output.FIL"` By default, the table data file (.FIL) is saved to the folder containing the Analytics project. Use either an absolute or relative file path to save the data file to a different, existing folder: • `TO "C:\Output.FIL"` • `TO "Results\Output.FIL"` **Note** Table names are limited to 64 alphanumeric characters, not including the .FIL extension. The name can include the underscore character (_), but no other special characters, or any spaces. The name cannot start with a number.

Analytics output variables

Name	Contains
S_TOPEV	The top certainty stratum cutoff value specified by the user, or if none was specified, the upper boundary of the top stratum previously calculated by the CVSPREPARE command. Also stores the count of the number of records in the top certainty stratum, and their total monetary value.

Name	Contains
SBOTTOMEV	The bottom certainty stratum cutoff value specified by the user, or if none was specified, the lower boundary of the bottom stratum previously calculated by the CVSPREPARE command. Also stores the count of the number of records in the bottom certainty stratum, and their total monetary value.
SBOUNDARYEV	All strata upper boundaries prefilled by the command, or specified by the user. Does not include top or bottom certainty strata.
SPOPULATION	The count of the number of records in each stratum, and the total monetary value for each stratum. Does not include top or bottom certainty strata.

Examples

Draw a classical variables sample

You are going to use classical variables sampling to estimate the total amount of monetary misstatement in an account containing invoices.

After stratifying the population, and calculating a statistically valid sample size for each stratum, you are ready to draw the sample.

The example below draws a stratified sample of records based on the **invoice_amount** field, and outputs the sampled records to the `Invoices_sample` table:

```
CVSSAMPLE ON invoice_amount NUMSTRATA 5 SEED 12345 CUTOFF 35000.00
STRATA 4376.88,9248.74,16904.52,23864.32,35000.00 SAMPLESIZE
37,36,49,36,39 POPULATION
1279,3382131.93,898,5693215.11,763,9987014.57,627,12657163.59,479,133463-
54.63 TO "Invoices_sample"
```

Remarks

For more information about how this command works, see "Performing classical variables sampling" on page 1069.

System-generated fields

Analytics automatically generates four fields and adds them to the sample output table. For each record included in the sample, the fields contain the following descriptive information:

- **STRATUM** – the number of the stratum to which the record is allocated
- **ORIGIN_RECORD_NUMBER** – the original record number in the source data table
- **SELECTION_ORDER** – on a per-stratum basis, the order in which the record was randomly selected
- **SAMPLE_RECORD_NUMBER** – the record number in the sample output table

DEFINE COLUMN command

Creates and adds one or more columns to an existing view.

Syntax

```
DEFINE COLUMN view_name field_name <AS display_name> <POSITION n> <WIDTH characters> <PIC format> <SORT|SORT D> <KEY> <PAGE> <NODUPS> <NOZEROS> <LINE n>
```

Parameters

Name	Description
view_name	The view to add the column to.
field_name	The field to create the column for. To use a field from a related table, specify the field name as `table_name.field_name`.
AS *display_name* optional	The display name (alternate column title) for the field in the view. If you want the display name to be the same as the field name do not use AS. Specify *display_name* as a quoted string. Use a semi-colon (;) between words if you want a line break in the column title.
POSITION *n* optional	The position of the column in the view numerically from left to right: ◦ if omitted, the column is placed as the rightmost column at the time that the column is added ◦ if a position number is missing, column positions are adjusted so that the columns are positioned sequentially ◦ if a position number is already in use, the new column is placed to the left of the column already using the position number
WIDTH *characters* optional	The display width of the field in characters. The specified value controls the display width of the field in Analytics views and reports. The display width never alters data, however it can hide data if it is shorter than the field length. If you omit WIDTH, the display width is set to the character width specified for the field in the table layout.

Name	Description
	Note The characters specified by WIDTH are fixed-width characters. Every character is allotted the same amount of space, regardless of the width of the actual character. By default, views in Analytics use a proportional width font that does not correspond with fixed-width character spacing. If you want a one-to-one correspondence between the WIDTH value and the characters in the view, you can change the **Proportional Font** setting in the **Options** dialog box to a fixed-width font such as Courier New.
PIC *format* optional	**Note** Applies to numeric or datetime fields only. ◦ **numeric fields** - the display format of numeric values in Analytics views and reports ◦ **datetime fields** - the physical format of datetime values in the source data (order of date and time characters, separators, and so on) **Note** For datetime fields, *format* must exactly match the physical format in the source data. For example, if the source data is 12/31/2014, you must enter the format as `"MM/DD/YYYY"`. *format* must be enclosed in quotation marks.
SORT | SORT D optional	Sorts the column: ◦ **ascending order** - `SORT` ◦ **descending order** - `SORT D`
KEY optional	The column is designated as a break field in reports. Reports are subtotaled and subdivided when the value of the column changes. The following restrictions apply to break fields: ◦ must be a character field or expression ◦ if a break field is set in the view, it must be the leftmost column ◦ the last column in the view cannot be a break field ◦ if you have more than one break field, all of the columns to the left of any additional break field must also be break fields
PAGE optional	Inserts a page break each time the value in the break field changes.
NODUPS optional	Substitutes blank values for repeated values in the field. For example, if the customer name is listed for each invoice record, the report is easier to read if it shows only the first instance of each customer name.
NOZEROS optional	Substitutes blank values for zero values in the field. For example, if a report includes a large number of zero values in a field, the report

Name	Description
	is easier to read if it only displays non-zero values.
LINE *n* optional	The number of lines in the column. If no value is specified, the column defaults to a single line. The value of *n* must be between 2 and 60.

Examples

Defining a view with six columns

With the **AR** table open, you define a view called **AR_Report**, and define six columns. The columns are displayed in the listed order:

```
OPEN Ar
DEFINE VIEW AR_Report OK
DEFINE COLUMN AR_Report No AS "Customer Number" WIDTH 7 KEY
DEFINE COLUMN AR_Report Date AS "Invoice Date" WIDTH 10
DEFINE COLUMN AR_Report Due AS "Due Date" WIDTH 10
DEFINE COLUMN AR_Report Reference AS "Reference Number" WIDTH 6
DEFINE COLUMN AR_Report Type AS "Transaction Type" WIDTH 5
DEFINE COLUMN AR_Report Amount AS "Transaction Amount" WIDTH 12 PIC "-
9999999999.99"
```

DEFINE FIELD command

Defines a physical data field in an Analytics table layout.

Syntax

```
DEFINE FIELD field_name data_type start_position length <decimals|date_format>
<NDATETIME> <PIC format> <AS display_name> <WIDTH characters> <SUPPRESS>
<field_note>
```

Parameters

Name	Description
field_name	The name of the field. **Note** Field names are limited to 256 upper and lowercase alphanumeric characters. The name can include the underscore character (_), but no other special characters, or any spaces. The name cannot start with a number. Analytics has a number of reserved keywords that cannot be used as field names. For a complete list, see "Reserved keywords" on page 1364.
data_type	The data type to use when interpreting the data. For a list of supported data types, see "Supported data types" on page 1631. For example, invoice numbers may be stored as numeric values in the data source. To treat these values as strings rather than numbers, you can define the field as character data instead.
start_position	The starting byte position of the field in the Analytics data file.

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>

Note

non-Unicode Analytics	1 byte = 1 character
Unicode Analytics, extended ASCII (ANSI) data	1 byte = 1 character
Unicode Analytics, Unicode data	2 bytes = 1 character

For Unicode data, typically you should specify an odd-numbered starting byte position. Specifying an even-numbered starting position can cause characters to display incorrectly.
</td></tr>
<tr><td>length</td><td>The length of the field in bytes.

Note

non-Unicode Analytics	1 byte = 1 character
Unicode Analytics, extended ASCII (ANSI) data	1 byte = 1 character
Unicode Analytics, Unicode data	2 bytes = 1 character

For Unicode data, specify an even number of bytes only. Specifying an odd number of bytes can cause characters to display incorrectly.
</td></tr>
<tr><td>decimals
optional</td><td>The number of decimals for numeric fields.</td></tr>
<tr><td>date_format
optional</td><td>The date format in the source date fields.
For datetime or time fields, use PIC format instead. You can also use PIC format for date fields.
If the source data includes separators such as slashes, you must include the separators in date_format. For example, if the source data is 12/31/2014, you must enter the format as <code>MM/DD/YYYY</code>. Do not enclose date_format in quotation marks.</td></tr>
<tr><td>NDATETIME
optional</td><td>Date, datetime, or time values stored in a numeric field are treated as datetime data.
NDATETIME requires that you also specify the source datetime format using PIC format.</td></tr>
<tr><td>PIC format
optional</td><td>

Note

Applies to numeric or datetime fields only.

- **numeric fields** - the display format of numeric values in Analytics views and reports
- **datetime fields** - the physical format of datetime values in the source data (order of date and time characters, separators, and so on)
</td></tr>
</table>

Name	Description
	Note For datetime fields, *format* must exactly match the physical format in the source data. For example, if the source data is 12/31/2014, you must enter the format as `"MM/DD/YYYY"`. *format* must be enclosed in quotation marks.
AS *display_name* optional	The display name (alternate column title) for the field in the view. If you want the display name to be the same as the field name do not use AS. Specify *display_name* as a quoted string. Use a semi-colon (;) between words if you want a line break in the column title.
WIDTH *characters* optional	The display width of the field in characters. The specified value controls the display width of the field in Analytics views and reports. The display width never alters data, however it can hide data if it is shorter than the field length. The display width cannot be less than the length of *field_name*, or *display_name*. If you omit WIDTH, the display width is set to the field length in characters. **Note** The characters specified by WIDTH are fixed-width characters. Every character is allotted the same amount of space, regardless of the width of the actual character. By default, views in Analytics use a proportional width font that does not correspond with fixed-width character spacing. If you want a one-to-one correspondence between the WIDTH value and the characters in the view, you can change the **Proportional Font** setting in the **Options** dialog box to a fixed-width font such as Courier New.
SUPPRESS optional	Only applies to numeric fields. Suppresses automatic totaling of a numeric field in Analytics reports. Totaling of some numeric fields is not appropriate. For example, a unit cost field, or a discount rate field.
field_note optional	Field note text that is added to the field definition in the table layout. *field_note* must be last, after all other required and optional parameters. The text cannot be multiline. Quotation marks are not required.

Examples

Defining a character field

Defines a character field called **ProdDesc**. The column title in the view is **Product Description**.

non-Unicode Analytics

- Starts at: byte 12 (character position 12)
- Length: 24 bytes (24 characters)

```
DEFINE FIELD ProdDesc ASCII 12 24 AS "Product Description"
```

Unicode Analytics, extended ASCII (ANSI) data

- Starts at: byte 12
- Length: 24 bytes (24 characters)

```
DEFINE FIELD ProdDesc ASCII 12 24 AS "Product Description"
```

Unicode Analytics, Unicode data

- Starts at: byte 13
- Length: 48 bytes (24 characters)

```
DEFINE FIELD ProdDesc UNICODE 13 48 AS "Product Description"
```

Defining a numeric field

Defines a numeric field called **QtyOH**. In the view, the column uses the specified display format, and the title is **Quantity On Hand**.

- Starts at: byte 61
- Length: 10 bytes
- Decimal places: none

```
DEFINE FIELD QtyOH NUMERIC 61 10 0 PIC "(9,999,999)" AS "Quantity On
Hand"
```

Defining a datetime field from character data

From source character data, the first two examples below define a datetime field called **Transaction_date**. In the source data, the date format is DD/MM/YYYY. No column title is specified, so the column title defaults to using the field name.

- Starts at: byte 20
- Length: 10 bytes

Here, the date format is specified using *date_format*:

```
DEFINE FIELD Transaction_date DATETIME 20 10 DD/MM/YYYY
```

Here, the date format is specified using PIC *format*:

```
DEFINE FIELD Transaction_date DATETIME 20 10 PIC "DD/MM/YYYY"
```

When defining datetime fields that include time data, you must use PIC *format*,

The example below defines a datetime field called **email_timestamp**. In the source data, the datetime format is YYYY/MM/DD hh:mm:ss-hh:mm.

- Starts at: byte 1
- Length: 25 bytes

```
DEFINE FIELD email_timestamp DATETIME 1 25 PIC "YYYY/MM/DD hh:mm:ss-
hh:mm"
```

Defining a datetime field from numeric data

From source numeric data, defines a datetime field called **Receipt_timestamp** that has the specified datetime format in the source data.

- Starts at: byte 15
- Length: 15 bytes

```
DEFINE FIELD Receipt_timestamp DATETIME 15 15 PIC "YYYYMMDD.hhmmss"
```

Defining a "numeric" datetime field

From source numeric data, defines a numeric field called **Receipt_timestamp** that has the specified datetime format in the source data.

The NDATETIME parameter allows datetime values stored in the numeric field to be treated as datetime data by Analytics.

- Starts at: byte 15
- Length: 15 bytes
- Decimal places: 6

```
DEFINE FIELD Receipt_timestamp PRINT 15 15 6 NDATETIME PIC "YYYYMMDD.h-
hmmss"
```

Defining a physical data field that reads mainframe Packed data

You can use the NDATETIME option to create a physical data field that reads date values from a Packed numeric field.

Analytics cannot recognize a date in a number that is compressed into fewer bytes than one per digit and that displays no date format. Consequently, you must unpack the number with NDATETIME to obtain the full number of digits, then specify the date format with PIC.

To accurately indicate which numbers represent the day, the month, and the year, you specify the same date format as the one in the Packed record layout:

```
DEFINE FIELD date_field_name NUMERIC 1 8 0 NDATETIME PIC "YYYYMMDD"
```

Remarks

For more information about how this command works, see "Physical fields" on page 715.

Overwriting fields in a script

You can overwrite a field in a table layout by defining a field that uses the same name as the existing field. If SET SAFETY is ON, a confirmation dialog box appears before overwriting the existing field.

To avoid interrupting a script, you can SET SAFETY to OFF. The existing field is overwritten without additional confirmation.

Supported data types

Data category	Data type
Character	ASCII
	CUSTOM
	EBCDIC
	NOTE
	PCASCII
	UNICODE

Data category	Data type
Numeric	ACCPAC
	ACL
	BASIC
	BINARY
	FLOAT
	HALFBYTE
	IBMFLOAT
	MICRO
	NUMERIC
	PACKED
	PRINT
	UNISYS
	UNSIGNED
	VAXFLOAT
	ZONED
Datetime	DATETIME
Logical	LOGICAL

DEFINE FIELD . . . COMPUTED command

Defines a computed field in an Analytics table layout.

Syntax

To define a computed field:

```
DEFINE FIELD field_name COMPUTED expression
```

To define a computed field with optional parameters:

```
DEFINE FIELD field_name COMPUTED
<IF test> <STATIC> <PIC format> <AS display_name> <WIDTH characters>
<SUPPRESS> <field_note>
expression
```

To define a conditional computed field:

```
DEFINE FIELD field_name COMPUTED
*** BLANK_LINE ***
value IF condition
<value IF condition>
<...n>
default_value
```

To define a conditional computed field with optional parameters:

```
DEFINE FIELD field_name COMPUTED
<IF test> <STATIC> <PIC format> <AS display_name> <WIDTH characters>
<SUPPRESS> <field_note>
value IF condition
<value IF condition>
```

```
<...n>
default_value
```

> **Note**
> Multiline syntax must be structured exactly as shown in the generic syntax above and the examples below.

Parameters

Name	Description
field_name	The name of the computed field. **Note** Field names are limited to 256 upper and lowercase alphanumeric characters. The name can include the underscore character (_), but no other special characters, or any spaces. The name cannot start with a number. Analytics has a number of reserved keywords that cannot be used as field names. For a complete list, see "Reserved keywords" on page 1364.
expression	A valid Analytics expression that defines the value of the computed field.
IF *test* optional	A conditional expression that must be true in order to process each record. The command is executed on only those records that satisfy the condition. **Note** The IF parameter is evaluated against only the records remaining in a table after any scope parameters have been applied (WHILE, FIRST, NEXT).
STATIC optional	The field displays the same value on every line of the table until a new value is encountered. For example, if there is a Last Name field in the source data where: ◦ the first record displays the value "Smith" ◦ the next five records display blank lines ◦ the seventh record displays the value "Wong" In this case, "Smith" displays on six consecutive lines, then "Wong" displays on the seventh line.
PIC *format* optional	**Note** Applies to numeric fields only. The display format of numeric values in Analytics views and reports.

Name	Description
	format must be enclosed in quotation marks.
AS *display_name* optional	The display name (alternate column title) for the field in the view. If you want the display name to be the same as the field name do not use AS. Specify *display_name* as a quoted string. Use a semi-colon (;) between words if you want a line break in the column title.
WIDTH *characters* optional	The display width of the field in characters. The specified value controls the display width of the field in Analytics views and reports. The display width never alters data, however it can hide data if it is shorter than the field length. The display width cannot be less than the length of *field_name*, or *display_name*. If you omit WIDTH, the display width is set to the field length in characters. **Note** The characters specified by WIDTH are fixed-width characters. Every character is allotted the same amount of space, regardless of the width of the actual character. By default, views in Analytics use a proportional width font that does not correspond with fixed-width character spacing. If you want a one-to-one correspondence between the WIDTH value and the characters in the view, you can change the **Proportional Font** setting in the **Options** dialog box to a fixed-width font such as Courier New.
SUPPRESS optional	Applies to numeric fields only. Suppresses automatic totaling of numeric computed fields in Analytics reports. Totaling of some numeric fields is not appropriate. For example, a unit cost field, or a discount rate field.
field_note optional	Field note text that is added to the field definition in the table layout. *field_note* must be last, after all other required and optional parameters. The text cannot be multiline. Quotation marks are not required.
value IF *condition*	Conditional computed field only. ◦ ***value*** – the computed field value or expression to use if the *condition* evaluates to true ◦ ***condition*** – the logical test that is evaluated
default_value	Conditional computed field only. The value or expression to use in the computed field if none of the conditions evaluate to true.

Name	Description
	Note The decimal precision of all numeric computed values is controlled by the precision of *default_value*. For example, if you specify a default value of 0.00, all computed values are calculated to two decimal places, and rounded if necessary. For greater precision, increase the number of decimal places in *default_value*.

Examples

Defining a computed field

You define a computed field named **Value** that is the product of the **Cost** and **Quantity** fields:

```
DEFINE FIELD Value COMPUTED Cost * Quantity
```

Defining a computed field with options

You define a computed field named **Value_03**, with several options defined. You include an IF condition that limits which records are processed by the computed field:

```
DEFINE FIELD Value_03 COMPUTED
IF Product_Class = "03" PIC "($9,999,999.99)" AS "Value Prod Class 3"
Value is cost times quantity
Cost * Quantity
```

Defining a conditional computed field

You define a conditional computed field named **Sales_tax** that calculates a different sales tax depending on the state in which the transaction occurred. Transactions that occurred outside the three states have a default sales tax of $0.00.

> **Note**
> The second line must be left blank because there are no optional parameters.

```
DEFINE FIELD Sales_tax COMPUTED

.0750 * Sale_amount IF State = "CA"
.0400 * Sale_amount IF State = "NY"
.0625 * Sale_amount IF State = "TX"
0.00
```

Defining a conditional computed field with options

You define a conditional computed field named **Sales_tax_100** that calculates a different sales tax depending on the state in which the transaction occurred. The field only calculates tax on amounts of $100 or greater.

Transactions that occurred outside the three states have a default sales tax of $0.00.

> **Note**
> When you specify optional parameters, do not leave any lines blank.

```
DEFINE FIELD Sales_tax_100 COMPUTED
IF Sale_amount >= 100
.0750 * Sale_amount IF State = "CA"
.0400 * Sale_amount IF State = "NY"
.0625 * Sale_amount IF State = "TX"
0.00
```

Remarks

For more information about how this command works, see "Computed fields" on page 722.

Two types of computed fields

There are two types of computed fields:

- **standard computed field**

  A standard computed field performs the same calculation on every record in a table.

For example, in an Inventory table you could create a computed field that multiplies the value in the Cost field by the value in the Quantity field to calculate the Inventory Value at Cost for each record.

- **conditional computed field**

  A conditional computed field is capable of performing different calculations on the records in a table, based on a set of conditions that you specify. The calculation performed on a record depends on which condition the record meets.

  For example, in a Transactions table, you could create a conditional computed field that calculates sales tax using a rate that is adjusted based on the state in which the transaction occurred. Conditions such as `IF State = "CA"` and `IF State = "NY"` would test each record to identify which rate to use.

Guidelines for creating a conditional computed field

> **Note**
> When defining a conditional computed field, if you do not specify any of the optional parameters on the second line, you must leave the second line **blank**.

In addition to a default value, conditional computed fields require at least one conditional value. You must use the following multiline syntax to define a conditional computed field:

- optional parameters appear on the second line
- if there are no optional parameters, the second line must be left blank
- the first condition statement appears on the third line
- each additional condition statement requires a separate line
- the default value appears on the last line

Overwriting field definitions

You can overwrite a field definition in a table layout by defining a field that uses the same name as the existing field.

If SET SAFETY is ON, Analytics displays a confirmation dialog box before overwriting the existing field. To avoid interrupting a script, you can SET SAFETY to OFF, and Analytics overwrites the existing field without asking for confirmation.

DEFINE RELATION command

Defines a relation between two Analytics tables.

> **Note**
> You can relate up to 18 Analytics tables and access and analyze data from any combination of fields in the related tables as if they existed in a single table. You must specify a separate DEFINE RELATION command for each pair of related tables.

Syntax

```
DEFINE RELATION key_field WITH related_table_name INDEX index_name <AS rela-
tion_name>
```

Parameters

Name	Description
key_field	The key field in the parent table. You can select only one key field for each relation. **Note** When creating relations between parent tables and grandchild tables, you must specify a fully qualified key field name in the format *table_name.field_name*. In "Relate three tables" on the next page, see: `Vouchers.created_by`
WITH *related_table_name*	The name of the related table.
INDEX *index_name*	The name of the index for the key field in the related table. You must index the related table on the key field before you can relate the table.
AS *relation_name* optional	A unique name for the relation. By default, the name of the child table is used as the relation name. If you are defining additional relations to the same child table, you must specify a unique name.

Examples

Relate two tables

The example below relates the open table to the **Customer** table by using the customer number field (**CustNum**) as the key field:

```
DEFINE RELATION CustNum WITH Customer INDEX Customer_on_CustNum
```

Customer_on_CustNum is the name of the child table index on the key field. A child table index is required when you relate tables.

If the child table index does not already exist when you run the DEFINE RELATION command, an error message appears and the relation is not performed.

> **Tip**
> If you define a relation in the Analytics user interface, the child table index is automatically created for you.

Create a child table index before relating two tables

If required, you can create a child table index immediately before relating two tables. The example below shows creating an index for the **Customer** child table before relating the **Ar** table to the **Customer** table.

```
OPEN Customer
INDEX ON CustNum TO Customer_on_CustNum
Open Ar
DEFINE RELATION CustNum WITH Customer INDEX Customer_on_CustNum
```

Relate three tables

The example below relates three tables in the `ACL_Rockwood.ACL` sample project:

- **Vouchers_items** - the parent table
- **Vouchers** - the child table
- **Employees** - the grandchild table

By using the **Vouchers** table as an intermediary table in the relation, you can relate each voucher item with the employee who processed the item.

```
OPEN Vouchers
INDEX ON voucher_number TO "Vouchers_on_voucher_number"
OPEN Vouchers_items
DEFINE RELATION voucher_number WITH Vouchers INDEX Vouchers_on_voucher_
number
OPEN Employees
INDEX ON employee_number TO "Employees_on_employee_number"
OPEN Vouchers_items
DEFINE RELATION Vouchers.created_by WITH Employees INDEX Employees_on_
employee_number
```

Explanation of the syntax logic

1. Open the **Vouchers** table and index it on the **voucher_number** field.
2. Open the **Vouchers_items** table and relate it to the **Vouchers** table using **voucher_number** as the key field.
3. Open the **Employees** table and index it on the **employee_number** field.
4. Open the **Vouchers_items** table and relate it to the **Employees** table using **Vouchers.created_by** as the key field.

> **Note**
>
> **Vouchers.created_by** is available as a key field in the second relation because you already related **Vouchers_items** and **Vouchers** in the first relation.

Remarks

For more information about how this command works, see "Relating tables" on page 927.

DEFINE REPORT command

Creates a new view or opens an existing view.

Syntax

```
DEFINE REPORT view_name
```

Parameters

Name	Description
view_name	The name of a new view or an existing view. ◦ **new view** - creates a blank view with the specified name in the open table Any spaces in the view name are replaced with underscore characters. ◦ **existing view** - opens the specified view in the open table

Examples

Creating a new view

You create a new view called **Q4_AR_review**:

```
DEFINE REPORT Q4_AR_review
```

DEFINE TABLE DB command

Defines an Analytics server table by connecting to a database table using AX Connector. You can connect to a Microsoft SQL Server, Oracle, or DB2 database.

Syntax

```
DEFINE TABLE DB {SOURCE database_profile <PASSWORD num> <PASSWORD num> |
SERVER server_profile <PASSWORD num>} <FORMAT format_name> SCHEMA schema
<TITLED acl_table_name> <PRIMARY|SECONDARY> DBTABLE db_tablename FIELDS
{field_names|ALL} <...n> <WHERE condition> <ORDER field_names>
```

Parameters

SOURCE *database_profile*	The Analytics database profile to use to access the database engine. Database profiles include information required to connect to the database engine, including: ◦ a reference to the associated server profile ◦ the database type ◦ the database name ◦ user account information **Note** DEFINE TABLE DB supports connecting to only the following databases: Microsoft SQL Server, Oracle, or DB2.
PASSWORD *num* optional	The password definition to use. You do not use PASSWORD *num* to prompt for, or specify, an actual password. The password definition refers to a password previously supplied or set using the PASSWORD command, the SET PASSWORD command, or the PASSWORD analytic tag. *num* is the number of the password definition. For example, if two passwords have been previously supplied or set in a script, or when scheduling an analytic, `PASSWORD 2` specifies that password #2 is used. For more information about supplying or setting passwords, see: ◦ "PASSWORD command" on page 1893 ◦ "SET command" on page 1960 ◦ "PASSWORD tag" on page 2530 The password is only required if the database profile does not contain saved passwords. Use PASSWORD twice after the SOURCE keyword. The first password logs you on to the server, and the second one logs you on to the database.

SERVER *server_profile*	No longer used. Prior to version 10.0 of Analytics, used when connecting to ACL Server Edition for z/OS. From version 10.0 of Analytics, ACL Server Edition for z/OS is no longer included.
FORMAT *format_name* optional	The name of an Analytics table, or table layout file (.layout), with a table layout that you want to use.
SCHEMA *schema*	The schema to connect to. You must enclose the schema name in quotation marks.
TITLED *acl_table_name* optional	The name of the Analytics table to create. *acl_table_name* must be a quoted string. If you omit TITLED, Analytics uses the database table name. When you access more than one table at a time, Analytics uses the name of the first one.
PRIMARY | SECONDARY optional	Use the table as either a primary or secondary table in multi-file commands. If neither option is specified, the default value of PRIMARY is used.
DBTABLE *database_table*	The database table that you want to access. *database_table* must be a quoted string.

FIELDS *field_names* | ALL	The fields to include in the output:
	◦ **FIELDS *field_names*** - use the specified fields
	field_names must be a quoted string.
	◦ **ALL** - use all fields in the table
	To use fields from more than one table:
	a. Enter the first table name followed by the fields from that table.
	b. Enter the next table name followed by the fields from that table.
	c. For each additional table, repeat step b.
	`DBTABLE "DSN1310" FIELDS "Field_A Field_B Field_C"`
	`DBTABLE "DSN2516" FIELDS "Field_L Field_M Field_N"`
	Note
	Using AX Connector, you can access an unlimited number of related tables, but no more than five is recommended. Processing time increases when you access multiple tables.

WHERE *condition* optional	An SQL WHERE clause that limits the data to those records that meet the specified condition. You must use valid SQL syntax entered as a quoted string. When you join tables, Analytics displays the condition of the join in the WHERE clause: `"Table_1.First_name = Table_2.First_name"`

ORDER *field_names* optional	The fields the database engine uses to sort records. *field_names* must be a quoted string. The command takes longer to run when sorting records. Only use ORDER when sorting is important.

Examples

Example

You want to access data from a Microsoft SQL Server database via AX Connector. To do this, you use the DEFINE TABLE DB command. You include the SOURCE parameter to connect to AX Connector through a database profile:

```
DEFINE TABLE DB SOURCE "SQLServer_Audit" SCHEMA "HR" TITLED "Payroll"
DBTABLE "HR.Employee" FIELDS "EmployeeID" DBTABLE "HR.Em-
ployeePayHistory" FIELDS "Rate PayFrequency" WHERE "HR.Em-
ployee.EmployeeID=HR.EmployeePayHistory.EmployeeID"
```

Remarks

How it works

The Analytics server table is defined as a query that uses a database profile to connect to a database table.

Suppressing the time portion of datetime values

Preface the DEFINE TABLE DB command with the SET SUPPRESSTIME command to suppress the time portion of datetime values.

Using SET SUPPRESSTIME ON is for pre-version-10.0 Analytics scripts that assume the time portion of datetime values will be truncated. If SET SUPPRESSTIME ON is not added to these scripts, they cannot run in the datetime-enabled version of Analytics.

For more information, see the "SET SUPPRESSTIME" section in "SET command" on page 1960.

DEFINE VIEW command

Defines a new view or overwrites an existing view.

Syntax

```
DEFINE VIEW view_name <RLINES n> <ALL> <SUPPRESS> <SUMMARIZED> <IF test>
<WHILE test> <HEADER header_text> <FOOTER footer_text> <TO report_file_name
<HTML>> <OK>
```

Parameters

Name	Description
view_name	The name of the view to create or overwrite. **Note** View names are limited to 64 alphanumeric characters. The name can include the underscore character (_), but no other special characters, or any spaces. The name cannot start with a number.
RLINES *n* optional	The line spacing for detail records in views and reports. By default, detail lines are single spaced.
ALL optional	All fields in the active Analytics table layout are added to the view.
SUPPRESS optional	Suppresses blank detail lines in reports generated from the view. When the report is generated the blank detail lines will be omitted from the output. This option applies to reports based on multiline views.
SUMMARIZED optional	Specifies that reports generated from the view should include subtotals and totals, but not include the detail lines. The subtotals are generated based on the break fields defined in the view. If this option is not selected, the report includes detail lines, as well as subtotals for each of the specified break fields.
IF *test* optional	A conditional expression that must be true in order to process each record. The command is executed on only those records that satisfy the condition.

Name	Description
	Note The IF parameter is evaluated against only the records remaining in a table after any scope parameters have been applied (WHILE, FIRST, NEXT).
WHILE *test* optional	A conditional expression that must be true in order to process each record. The command is executed until the condition evaluates as false, or the end of the table is reached. **Note** If you use WHILE in conjunction with FIRST or NEXT, record processing stops as soon as one limit is reached.
HEADER *header_text* optional	The text to insert at the top of each page of a report. *header_text* must be specified as a quoted string. The value overrides the Analytics HEADER system variable.
FOOTER *footer_text* optional	The text to insert at the bottom of each page of a report. *footer_text* must be specified as a quoted string. The value overrides the Analytics FOOTER system variable.
TO *report_filename* HTML optional	The filename and type for reports created from this view. Use the HTML keyword to save reports generated from this view as HTML files (.htm). By default, generated reports are output as ASCII text files.
OK optional	Deletes or overwrites items without asking you to confirm the action.

Examples

Creating a view

You open the **Ar** table and create a view called **AR_Report**, which includes all of the fields in the table layout:

```
OPEN Ar
DEFINE VIEW AR_Report HEADER "AR Report" ALL OK
```

DELETE command

Deletes an Analytics project item, a field from a table layout, a variable, one or more table history entries, a relation between tables, or a file in a Windows folder. Also removes a column from a view.

Syntax

Purpose	Syntax
To delete an Analytics project item	`DELETE item_type item_name <OK>`
To delete a field from a table layout	`DELETE field_name <OK>`
To remove a column from a view	`DELETE COLUMN view_name field_name <ALL> <OK>`
To delete a variable or all variables	`DELETE {variable_name|ALL} <OK>`
To delete the history for the current Analytics table	`DELETE HISTORY <retain_history_entries> <OK>`
To delete a relation between two tables	`DELETE RELATION <child_table_name|relation_name> <OK>`
To delete a file	`DELETE file_name <OK>`
To delete all record notes, and the auto-generated **RecordNote** field, from the open table	`DELETE NOTES <OK>`

Parameters

Name	Description
item_type item_name	The type and name of the item to delete.

Name	Description
	Specify one of the following item types: ◦ **FOLDER** - the specified project folder and all its contents ◦ **FORMAT** - the specified table layout, all its views, and its associated indexes and relations Any other table layouts for the associated table are retained. The data file (.fil) associated with the table layout is not deleted unless the **Delete Data File with Table** option is selected in the **Table** tab in the **Options** dialog box (**Tools > Options**). You can also use the `SET DELETE_FILE {ON|OFF}` command in a script or on the command line to turn this option on or off. For more information, see "SET command" on page 1960. **Caution** Use caution when turning on the **Delete Data File with Table** option. It may be an original data file that is deleted along with the table layout. Data files are deleted outright. They are not sent to the Windows Recycle Bin. ◦ **REPORT** - the specified view You cannot delete a view if it is currently active. ◦ **COLUMN** - the specified column ◦ **SCRIPT (or BATCH)** - the specified script ◦ **WORKSPACE** - the specified workspace ◦ **INDEX** - the specified index ◦ **NOTES** - all record notes from the open table, and the **RecordNote** field from the table layout
field_name ALL	## Delete a field The name of the field to delete from the current Analytics table layout. You can delete a field from a table layout even if the field is included in the current view. **Note** You cannot delete a field referenced by a computed field unless you first delete the computed field. ## Remove a column The name of the column to remove from the specified view. **Note** Use the physical field name, not the column display name. ◦ **ALL included** - removes all occurrences of the column in the view ◦ **ALL omitted** - removes the first (leftmost) occurrence of the column in the view

Name	Description
view_name	The name of the view to remove a column from.
variable_name | ALL	The name of the variable to delete. Use ALL to delete all variables. If you specify ALL, all occurrences of the following types of the variables are deleted from the project: ◦ system variables ◦ temporary user-defined variables ◦ permanent user-defined variables **Note** You cannot delete a variable referenced by a computed field unless you first delete the computed field.
HISTORY *retain_history_entries*	Deletes all table history entries except for the number of most recent entries specified by *retain_history_entries*. Omit *retain_history_entries* to delete all entries.
RELATION *child_table_name* | *relation_name*	Deletes any relation that has no dependent relations and no related fields referenced in either the active view or in an active computed field. Use the options to specify which relation to delete: ◦ ***child_table_name*** - use when the relation was not specifically named (default name when a relation is created) ◦ ***relation_name*** - use when the relation was specifically named when it was created. Otherwise, use *child_table_name* If you do not use either option, the last relation that was defined gets deleted.
file_name	The name of a physical file to delete. You can specify an absolute or relative path to a file you want to delete. If the path has spaces, enclose it in double quotation marks.
OK optional	Deletes items without presenting a confirmation dialog box.

Examples

Deleting a date field

You delete the **Date** field from the table layout associated with the **Ar** table:

```
OPEN Ar
DELETE Date
```

Deleting multiple columns from a view

You delete two columns from the **AR_Report** view associated with the **Ar** table. You specify OK for both DELETE commands so that no confirmation prompt is displayed when the script runs:

```
OPEN Ar
DELETE COLUMN AR_Report Date OK
DELETE COLUMN AR_Report Invoice_Date OK
```

DIALOG command

Creates a custom dialog box that interactively prompts users for one or more script input values. Each input value is stored in a named variable.

> **Note**
>
> Using the DIALOG command to enter passwords is not secure. You should use the "PASSWORD command" on page 1893 instead.
>
> The DIALOG command is not supported in AX Server analytics.
>
> You can create a basic interactive dialog box with the "ACCEPT command" on page 1539.

> **Tip**
>
> The easiest way to create custom dialog boxes is with the **Dialog Builder**. For more information, see "Creating custom dialog boxes" on page 1508.

Syntax

```
DIALOG (DIALOG TITLE title_text WIDTH pixels HEIGHT pixels) (BUTTONSET TITLE
"&OK;&Cancel" AT x_pos y_pos <WIDTH pixels> <HEIGHT pixels> DEFAULT item_num
<HORZ>) <[label_syntax]|[text_box_syntax]|[check_box_syntax]|[radio_button_syn-
tax]|[drop_down_list_syntax]|[project_item_list_syntax]> <...n>
```

```
label_syntax ::=
(TEXT TITLE title_text AT x_pos y_pos <WIDTH pixels> <HEIGHT pixels>
<CENTER|RIGHT>)
```

```
text_box_syntax ::=
(EDIT TO var_name AT x_pos y_pos <WIDTH pixels> <HEIGHT pixels> <DEFAULT
string>)
```

```
check_box_syntax ::=
(CHECKBOX TITLE title_text TO var_name AT x_pos y_pos <WIDTH pixels> <HEIGHT
pixels> <CHECKED>)
```

```
radio_button_syntax ::=
(RADIOBUTTON TITLE value_list TO var_name AT x_pos y_pos <WIDTH pixels>
<HEIGHT pixels> <DEFAULT item_num> <HORZ>)
```

```
drop_down_list_syntax ::=
(DROPDOWN TITLE value_list TO var_name AT x_pos y_pos <WIDTH pixels> <HEIGHT
pixels> <DEFAULT item_num>)
```

```
project_item_list_syntax ::=
(ITEM TITLE project_item_category TO var_name AT x_pos y_pos <WIDTH pixels>
<HEIGHT pixels> <DEFAULT string>)
```

Parameters

General parameters

Name	Description
DIALOG TITLE *title_text*	Creates the main dialog box and the dialog box title. *title_text* must be specified as a quoted string.
BUTTONSET TITLE "&OK;&Cancel"	The labels for the **OK** and **Cancel** buttons in the dialog box. The values should normally not be edited, but if you do edit the values make sure that the positive value comes before the negative value. For example: `"&Yes;&No"`
WIDTH *pixels*	The width of the individual control, or the width of the dialog box if specified for the DIALOG control. The value is specified in pixels. If no value is specified for a control the width is calculated based on the longest value contained by the control.
HEIGHT *pixels*	The height of the individual control, or the height of the dialog box if specified for the DIALOG control. The value is specified in pixels.
AT *x_posy_pos*	The location of the top left corner of the control in the custom dialog box: ◦ *x_pos* is the horizontal distance in pixels from the left-hand side of the dialog box ◦ *y_pos* is the vertical distance in pixels from the top of the dialog box
DEFAULT *item_num*	The numeric value that corresponds to the BUTTONSET value that you want to select as the default. For example, if the BUTTONSET values are "&OK;&Cancel", specify `DEFAULT 1` to

Name	Description
	select OK by default.
HORZ optional	Displays the values for the BUTTONSET control horizontally. Values are displayed vertically by default.

> **Note**
>
> For most of the control types, the DIALOG command creates a variable to store user input. You cannot use non-English characters, such as é, in the names of variables that will be used in variable substitution. Variable names that contain non-English characters will cause the script to fail.
>
> By default, some of the DIALOG variables are created as character variables. If you use a character variable to store numeric or datetime values, you must convert the variable to the required data type in subsequent processing in a script. For more information, see "Input data type" on page 1659.

Label parameters

Name	Description
TEXT	Creates a text label to identify, notify, or instruct.
TITLE *title_text*	The control label. *title_text* must be specified as a quoted string.
CENTER | RIGHT optional	The alignment of the text in the control. If you omit CENTER or RIGHT, left alignment is used by default.

Text box parameters

Name	Description
EDIT	Creates a text box for user input.
TO *var_name*	The name of the character variable that stores the input specified by the user. If the variable already exists, the specified value is assigned. If the variable does not exist, it is created, and the specified value is assigned.
DEFAULT *string* optional	The default text string to display in the control. *string* must be specified as a quoted string.

Check box parameters

Name	Description
CHECKBOX	Creates a check box to present an option to the user.
TITLE *title_text*	The control label. *title_text* must be specified as a quoted string.
TO *var_name*	The name of the logical variable that stores the True or False value specified by the user. If the variable already exists, the specified value is assigned. If the variable does not exist, it is created, and the specified value is assigned.
CHECKED optional	Sets the control to checked by default.

Radio button parameters

Name	Description
RADIOBUTTON	Creates radio buttons to present mutually exclusive options to the user.
TITLE *value_list*	The list of values displayed for the control. The values must be specified as a quoted string. Separate each value with a semi-colon (;).
TO *var_name*	The name of the numeric variable that stores the numeric position of the radio button value selected by the user. If the variable already exists, the specified value is assigned. If the variable does not exist, it is created, and the specified value is assigned.
DEFAULT *item_num* optional	The numeric value that corresponds to the list item that you want to select as the default. For example, if *value_list* is "Red;Green;Blue", specify `DEFAULT 2` to select Green by default.
HORZ optional	Displays the values for the control horizontally. Values are displayed vertically by default.

Drop-down list parameters

Name	Description
DROPDOWN	Creates a drop-down list to present a list of options to the user.
TITLE *value_list*	The list of values displayed for the control. The values must be specified as a quoted string. Separate each value with a semi-colon (;).
TO *var_name*	The name of the character variable that stores the drop-down list value selected by the user. If the variable already exists, the specified value is assigned. If the variable does not exist, it is created, and the specified value is assigned.
DEFAULT *item_num* optional	The numeric value that corresponds to the list item that you want to select as the default. For example, if *value_list* is "Red;Green;Blue", specify `DEFAULT 2` to select Green by default when the drop-down list is displayed.

Project item list parameters

Name	Description
ITEM	Creates a project item list to present a list of Analytics project items, such as fields, to the user.
TITLE *project_item_category*	The category of project item to include in the control. You can specify one or more categories. The user can select a single value from the project item list. Enclose *project_item_category* in quotation marks, with no space or punctuation between categories. For the codes used to specify categories, see "Codes for project item categories" on page 1658. **Note** Do not mix dissimilar categories in the same ITEM control, unless you have a reason for doing so. For example, do not mix tables and fields. The resulting project item list is potentially confusing for the user.
TO *var_name*	The name of the character variable that stores the name of the project item selected by the user. If the variable already exists, the specified value is assigned. If the variable does not exist, it is created, and the specified value is assigned.

Name	Description
DEFAULT *string* optional	The exact name of the project item that you want to select as the default. *string* must be specified as a quoted string.

Examples

Prompting the user for a table and script

In your script, you need to prompt the user to select the Analytics table and script to use to run an analysis .

You specify that the **Metaphor_Inventory_2012** table from the `ACL_Demo.acl` project is selected by default as the Analytics table, but the user can select any table in the project.

The script to run must also be selected from the list of scripts in the Analytics project:

```
DIALOG (DIALOG TITLE "Inventory analysis" WIDTH 500 HEIGHT 200 )
(BUTTONSET TITLE "&OK;&Cancel" AT 370 12 DEFAULT 1 ) (TEXT TITLE "Choose
the Analytics project items to analyze." AT 50 16 ) (TEXT TITLE "Table:"
AT 50 50 ) (ITEM TITLE "f" TO "v_table" AT 50 70 DEFAULT "Metaphor_
Inventory_2012" ) (TEXT TITLE "Script:" AT 230 50 ) (ITEM TITLE "b" TO
"v_script" AT 230 70 )
```

Additional examples

For additional DIALOG examples, see "Example script: filter records by date, and group filtered records by month" on page 1462.

Remarks

For more information about how this command works, see "Creating custom dialog boxes" on page 1508.

Interactivity

Use DIALOG to create an interactive script. When the DIALOG command is processed, the script pauses and a dialog box is displayed that prompts the user for input that Analytics uses in subsequent processing.

You can create separate dialog boxes that prompt for one item at a time, or you can create one dialog box that prompts for multiple items.

ACCEPT versus DIALOG

The ACCEPT command allows you to create a basic interactive dialog box that can have one or more of the following types of controls:

- text box
- project item list

For basic interactivity, ACCEPT may be all you need. For more information, see "ACCEPT command" on page 1539.

Codes for project item categories

Use the following codes to specify the category of project item to display in a project item list.

Project categories

Code	Category
f	Tables
b	Scripts
i	Indexes
r	Views and reports
w	Workspaces

Field categories

Code	Category
C	Character fields
N	Numeric fields

Code	Category
D	Datetime fields
L	Logical fields

Variable categories

Code	Category
c	Character variables
n	Numeric variables
d	Datetime variables
l	Logical variables

Input data type

Some of the controls in the DIALOG command store user input in character variables. If you need numeric or datetime input, you can use the VALUE( ) or CTOD( ) functions to convert the contents of a character variable to a numeric or datetime value:

```
SET FILTER TO BETWEEN(%v_date_field%, CTOD(%v_start_date%), CTOD(%v_end_
date%))
```

In the example, the start and end dates for this filter are stored as character values. They must be converted to date values in order to be used with a date field that uses a Datetime data type.

Enclosing the variable name in percent signs (%) substitutes the character value contained by the variable for the variable name. The CTOD( ) function then converts the character value to a date value.

Position of the DIALOG command

It is good practice to place all DIALOG commands at the beginning of a script, if possible. If you ask for all input at the beginning, the script can then run unimpeded once the user enters the necessary information.

> **Note**
> You cannot use the DIALOG command inside the GROUP command.

DIRECTORY command

Generates a list of files and folders in the specified directory.

Syntax

```
DIRECTORY <file_spec> <SUPPRESS> <SUBDIRECTORY> <APPEND> <TO table_name|file-
name>
```

Parameters

Name	Description
file_spec optional	The Windows folder or files to list and display information for. You can use the asterisk wildcard (*) to list all files with a particular extension, all files that start with a particular string, or all files in a folder. For example: ◦ `*.fil` - lists all files with the .fil extension (Analytics data files) ◦ `Inv*.*` - lists all files that begin with "Inv" regardless of what file extension they have ◦ `Results\*` or `Results\*.*` - lists all files in the Results folder To limit the files listed to a particular folder, you can specify a path relative to the Analytics project folder, or specify a full path. For example: ◦ `Results\*.*` - displays the contents of the Results subfolder in the Analytics project folder ◦ `C:\ACL Data\Results\*.*` - displays the contents of the specified folder **Note** The wildcard character cannot be used in intermediary levels of a specified file path. It can only be used in the final level of the path, as shown above. Paths or file names that contain spaces must be enclosed in double quotation marks. If you use *file_spec*, it must be placed before any of the other parameters. If *file_spec* appears in any other position, the DIRECTORY command is not processed and an error is generated. If you omit *file_spec*, all files in the folder containing the Analytics project are listed. You cannot use any of the other parameters if you omit *file_spec*.
SUPPRESS optional	Suppresses path information in the output, leaving only the file names and properties.

Name	Description
SUBDIRECTORY optional	Includes the contents of subfolders in the directory listing. For example, if *file_spec* specifies `Results\*.fil`, the Results folder, and all subfolders contained in the Results folder, are searched for `.fil` files. Depending on the number of subfolders and files that need to be listed, using SUBDIRECTORY may result in a delay while the subfolders are searched. Analytics displays a dialog box showing progress of the command.
APPEND optional	Appends the command output to the end of an existing file instead of overwriting it. **Note** You must ensure that the structure of the command output and the existing file are identical: • the same fields • the same field order • matching fields are the same length • matching fields are the same data type Analytics appends output to an existing file regardless of its structure. If the structure of the output and the existing file do not match, jumbled, missing, or inaccurate data can result.
TO *table_name* | *filename* optional	The location to send the results of the command to: ◦ ***table_name*** - saves the results to an Analytics table Specify *table_name* as a quoted string with a .FIL file extension. For example: `TO "Output.FIL"` By default, the table data file (.FIL) is saved to the folder containing the Analytics project. Use either an absolute or relative file path to save the data file to a different, existing folder: • `TO "C:\Output.FIL"` • `TO "Results\Output.FIL"` **Note** Table names are limited to 64 alphanumeric characters, not including the .FIL extension. The name can include the underscore character (_), but no other special characters, or any spaces. The name cannot start with a number. ◦ ***filename*** - saves the results to a file Specify *filename* as a quoted string with the appropriate file extension. For example: `TO "Output.TXT"` By default, the file is saved to the folder containing the Analytics project. Use either an absolute or relative file path to save the file to a different, existing folder: • `TO "C:\Output.TXT"` • `TO "Results\Output.TXT"` If you omit TO, the directory listing appears in the Analytics display area.

Examples

Different options for listing files

The ability to list files is useful for ad hoc investigation, and for incorporating in scripting.

A number of different options for listing files with the DIRECTORY command appear below.

List all files

Lists all the files in the folder containing the Analytics project:

```
DIRECTORY
```

List all the files of a specific type

Lists all the .fil files (Analytics data files) in the folder containing the Analytics project:

```
DIRECTORY *.fil
```

Use wildcards to list files

Lists all the file names beginning with "Inv" in the folder containing the Analytics project:

```
DIRECTORY Inv*.*
```

List all the files in a subfolder relative to the Analytics project folder

Lists all the files in the `Results` subfolder in the folder containing the Analytics project:

```
DIRECTORY "Results\*"
```

List all the files in a specified folder

Lists all the files in the `Results` subfolder:

```
DIRECTORY "C:\ACL Data\Results\*"
```

List all the files of a specific type in a specified location

Lists all the .fil files (Analytics data files) in the specified folder and any subfolders:

```
DIRECTORY "C:\ACL Data\Results\*.fil" SUBDIRECTORY
```

List all the files in a specified folder and output the list to an Analytics table

Lists all the files in the `Results` folder and outputs the list to an Analytics table in the folder containing the Analytics project:

```
DIRECTORY "C:\ACL Data\Results\*" TO Results_Folder_Contents.fil
```

The new table `Results_Folder_Contents` is added to the open project.

List all the files in one folder and output the list to an Analytics table in another folder

Lists all the files in the `ACL Data\Results` folder and outputs the list to an Analytics table in the `GL Audit 2014\Results` folder:

```
DIRECTORY "C:\ACL Data\Results\*" TO "C:\ACL Projects\GL Audit 2014\Res-
ults\Results_Folder_Contents.fil"
```

The new table `Results_Folder_Contents` is added to the open project. The associated data file (`Results_Folder_Contents.fil`) is created in the specified output folder, which may or may not be the folder containing the Analytics project.

Remarks

Properties displayed by DIRECTORY

The DIRECTORY command is similar to the DIR command in Windows. In addition to listing files and subfolders in a folder, the DIRECTORY command also displays the following file and folder properties:

◦ File Size ◦ Attributes	◦ Create Date ◦ Create Time	◦ Access Date ◦ Access Time	◦ Modified Date ◦ Modified Time ◦ the total number of files and folders that match the specified criteria

Uses for DIRECTORY in a script

When used in a script, the DIRECTORY command provides the ability to examine the file system. For example, you could use DIRECTORY in conjunction with other commands to detect the presence or absence of files, check a file's size, or make decisions based on other file properties.

Outputting the results of DIRECTORY

You can run the command from the command line to display a directory listing on screen, or save the listing to an Analytics table or .txt file.

How to open table-based results of DIRECTORY

The DIRECTORY command does not include the OPEN parameter. If you are using the command in a script and outputting the results to an Analytics table, and you want to open the resulting table,

follow the DIRECTORY command with the OPEN command. For example:

```
DIRECTORY "C:\ACL Data\Results\*" TO Results_Folder_Contents.fil
OPEN Results_Folder_Contents
```

DISPLAY command

Displays information about the specified Analytics item type. Can also display the result of an expression, or the output of a function.

Syntax and parameters

Syntax	Purpose
`DISPLAY`	Displays the field definitions, and any related child tables, for the currently active Analytics table.
`DISPLAY OPEN`	Displays a list of open Analytics tables and project files. ◦ **Analytics tables** - displays the name of the source data file, not the table layout name. ◦ **multiple-table mode** - the source data file identified as PRIMARY is associated with the currently active table. ◦ **related tables** - if the parent table is open, displays the source data file for both the parent and the child tables, even if the child table is not open in the View tab.
`DISPLAY {<PRIMARY>|SECONDARY}`	Displays the name and table layout information for the primary or secondary table. ◦ **PRIMARY (or no keyword specified)** - display information for the currently active table. ◦ **SECONDARY** - display information for the secondary table. In multiple-table mode, SECONDARY refers to the secondary table associated with the currently active table. The information displayed includes: ◦ the table layout name ◦ the source data file name ◦ any relations between the table and other tables ◦ the field definition information from the table layout
`DISPLAY HISTORY`	Displays the table history for the currently active Analytics table.

Syntax	Purpose
	Note A table may or may not have associated table history.
`DISPLAY RELATION`	Displays relation information for the currently active Analytics table: ◦ the names of any child tables ◦ key field names ◦ index names
`DISPLAY {variable_name|VARIABLES}`	Displays the value of a single variable or all variables. ◦ ***variable_name*** - the name of a single variable to display the value of. ◦ **VARIABLES** - display the values of all system and user-defined variables, and the remaining memory available to store variables.
`DISPLAY VERSION`	Displays information in the following format about the installed version of Analytics: ◦ **Version** - *major version number.minor version number* ◦ **Patch** - *patch number* ◦ **Type** - 000 (non-Unicode), or 001 (Unicode) edition of Analytics ◦ **Build** - *software build number*
`DISPLAY {DATE|TIME}`	Displays the current operating system date and time. **DATE | TIME** - specify either keyword. The two keywords do the same thing.
`DISPLAY {FREE|SPACE}`	Displays the amount of physical memory (RAM) available for use by Analytics. The amount displayed does not include memory reserved for variables. By default, Analytics reserves 60 KB of physical memory to store variables, but the amount is automatically increased as necessary. **FREE | SPACE** - specify either keyword. The two keywords do the same thing.
`DISPLAY expression`	Displays the result of an expression. ***expression*** - the expression to display the result of.
`DISPLAY function`	Displays the output of a function. ***function*** - the function to display the output of.

Examples

Display the layout of an Analytics table

Displaying the layout of a table can be useful in a number of circumstances. For example, you may want to combine two or more tables, and you need to examine field lengths and data types.

The example below displays the layout of the Ap_Trans table:

```
OPEN Ap_Trans
DISPLAY
```

The DISPLAY command produces the output to screen shown below.

> **Note**
>
> If you enter `DISPLAY` directly in the Analytics command line, the output appears immediately.
>
> If you run `DISPLAY` in a script, double-click the corresponding **DISPLAY** entry in the command log to display the output.

Output to screen

Relationship

'Vendor' related by 'Vendor_No' using index 'Vendor_on_Vendor_No'

File

'Ap_Trans.fil' (format 'Ap_Trans') is your PRIMARY file.

The record length is 59

Fields

Name	Type	Start	Length	Decimals	Field explanation
Vendor_No	ASCII	1	5		AS "Vendor;Number" WIDTH 7
Invoice_No	ASCII	6	15		AS "Invoice;Number"
Invoice_ Date	DATETIME	21	8		PICTURE "MM/DD/YY" AS "Invoice;Date" WIDTH 8

Name	Type	Start	Length	Decimals	Field explanation
Invoice_ Amount	NUMERIC	29	12	2	PICTURE "(9,999,999.99)" AS "Invoice;Amount" WIDTH 12
Prodno	ASCII	41	9		AS "Product;Number"
Quantity	MICRO	50	4	0	PICTURE "(9,999,999)"
Unit_Cost	NUMERIC	54	6	2	PICTURE "(9,999,999)" AS "Unit;Cost" SUPPRESS

Display the values of all variables in an Analytics project

DISPLAY VARIABLES generates the same information that appears in the **Variables** tab in the **Navigator**. One benefit of using DISPLAY VARIABLES is that you can copy-and-paste the displayed information.

The example below creates two user-defined variables and two system variables and then displays the values of the variables:

```
ASSIGN v_table_name = "Ap_Trans"
ASSIGN v_field_name = "Invoice_Amount"
OPEN %v_table_name%
TOTAL FIELDS %v_field_name%
DISPLAY VARIABLES
```

The DISPLAY command produces the output to screen shown below.

> **Note**
>
> If you enter `DISPLAY VARIABLES` directly in the Analytics command line, the output appears immediately.
>
> If you run `DISPLAY VARIABLES` in a script, double-click the corresponding **DISPLAY VARIABLES** entry in the command log to display the output.

Output to screen

Name	Type	Value
TOTAL1	N	278,641.33
OUTPUTFOLDER	C	"/Tables/Accounts_Payable"
v_field_name	C	"Invoice_Amount"
v_table_name	C	"Ap_Trans"

Display the result of an expression

For the selected record, the example below displays the result of multiplying the value in the Sale_Price field by the value in the Quantity_on_Hand field:

```
DISPLAY Sale_Price * Quantity_on_Hand
```

Display the output of a function

For the selected record, the example below displays the number of days that have elapsed since the date in the Invoice_Date field:

```
DISPLAY AGE(Invoice_Date)
```

Remarks

Location of command results

DISPLAY run from the Analytics command line – the results are displayed on screen.

DISPLAY executed in a script – the results are written to the Analytics command log. You can double-click the command log entry to display the results on screen.

DO REPORT command

Generates the specified Analytics report.

Syntax

```
DO REPORT report_name
```

Parameters

Name	Description
report_name	The name of the view to generate and print as a report.

Example

Printing the default view

You open the **AP_Trans** table and print the default view:

```
OPEN AP_Trans
DO REPORT Default_View
```

Remarks

Running DO REPORT on the command line vs in a script

The settings used to print the report depend on where you run the command:

- **from the command line** – the **Print** dialog box opens for you to select the pages to print and configure other options for the report
- **in a script** – the report is printed immediately using the default settings for the report

DO SCRIPT command

Executes a secondary script, or an external script, from within an Analytics script.

Syntax

```
DO <SCRIPT> script_name {<IF test>|<WHILE test>}
```

Parameters

Name	Description
SCRIPT *script_name*	The name of the script to run. You can run secondary scripts in the Analytics project, or external scripts stored in text files with extensions such as .aclscript, .txt. or .bat. You can specify a file path to an external script. You must enclose the path in quotation marks if it contains any spaces. **Note** You cannot call a script that is already running. For example, if ScriptA calls ScriptB, ScriptB cannot call ScriptA. ScriptA is still running while it waits for ScriptB to complete.
IF *test* optional	A conditional expression that is evaluated one time to determine if the script should be executed. If the condition evaluates to true the script runs, otherwise it does not. Cannot be used with WHILE in the same command. If both are used, WHILE is ignored when the script is processed. A comment is entered in the log, but the script does not stop executing.
WHILE *test* optional	A conditional expression that is evaluated after the script runs to determine if the script should be executed again. If the test evaluates to true the script runs again, otherwise it does not. **Note** If you use WHILE, ensure that your test eventually evaluates to false. If you do not, the script enters an infinite loop. In case of an infinite loop, press the **Esc** key to cancel script processing. Cannot be used with IF in the same command. If both are used, WHILE is ignored when the script is processed. A comment is entered in the log, but the script does not stop executing.

Examples

Executing a subscript repeatedly until the input is validated

You have a subscript that gathers user input using a dialog box. It does the following:

1. Prompts the user for the required values.
2. Checks the user input.
3. Sets the *v_validated* variable to true when the input values are validated.

To ensure that the user enters valid input, you use DO SCRIPT and include a WHILE condition so that the script repeats this command until input is validated. Once the value of the variable changes, the main script moves on to the next command:

```
DO SCRIPT GetUserInput WHILE v_validated = F
```

Running a subscript from a shared location

You maintain utility subscripts in a shared location. When you require one of the subscripts during an analysis, you reference it using the full path to the shared location:

```
DO SCRIPT "C:\My utility scripts\GetUserInput.aclscript"  WHILE v_val-
idated = F
```

Remarks

Related commands

DO SCRIPT is the equivalent of the DO BATCH command found in scripts created with earlier releases of Analytics.

You cannot include the DO SCRIPT command inside a GROUP command.

Usefulness of an external script

Storing a script externally and calling it from within an Analytics script is useful if you want to reuse the same subscript in different Analytics scripts and projects.

You can store a single copy of the script in one location, and make updates to it in one place, rather than maintaining it in multiple locations.

DUMP command

Displays the contents of a file, or the current record, in hexadecimal, ASCII, and EBCDIC character encodings.

> **Note**
> This command can only be entered in the command line. It cannot be used in scripts.

Syntax

```
DUMP {RECORD|file_name} <SKIP bytes> <COLUMN bytes> <HORIZONTAL>
```

Parameters

Name	Description
RECORD	Displays the contents of the selected record. Required if you do not specify a *file_name*.
file_name	The name of the file you want to display. Required if you do not specify RECORD. **Note** To display the character encodings for an Analytics table you must specify the name of the source data file and the file extension. For example: `Ap_Trans.fil`
SKIP *bytes* optional	The number of bytes to bypass before the dump begins. The default is 0.
COLUMN *bytes* optional	In the output, the width of the columns in bytes. **Note** The number specified by *bytes* refers to the bytes contained by the Analytics record or table. The encoded characters in the output may not have a one-to-one relation with the characters in the view. For example, the hexadecimal encoding for the number 1 is `31`. The default is 16 bytes for each column in a vertical display, and 64 bytes for the single column in a horizontal display. The maximum number of bytes you can

Name	Description
	specify is 255.
HORIZONTAL optional	Displays the character encodings in horizontal rows rather than in side-by-side vertical blocks (the default).

Examples

Display the character encoding of the Inventory table

The example below displays the hexadecimal, ASCII, and EBCDIC character encoding of the data in the Inventory table. The output is arranged in horizontal rows (hexadecimal encoding uses a double row). Each row represents 97 bytes of data in the Analytics table:

```
DUMP Inventory.fil COLUMN 97 HORIZONTAL
```

DUPLICATES command

Detects whether duplicate values or entire duplicate records exist in an Analytics table.

Syntax

```
DUPLICATES {<ON> key_field <D> <...n>|<ON> ALL <EXCLUDE field_name <...n>>}
<OTHER field <...n>|OTHER ALL <EXCLUDE field_name <...n>>> <UNFORMATTED>
<ADDGROUP> <PRESORT> <IF test> <WHILE test> <FIRST range|NEXT range> <APPEND>
<OPEN> <TO {SCREEN|table_name|filename|PRINT}> <LOCAL> <HEADER header_text>
<FOOTER footer_text> <ISOLOCALE locale_code>
```

Parameters

Name	Description
ON *key_field* D <...*n*> | ON ALL	The key field or fields, or the expression, to test for duplicates. ◦ **ON *key_field*** - use the specified field or fields If you test by more than one field, records identified as duplicates require identical values in every specified field. Fields are included in the output results in the order that you list them. Include D to sort a key field in descending order. The default sort order is ascending. ◦ **ON ALL** - use all fields in the table If you test by all the fields in a table, records identified as duplicates must be entirely identical. Fields are included in the output results in the order that they appear in the table layout. An ascending sort order is the only option for ON ALL. **Note** Undefined portions of records are not tested.
EXCLUDE *field_name* optional	Only valid when testing for duplicates using ON ALL. The field or fields to exclude from the command. EXCLUDE allows you to fine-tune ON ALL, by excluding the specified fields. EXCLUDE must immediately follow ON ALL. For example:

Name	Description
	`ON ALL EXCLUDE field_1 field_2`
OTHER *field <...n>* | OTHER ALL optional	One or more additional fields to include in the output. ◦ **OTHER *field <...n>*** - include the specified field or fields ◦ **OTHER ALL** - include all fields in the table that are not specified as key fields
EXCLUDE *field_name* optional	Only valid when using OTHER ALL. The field or fields to exclude from the command. EXCLUDE allows you to fine-tune OTHER ALL, by excluding the specified fields. EXCLUDE must immediately follow OTHER ALL. For example: `OTHER ALL EXCLUDE field_1 field_2`
UNFORMATTED optional	Suppresses page headings and page breaks when the results are output to a file.
ADDGROUP optional	Include the **Group Number** field (`GROUP_NUM`) in the output table. The **Group Number** field assigns a sequentially incremented number to each unique group of duplicates. **Tip** The ability to reference groups of duplicates by number can be useful when you analyze data in the output table.
PRESORT optional	Sorts the table on the key field before executing the command. **Note** You cannot use PRESORT inside the GROUP command.
IF *test* optional	A conditional expression that must be true in order to process each record. The command is executed on only those records that satisfy the condition. **Note** The IF parameter is evaluated against only the records remaining in a table after any scope parameters have been applied (WHILE, FIRST, NEXT).
WHILE *test* optional	A conditional expression that must be true in order to process each record. The command is executed until the condition evaluates as false, or the end of the table is reached. **Note** If you use WHILE in conjunction with FIRST or NEXT, record processing stops as soon as one limit is reached.

Name	Description
FIRST *range* | NEXT *range* optional	The number of records to process: ◦ **FIRST** - start processing from the first record until the specified number of records is reached ◦ **NEXT** - start processing from the currently selected record until the specified number of records is reached Use *range* to specify the number of records to process. If you omit FIRST and NEXT, all records are processed by default.
APPEND optional	Appends the command output to the end of an existing file instead of overwriting it. **Note** You must ensure that the structure of the command output and the existing file are identical: • the same fields • the same field order • matching fields are the same length • matching fields are the same data type Analytics appends output to an existing file regardless of its structure. If the structure of the output and the existing file do not match, jumbled, missing, or inaccurate data can result.
OPEN optional	Opens the table created by the command after the command executes. Only valid if the command creates an output table.
TO SCREEN | *table_name* | *filename* | PRINT optional	The location to send the results of the command to: ◦ **SCREEN** - displays the results in the Analytics display area **Tip** You can click any linked result value in the display area to drill down to the associated record or records in the source table. ◦ ***table_name*** - saves the results to an Analytics table Specify *table_name* as a quoted string with a .FIL file extension. For example: `TO "Output.FIL"` By default, the table data file (.FIL) is saved to the folder containing the Analytics project. Use either an absolute or relative file path to save the data file to a different, existing folder: • `TO "C:\Output.FIL"` • `TO "Results\Output.FIL"` **Note** Table names are limited to 64 alphanumeric characters, not including the .FIL extension. The name can include the underscore character (_), but no other special characters, or any spaces. The name cannot start with a number.

Name	Description
	◦ ***filename*** - saves the results to a file Specify *filename* as a quoted string with the appropriate file extension. For example: `TO "Output.TXT"` By default, the file is saved to the folder containing the Analytics project. Use either an absolute or relative file path to save the file to a different, existing folder: • `TO "C:\Output.TXT"` • `TO "Results\Output.TXT"` ◦ **PRINT** - sends the results to the default printer
LOCAL optional	Saves the output file in the same location as the Analytics project. **Note** Applicable only when running the command against a server table with an output file that is an Analytics table. The LOCAL parameter must immediately follow the TO parameter.
HEADER *header_text* optional	The text to insert at the top of each page of a report. *header_text* must be specified as a quoted string. The value overrides the Analytics HEADER system variable.
FOOTER *footer_text* optional	The text to insert at the bottom of each page of a report. *footer_text* must be specified as a quoted string. The value overrides the Analytics FOOTER system variable.
ISOLOCALE *locale_code* optional	**Note** Applicable in the Unicode edition of Analytics only. The system locale in the format *language_country*. For example, to use Canadian French, enter `fr_ca`. Use the following codes: ◦ **language** - ISO 639 standard language code ◦ **country** - ISO 3166 standard country code If you do not specify a country code, the default country for the language is used. If you do not use ISOLOCALE, the default system locale is used.

Analytics output variables

Name	Contains
GAPDUP*n*	The total number of gaps, duplicates, or fuzzy duplicate groups identified by the command.

Examples

Test for duplicate values in one field

The following example:

- tests for duplicate values in the **Invoice_Number** field
- outputs any records that contain duplicate invoice numbers to a new Analytics table

```
DUPLICATES ON Invoice_Number OTHER Vendor_Number Invoice_Date Invoice_
Amount PRESORT TO "Duplicate_Invoices.FIL"
```

Test for duplicate values in two or more fields in combination

The following example:

- tests for duplicate combinations of values in the **Invoice_Number** and **Vendor_Number** fields
- outputs any records that contain the same invoice number and the same vendor number to a new Analytics table

The difference between this test and the previous test is that a identical invoice number from two different vendors is not reported as a false positive.

```
DUPLICATES ON Invoice_Number Vendor_Number OTHER Invoice_Date Invoice_
Amount PRESORT TO "Duplicate_Invoices.FIL"
```

Test for duplicate records

The following examples:

- test for duplicate values in every field in an Inventory table
- output any entirely identical records to a new Analytics table

```
DUPLICATES ON ProdNum ProdClass Location ProdDesc ProdStatus UnitCost
CostDate SalePrice PriceDate PRESORT TO "Duplicate_Inventory_Items.FIL"
```

You can simplify the syntax by using `ALL`:

```
DUPLICATES ON ALL PRESORT TO "Duplicate_Inventory_Items.FIL"
```

Filter duplicates output table by group number

You use several key fields in combination to test an accounts payable table for duplicate records:

- vendor number
- invoice number
- invoice date
- invoice amount

You want to filter the resulting duplicates output table so that only some of the groups of duplicates are subject to additional processing.

To create a filter using the combination of key fields would be laborious. For example:

```
SET FILTER TO ((Vendor_No = "11475") AND (Invoice_No = "8752512") AND
(Invoice_Date = `20191021`) AND (Invoice_Amount = 7125.80)) OR ((Vendor_
No = "12130") AND (Invoice_No = "589134") AND (Invoice_Date =
`20191117`) AND (Invoice_Amount = 10531.71)) OR ((Vendor_No = "13440")
AND (Invoice_No = "5518912") AND (Invoice_Date = `20191015`) AND
(Invoice_Amount = 11068.20))
```

Instead, you achieve the same result by creating a filter based on group number:

```
SET FILTER TO MATCH(GROUP_NUM, 3 , 8, 11)
```

Remarks

For more information about how this command works, see "Testing for duplicates" on page 1204.

Sorting and duplicates

Generally, you should run the duplicates command only on a sorted key field or fields. Duplicate values in a key field are only found if they are immediately adjacent.

If you run the duplicates command on an unsorted key field, non-adjacent duplicate values are not reported as duplicates. If two or more clusters of the same duplicate value exist, they are reported as duplicates, but in separate groups.

Depending on your analysis goal, it may make sense to run the duplicates command on an unsorted key field. For example, you may want to find only those duplicate values that are immediately adjacent in the source table, and ignore duplicate values that are non-adjacent.

ESCAPE command

Terminates the script being processed, or all scripts, without exiting Analytics.

Syntax

```
ESCAPE <ALL> <IF test>
```

Parameters

Name	Description
ALL optional	Terminates all active scripts. If omitted, the current script is terminated.
IF *test* optional	A test that must evaluate to true before the command is executed. If the test evaluates to false the command does not run.

Examples

Terminating a script conditionally

You count the number of records in a table, and use the ESCAPE command to terminate the script if the number of records is less than 100:

```
COUNT
ESCAPE IF COUNT1 < 100
```

Remarks

When to use ESCAPE

Use ESCAPE to halt the execution of a script or subscript based on a condition, or to stop the execution of all running scripts.

Using ESCAPE in subscripts

If you execute ESCAPE inside a subscript, then the subscript stops executing and the main script resumes processing from the DO SCRIPT command that invoked the subscript.

If you include the ALL option in the ESCAPE command in the subscript, then both the subscript and the main script stop processing:

```
ESCAPE ALL
```

EVALUATE command

For record sampling or monetary unit sampling, projects errors found in sampled data to the entire population, and calculates upper limits on deviation rate, or misstatement amount.

Record samplingMonetary unit sampling

Syntax

```
EVALUATE RECORD CONFIDENCE confidence_level SIZE sample_size ERRORLIMIT number_of_errors <TO {SCREEN|filename}>
```

Parameters

> **Note**
> Do not include thousands separators, or percentage signs, when you specify values.

Name	Description
RECORD	Evaluate errors found in a record sample.
CONFIDENCE *confidence_level*	The same confidence level that you specified when you calculated the sample size.
SIZE *sample_size*	The number of records in the sample. **Note** Specify the actual sample size as drawn, which might differ from the sample size initially calculated by Analytics.
ERRORLIMIT *number_of_errors*	The total number of errors, or deviations, that you found in the sample.
TO SCREEN | *filename*	The location to send the results of the command to: ◦ **SCREEN** - displays the results in the Analytics display area **Tip** You can click any linked result value in the display area to drill down to the associated record or records in the source table.

Name	Description
	◦ ***filename*** - saves the results to a file Specify *filename* as a quoted string with the appropriate file extension. For example: `TO "Output.TXT"` By default, the file is saved to the folder containing the Analytics project. Use either an absolute or relative file path to save the file to a different, existing folder: • `TO "C:\Output.TXT"` • `TO "Results\Output.TXT"`

Analytics output variables

Name	Contains
MLE*n*	The Upper error limit frequency rate (computed upper deviation rate) calculated by the command.

Examples

Project errors found in the sampled data to the entire population

You have completed your testing of the sampled data and recorded the control deviations you found. You can now project the errors you found to the entire population.

The example below projects two errors found in the sampled data to the entire population, and calculates **an upper error limit frequency rate** (computed upper deviation rate) of 6.63%.

```
EVALUATE RECORD CONFIDENCE 95 SIZE 95 ERRORLIMIT 2 TO SCREEN
```

For a detailed explanation of how Analytics calculates values when evaluating errors, see "Evaluating errors in a record sample" on page 988.

Remarks

For more information about how this command works, see "Evaluating errors in a record sample" on page 988.

Syntax

```
EVALUATE MONETARY CONFIDENCE confidence_level <ERRORLIMIT book_value, mis-
statement_amount <,...n>> INTERVAL interval_value <TO {SCREEN|filename}>
```

Parameters

> **Note**
> Do not include thousands separators, or percentage signs, when you specify values.

Name	Description
MONETARY	Evaluate errors found in a monetary unit sample.
CONFIDENCE *confidence_level*	The same confidence level that you specified when you calculated the sample size.
ERRORLIMIT *book_value,misstatement_amount*	All misstatement errors that you found in the sample. Specify the book value of the amount and the misstatement amount, separated by a comma. For example, if an amount has a book value of $1,000 and an audit value of $930, specify `1000,70`. Specify overstatements as positive amounts, and understatements as negative amounts. For example, if an amount has a book value of $1,250 and an audit value of $1,450, specify `1250,-200`. Separate multiple *book_value, misstatement_amount* pairs with a comma: `1000,70,1250,-200`
INTERVAL *interval_value*	The interval value that you used when you drew the sample. **Note** The interval value that you used might differ from the interval value initially calculated by Analytics.
TO SCREEN | *filename*	The location to send the results of the command to:

Name	Description
	◦ **SCREEN** - displays the results in the Analytics display area **Tip** You can click any linked result value in the display area to drill down to the associated record or records in the source table. ◦ ***filename*** - saves the results to a file Specify *filename* as a quoted string with the appropriate file extension. For example: `TO "Output.TXT"` By default, the file is saved to the folder containing the Analytics project. Use either an absolute or relative file path to save the file to a different, existing folder: • `TO "C:\Output.TXT"` • `TO "Results\Output.TXT"`

Analytics output variables

Name	Contains
MLE*n*	The Most Likely Error amount (projected misstatement) calculated by the command.
UEL*n*	The Upper Error Limit amount (upper misstatement limit) calculated by the command.

Examples

Project errors found in the sampled data to the entire population

You have completed your testing of the sampled data and recorded the misstatements you found. You can now project the errors you found to the entire population.

The example below projects three errors found in the sampled data to the entire population, and calculates several values, including:

- **Basic Precision** - the basic allowance for sampling risk (18,850.00)
- **Most Likely Error** - the projected misstatement amount for the population (1,201.69)
- **Upper Error Limit** - the upper misstatement limit for the population (22,624.32)

```
EVALUATE MONETARY CONFIDENCE 95 ERRORLIMIT 1000,70,1250,-200,3200,900
INTERVAL 6283.33 TO SCREEN
```

For a detailed explanation of how Analytics calculates values when evaluating errors, see "Evaluating errors in a monetary unit sample" on page 1022.

Remarks

For more information about how this command works, see "Evaluating errors in a monetary unit sample" on page 1022.

EXECUTE command

Executes an application or process external to Analytics. Emulates the Windows Run command. Can be used to interact with the Windows command prompt.

> **Note**
>
> Because the EXECUTE command gives you the ability to interact with the operating system and applications external to Analytics, technical issues may arise that are beyond the scope of Analytics's native functionality.
>
> Support can assist with operation of the EXECUTE command inside Analytics, but issues that arise with processes and applications external to Analytics are not covered under Support.

Syntax

```
EXECUTE Windows_Run_command_syntax <ASYNC>
```

Parameters

Name	Description
Windows_Run_command_syntax	The name of the application to execute, the folder or file to open, or the command to run, followed by any required arguments or command switches. Requires valid Windows Run command syntax enclosed by quotation marks.
ASYNC optional	Runs the command in asynchronous mode. In asynchronous mode, the Analytics script continues running without waiting for the process started by the EXECUTE command to complete. If you omit ASYNC, then the process started by the EXECUTE command must complete before the Analytics script continues. Analytics is inaccessible while external processes are running. **Note** When running EXECUTE from the Analytics command line, you must specify ASYNC.

Analytics output variables

Name	Contains
RETURN_CODE	The code returned by an external application or process run using the EXECUTE command. **What are return codes?** Return codes are numbers generated by the external application or process and sent back to Analytics to indicate the outcome of the external process. Analytics does not generate the return code, it only receives it. **Typical return codes** Typical return codes are integer values that map to specific notifications or error messages. For example, the return code "0" could mean "The operation completed successfully". The return code "2" could mean "The system cannot find the file specified". **The meaning of specific return codes** Specific return codes and their meanings vary depending on the external application or process. Lists of return codes, also called 'error codes' or 'exit codes', and their meanings, can often be found in the documentation for the associated external application. Lists of return codes can also be found on the Internet. **Variable created in default mode only** The RETURN_CODE variable is created when the EXECUTE command is run in default mode. The variable is not created when the command is run in asynchronous mode.

Examples

Open an application

Opens Microsoft Excel:

```
EXECUTE "Excel"
```

Opens Adobe Acrobat Reader:

```
EXECUTE "AcroRd32.exe"
```

Close an application

Closes Microsoft Excel:

```
EXECUTE "TASKKILL /f /im Excel.exe"
```

> **Note**
> Use the `/f` switch with caution. It forces an application to close without presenting any dialog boxes, such as those for saving changes.

Open a file

Opens the Excel workbook `AP_Trans.xlsx`:

```
EXECUTE '"C:\ACL Projects\Source Data\AP_Trans.xlsx"'
```

Create a new folder

Creates a new folder named `Source Data`:

```
EXECUTE 'cmd /c MD "C:\ACL Projects\Source Data"'
```

Run external scripts or non-Analytics batch files (.bat)

Runs the script `My_Batch.bat`:

```
EXECUTE '"C:\ACL Projects\Batch Files\My_Batch.bat"'
```

Pass parameters to a non-Analytics batch file

Passes two parameters to `My_Batch.bat`. Parameters can be literals or Analytics variables:

```
EXECUTE '"C:\ACL Projects\Batch Files\My_Batch.bat" param1%v_param2%'
```

Run Analytics scripts in other Analytics projects

Runs "AP_Trans_script" in `AP Trans Tests.acl`"

```
EXECUTE 'aclwin.exe "C:\ACL Projects\AP Trans Tests.acl" /b AP_Trans_
script'
```

> **Note**
> Running an Analytics script in another project launches a second instance of Analytics. The script in the second project should end with the QUIT command so that the second instance of Analytics closes and control is returned to the initial instance of Analytics.

Incorporate a waiting period in an Analytics script

Both examples create a waiting period of 30 seconds:

```
EXECUTE "TIMEOUT /t 30"
```

```
EXECUTE "cmd /c PING -n 31 127.0.0.1 > nul"
```

Remarks

Use EXECUTE to perform useful tasks

The EXECUTE command allows you to run Windows and DOS commands from the Analytics command line or from an Analytics script.

You can use this ability to increase the automation of Analytics scripts by performing a variety of useful tasks that are not possible using ACLScript syntax alone.

Examples of tasks that can be started using EXECUTE

Open other programs and applications and perform tasks required by the Analytics script	Pass parameters to a batch file	Access data from network locations	Incorporate Active Directory account lists
Open any file in its default application	Run Analytics scripts in other Analytics projects	Use FTP to access data from remote locations	Integrate with VBScript
Perform file and folder administrative tasks such as copying, moving, creating, deleting, or comparing files or folders that exist outside of Analytics	Incorporate waiting periods in Analytics scripts	Zip or unzip data	Integrate with SQL databases
Run external scripts or non-Analytics batch files (.bat)	Incorporate Windows task scheduling in Analytics scripts	Encrypt or decrypt data	Open web pages

> **Note**
> Specific details of how to perform any of these tasks are beyond the scope of Galvanize Help documentation. For assistance, consult appropriate Windows operating system documentation, or other third-party documentation.

Default mode and asynchronous mode

You can run the EXECUTE command in either default mode or asynchronous mode:

- **Default mode** – the process started by EXECUTE must complete before the Analytics script can continue.

  Analytics is inaccessible while external processes are running.
- **Asynchronous mode** – the Analytics script continues to run without waiting for the process started by EXECUTE to complete.

Analytics continues to be accessible while external processes are running.

If you specify ASYNC, the EXECUTE command runs in asynchronous mode.

Which mode should I use?

When you create Analytics scripts that use the EXECUTE command you need to consider which mode of operation is appropriate.

Use default mode	Use asynchronous mode / ASYNC
◦ file and folder administrative tasks ◦ specifying waiting periods ◦ any task that subsequent tasks depend on ◦ subsequent script execution depends on the result in the RETURN_CODE variable	◦ external tasks cause an application interface or pop-up dialog box to open

Analytics scripts that run unattended

If you want an Analytics script that contains the EXECUTE command to run unattended, use one of the following methods:

- use asynchronous mode for any tasks that cause an application interface or pop-up dialog box to open
- avoid opening interface elements in unattended scripts

> **Note**
> Until interface elements are closed, they represent processes that are still running. If these interface elements are opened with EXECUTE in default mode, they prevent subsequent lines in an Analytics script from executing, and cause the script to hang.

EXECUTE command in analytic scripts

If you want to use the EXECUTE command in analytic scripts in Robots or on AX Server, you must specifically configure the command to run. For more information, see:

- **Robots** – Configuring a Robots Agent
- **AX Server** – AX Server settings

Quotation marks

The Windows Run command syntax that you use with the EXECUTE command must be enclosed by either single or double quotation marks.

The following example uses the Windows `MD` command to create a new folder:

```
EXECUTE 'cmd /c md C:\New_Data_Folder'
```

Nested quotation marks

If any paths within the Run command syntax contain spaces, the paths must also be enclosed within quotation marks.

You have two options when enclosing paths within quotation marks:

- **Double quotation marks inside single quotation marks** – Use single quotation marks to enclose the entire Run command string, and use double quotation marks internally to enclose paths:

  ```
  EXECUTE 'cmd /c md "C:\New Data Folder"'
  ```

  You may find this method easier to read than the second method.

  > **Note**
  > Reversing the order of the nesting – using double quotation marks to enclose the entire string, and single quotation marks to enclose paths – does not work.

- **Two double quotation marks** – Use double quotation marks to enclose the entire Run command string, and use two double quotation marks internally to enclose paths:

  ```
  EXECUTE "cmd /c md ""C:\New Data Folder"""
  ```

  If you use this second method, the two double quotation marks used internally must be immediately adjacent and cannot have a space between them.

Internal and external commands

Windows commands can be either **internal** or **external**.

- **Internal commands** – can be run from the command prompt only, which means that you have to open the command shell using `cmd /c` or `cmd /k` before specifying the command.
- **External commands** – can be run from either the command prompt or directly using the EXECUTE command, which means opening the command shell is an option, but not required.

The example below uses the internal Windows `DIR` command (displays the contents of a directory), and the external Windows `COMP` command (compares two files), to illustrate the difference:

```
EXECUTE 'cmd /k dir "C:\ACL DATA\Sample Data Files"'
EXECUTE 'comp C:\File_1.txt C:\File_2.txt'
```

You can avoid this complication by creating external scripts or batch files to contain Windows commands, and by using the EXECUTE command only to start the batch file. For example:

```
EXECUTE 'C:\My_Batch.bat'
```

Multi-line Run command syntax

The EXECUTE command does not support multi-line Run command syntax. To incorporate multi-line Run commands in an Analytics script, use one of the following methods:

Method	Example
Repeat the EXECUTE command for each Run command.	`EXECUTE 'cmd /c md "C:\New Data Folder"'` `EXECUTE 'cmd /c copy C:\File_1.txt "C:\New Data Folder"'`
Combine Run commands using '&'.	`EXECUTE 'cmd /c md "C:\New Data Folder" & copy C:\File_1.txt "C:\New Data Folder"'`
Create an external script or batch file to contain multi-line Run commands, and use the EXECUTE command only to start the batch file.	`EXECUTE 'C:\My_Batch.bat'`

EXPORT command

Exports data from Analytics to the specified file format, or to HighBond Results.

Syntax

```
EXPORT {<FIELDS> field_name <AS export_name> <...n>|<FIELDS> ALL <EXCLUDE
field_name <...n>>} <UNICODE> export_type <SCHEMA> PASSWORD num TO
{filename|aclgrc_id} <OVERWRITE> <IF test> <WHILE test> <{FIRST range|NEXT
range}> <APPEND> <KEEPTITLE> <SEPARATOR character> <QUALIFIER character>
<WORKSHEET worksheet_name> <DISPLAYNAME>
```

Parameters

Name	Description
FIELDS *field_name* AS *export_name* <...*n*> | FIELDS ALL	The fields to export. ◦ **FIELDS *field_name*** - export the specified field or fields Separate field names with spaces. Fields are exported in the order that you list them. You can optionally include a different name for the field in the export file using AS *export_name*. Enclose *export_name* in quotation marks. If you are exporting to HighBond Results (`ACLGRC`), it is possible to combine AS with the DISPLAYNAME parameter. For more information, see "How DISPLAYNAME interacts with AS when exporting to HighBond Results" on page 1711. ◦ **FIELDS ALL** - export all fields in the table Fields are exported in the order that they appear in the table layout.
EXCLUDE *field_name* optional	Only valid when exporting using FIELDS ALL. The field or fields to exclude from the command. EXCLUDE allows you to fine-tune FIELDS ALL, by excluding the specified fields. EXCLUDE must immediately follow FIELDS ALL. For example: `FIELDS ALL EXCLUDE field_1 field_2`
UNICODE	Available in the Unicode edition of Analytics only. Applies to text (ASCII), delimited

Name	Description
optional	text (DELIMITED), and XML files only, and to Windows Clipboard (CLIPBOARD) output. Exports Analytics data with Unicode UTF-16 LE character encoding applied. ◦ **Specify UNICODE** – if the data you are exporting contains characters that are not supported by extended ASCII (ANSI) ◦ **Do not specify UNICODE** – if all the characters in the data you are exporting are supported by extended ASCII (ANSI) The exported data is encoded as extended ASCII (ANSI). **Note** Any unsupported characters are omitted from the exported file. For more information, see "Galvanize Unicode products" on page 2576.
export_type	The output file format or destination using one of the following options: ◦ **ACCESS** – Microsoft Access database file (.mdb) By default, the data is exported as Unicode. ◦ **ACLGRC** – HighBond Results ◦ **ASCII** – ASCII plain text (.txt) ◦ **CLIPBOARD** – Windows Clipboard ◦ **DBASE** – dBASE compatible file (.dbf) ◦ **DELIMITED** – delimited text file (.del), or comma-separated values file (.csv) ◦ **EXCEL** – Microsoft Excel file (.xls) compatible with Excel 1997 to 2003 ◦ **JSON** – JSON file (.json) ◦ **LOTUS** – Lotus 123 file ◦ **WDPF6** – Wordperfect 6 file ◦ **WORD** – MS Word file (.doc) ◦ **WP** – Wordperfect file ◦ **XLS21** – Microsoft Excel version 2.1 file ◦ **XLSX** – Microsoft Excel .xlsx file By default, the data is exported as Unicode. ◦ **XML** – XML file (.xml)
SCHEMA optional	Applies to XML file output only. Include the XML schema in the exported XML file. The XML schema contains metadata that describes the structure of the XML file, including the data type of the fields. You can validate the file against the schema once the file has been exported.
PASSWORD *num*	Applies to HighBond Results (`ACLGRC`) only. The password definition to use. You do not use PASSWORD *num* to prompt for, or specify, an actual password. The password definition refers to a password previously supplied or set using the PASSWORD command, the SET PASSWORD command, or the PASSWORD analytic tag. *num* is the number of the password definition. For example, if two passwords have been previously supplied or set in a script, or when scheduling an analytic,

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>
<code>PASSWORD 2</code> specifies that password #2 is used.
For more information about supplying or setting passwords, see:
◦ "PASSWORD command" on page 1893
◦ "SET command" on page 1960
◦ "PASSWORD tag" on page 2530
PASSWORD num must be placed immediately before TO, or at the end of the string of command syntax.
The required password value is a HighBond access token. For more information, see "Exporting to HighBond Results" on page 1709.
Note
PASSWORD may or may not be required, depending on the environment in which the script runs:
<table>
<tr><td>Analytics
(online activation)</td><td>PASSWORD is not required.
The current user's HighBond access token is automatically used.</td></tr>
<tr><td>Analytics
(offline activation)</td><td rowspan="4">PASSWORD is required.</td></tr>
<tr><td>Robots</td></tr>
<tr><td>Analytics Exchange</td></tr>
<tr><td>Analysis App window</td></tr>
</table>
</td></tr>
<tr><td>TO filename | aclgrc_id</td><td>
The destination for the export:
◦ TO filename - export data to a file
If required, you can include either an absolute or relative file path, but the Windows folder must already exist. You must specify the filename value as a quoted string.
Note
To export to a comma-separated values file (<code>*.csv</code>), you must specify the .csv file extension as part of filename. For example: <code>vendors.csv</code>
◦ TO aclgrc_id - export data to HighBond Results
The aclgrc_id value must include the control test ID number, and if you are exporting to a data center other than North America (US), the data center code. The aclgrc_id value must be enclosed in quotation marks.
The control test ID number and the data center code must be separated by the at sign (@). For example, <code>TO "99@eu"</code>.
If you do not know the control test ID number, use the Analytics user interface to begin an export to Results. Cancel the export once you have identified the
</td></tr>
</table>

Name	Description
	control test ID number. For more information, see "Exporting exceptions to HighBond Results" on page 208. The data center code specifies which regional HighBond server you are exporting the data to: • `af` – Africa (South Africa) • `ap` – Asia Pacific (Singapore) • `au` – Asia Pacific (Australia) • `ca` – North America (Canada) • `eu` – Europe (Germany) • `sa` – South America (Brazil) • `us` – North America (US) You can use only the data center code or codes authorized for your organization's instance of HighBond. The North America (US) data center is the default, so specifying `@us` is optional.
OVERWRITE optional	Applies to HighBond Results (`ACLGRC`) only. ◦ **OVERWRITE specified** – Exported data overwrites any existing data in the target control test (table). You must have a Professional Manager role in the target Collection to overwrite data. ◦ **OVERWRITE omitted** – Exported data is appended to any existing data in the target control test (table). For more information, see "Exporting to HighBond Results" on page 1709. Any interpretations related to the target control test (table) dynamically update to reflect the imported data, whether you overwrite or append.
IF *test* optional	A conditional expression that must be true in order to process each record. The command is executed on only those records that satisfy the condition. **Note** The IF parameter is evaluated against only the records remaining in a table after any scope parameters have been applied (WHILE, FIRST, NEXT).
WHILE *test* optional	A conditional expression that must be true in order to process each record. The command is executed until the condition evaluates as false, or the end of the table is reached. **Note** If you use WHILE in conjunction with FIRST or NEXT, record processing stops as soon as one limit is reached.
FIRST *range* | NEXT *range* optional	The number of records to process: ◦ **FIRST** – start processing from the first record until the specified number of records is reached ◦ **NEXT** – start processing from the currently selected record until the specified number of records is reached Use *range* to specify the number of records to process. If you omit FIRST and NEXT, all records are processed by default.

Name	Description
APPEND optional	Applies to text (ASCII) and delimited text (DELIMITED) files only. Appends the command output to the end of an existing file instead of overwriting it. **Note** You must ensure that the structure of the command output and the existing file are identical: • the same fields • the same field order • matching fields are the same length • matching fields are the same data type Analytics appends output to an existing file regardless of its structure. If the structure of the output and the existing file do not match, jumbled, missing, or inaccurate data can result.
KEEPTITLE optional	Applies only to text files (ASCII), and delimited text and comma-separated values files (DELIMITED). Include the Analytics field names with the exported data. If omitted, no field names appear in the output file.
SEPARATOR *character* optional	Applies only to delimited text and comma-separated values files (DELIMITED). The character to use as the separator between fields. You must specify the character as a quoted string. By default, Analytics uses a comma. Do not specify any character other than a comma if you are exporting to a comma-separated values file.
QUALIFIER *character* optional	Applies only to delimited text and comma-separated values files (DELIMITED). The character to use as the text qualifier to wrap and identify field values. You must specify the character as a quoted string. By default, Analytics uses double quotation marks.
WORKSHEET *worksheet_name* optional	Applies to Microsoft Excel (.xlsx) files only. The name of the Excel worksheet created in a new or existing Excel file. By default, Analytics uses the name of the Analytics table you are exporting as the worksheet name. The *worksheet_name* can contain only alphanumeric characters or the underscore character (_). The name cannot contain special characters, spaces, or start with a number. Enclosing the value in quotation marks is optional. For details about overwriting Excel workbooks and worksheets when exporting, see "The WORKSHEET parameter and overwriting" on page 1708.
DISPLAYNAME optional	Applies to HighBond Results (`ACLGRC`) only. Exports field names as field names and display names as display names so the display names appear in column headings in Results without affecting the actual field name. It is possible to combine DISPLAYNAME with AS. For more information see "How DISPLAYNAME interacts with AS when exporting to HighBond Results" on

Name	Description
	page 1711.

Examples

Export data to an Excel .xlsx file

You export specific fields from the **Vendor** table to an Excel .xlsx file:

```
OPEN Vendor
EXPORT FIELDS Vendor_No Vendor_Name Vendor_City XLSX TO "VendorExport"
```

Export data to an Excel .xlsx file and specify a worksheet name

You export specific fields from the **Vendor** table to a worksheet called **Vendors_US** in an Excel .xlsx file:

```
OPEN Vendor
EXPORT FIELDS Vendor_No Vendor_Name Vendor_City XLSX TO "VendorExport"
WORKSHEET Vendors_US
```

Export all fields to a delimited file

You export all fields from the **Vendor** table to a delimited file:

```
OPEN Vendor
EXPORT FIELDS ALL DELIMITED TO "VendorExport"
```

Export a subset of fields to a delimited file

You have two options when exporting a subset of the fields in a table to an external file:

- specify the individual fields to export
- specify FIELDS ALL and specify the fields to exclude from the export

> **Tip**
> Use whichever method is the least labor-intensive.

The examples below refer to the **Vendor** table, which has eight fields:

- vendor number
- vendor name
- vendor street
- vendor city
- vendor state
- vendor ZIP
- last active date
- review date

Specify the fields to export

You export two fields from the **Vendor** table to a delimited file:

```
OPEN Vendor
EXPORT FIELDS Vendor_No Vendor_Name DELIMITED TO "Vendors" KEEPTITLE
SEPARATOR "|" QUALIFIER '"'
```

Specify FIELDS ALL and specify the fields to exclude

You export all fields, except the last active date and review date fields, from the **Vendor** table to a delimited file:

```
OPEN Vendor
EXPORT FIELDS ALL EXCLUDE Vendor_Last_Active Vendor_Review_Date
DELIMITED TO "Vendor_addresses" KEEPTITLE SEPARATOR "|" QUALIFIER '"'
```

Export all fields to a comma-separated values file

You export all fields from the **Vendor** table to a comma-separated values file:

```
OPEN Vendor
EXPORT FIELDS ALL DELIMITED TO "VendorExport.csv"
```

Export data to multiple delimited files using GROUP

You export specific fields from the **Vendor** table to two delimited files:

- one file for vendor names from "A" to "M"
- one file for vendor names from "N" to "Z"

Using the GROUP command, you test the vendor name of each record with an IF condition:

```
GROUP
  EXPORT FIELDS Vendor_No Vendor_Name DELIMITED TO "AtoM" IF BETWEEN
(UPPER(VENDOR_NAME), "A", "M")
  EXPORT FIELDS Vendor_No Vendor_Name DELIMITED TO "NtoZ" IF BETWEEN
(UPPER(VENDOR_NAME), "N", "Z")
END
```

Export data to HighBond Results

You export specific fields from the **AR_Exceptions** table to HighBond Results. You overwrite existing data in the target control test (table):

```
OPEN AR_Exceptions
EXPORT FIELDS No Due Date Ref Amount Type ACLGRC PASSWORD 1 TO
"10926@us" OVERWRITE
```

Remarks

For more information about how this command works, see "Exporting data" on page 203.

Using EXPORT with the GROUP command

For most export formats, you can export data into multiple files simultaneously using the GROUP command.

Only one file can be created at a time when you are exporting data to Microsoft Excel and Microsoft Access.

Exporting to Excel

The following limits apply when exporting data to an Excel file:

Number of records	◦ **Excel 2007 and later (*.xlsx)** - a maximum of 1,048,576 records ◦ **Excel 97 and 2003** - a maximum of 65,536 records Analytics tables that exceed these maximums export successfully, but the excess records are ignored and not exported.
Length of fields	◦ no specific field length limit ◦ combined field lengths cannot exceed the overall record length limit of 32 KB (32,765 characters in non-Unicode Analytics, 16,382 characters in Unicode Analytics) ◦ for Excel 2.1, a maximum of 247 characters
Length of field names	◦ a maximum of 64 characters ◦ for Excel 2.1, a maximum of 248 characters

The WORKSHEET parameter and overwriting

The result of using or not using the WORKSHEET parameter when exporting from an Analytics table to an Excel file is explained below:

Matching	Description	WORKSHEET parameter used	WORKSHEET parameter not used
No matching Excel file name	◦ TO *filename* value does not match any existing Excel file name	A new Excel file is created, with a worksheet with the specified name	A new Excel file is created, with a worksheet that uses the name of the exported Analytics table
Matching Excel file name No matching worksheet name	◦ TO *filename* value, and an existing Excel file name, are identical ◦ WORKSHEET *worksheet_name* does not match a worksheet name in the Excel file	A worksheet with the specified name is added to the existing Excel file	The existing Excel file is overwritten by a new Excel file, with a worksheet that uses the name of the exported Analytics table

Matching	Description	WORKSHEET parameter used	WORKSHEET parameter not used
Matching Excel file name and worksheet name	◦ TO *filename* value, and an existing Excel file name, are identical ◦ WORKSHEET *worksheet_name* matches a worksheet name in the Excel file	A worksheet with the specified name overwrites the existing worksheet if it was originally created from Analytics. An error message appears and the export operation is canceled if the existing worksheet was originally created directly in Excel.	The existing Excel file is overwritten by a new Excel file, with a worksheet that uses the name of the exported Analytics table

Exporting to HighBond Results

The table below contains additional information about exporting to a control test in Results.

Item	Details
Required permissions	The ability to export results to a control test in Results requires a specific HighBond role assignment, or administrative privileges: ◦ Users with a Professional User or Professional Manager role for a Results collection can export results to any control test in the collection. **Note** Only users with the Professional Manager role can export and overwrite existing data in a control test. ◦ HighBond System Admins and Results admins automatically get a Professional Manager role in all collections in the HighBond organization or organizations they administer.
Export limits	The following limits apply when exporting to a control test: ◦ A maximum of 100,000 records per export ◦ A maximum of 100,000 records per control test ◦ A maximum of 500 fields per record ◦ A maximum of 256 characters per field You can export multiple times to the same control test, but you cannot exceed the overall limits.
Appending fields (OVERWRITE not specified)	Regardless of their order in an Analytics table, exported fields are appended to existing fields in a control test if they have matching physical field names. In Analytics, the physical field name is the name in the table layout. Exported fields that do not match the name of any existing field are added as additional columns to the table in Results. Display names of fields in Analytics, and in Results, are not considered. However, if you

Item	Details
	use the optional `AS export_name` parameter, the `export_name` value is used as the physical field name if you do not use `DISPLAYNAME`. When appending data to questionnaire fields, the display name of the column in Results remains the name that is specified in the questionnaire configuration. Appending works differently if the target control test has a primary key field specified. For more information, see "Exporting exceptions to HighBond Results" on page 208. **Note** If you are round-tripping data between Results and Analytics, and data ends up misaligned in Results, you probably have mismatched field names. For more information, see "Field name considerations when importing and exporting Results data" on page 1789.
Creating a password definition and specifying a password value	PASSWORD command If you use the PASSWORD command to create the numbered password definition for connecting to HighBond, no password value is specified, so a password prompt is displayed when the script attempts to connect. For more information, see "PASSWORD command" on page 1893. SET PASSWORD command If you use the SET PASSWORD command to create the numbered password definition for connecting to HighBond, a password value is specified, so no password prompt is displayed, which is appropriate for scripts designed to run unattended. For more information, see SET PASSWORD command. Acquire a HighBond access token Regardless of which method you use to create the password definition, the required password value is a HighBond access token, which users can generate in Launchpad. **Caution** The generated access token matches the account used to sign in to Launchpad. As a scriptwriter, specifying your own access token in a script may not be appropriate if the script will be used by other people. a. Do one of the following: • From the Analytics main menu, select **Tools > HighBond Access Token**. • In the **Script Editor**, right-click and select **Insert > HighBond Token**. The **Manage API tokens** page opens in your browser. You may be required to first sign in to Launchpad. b. Do one of the following: • **Use an existing token** - In the **Token** column, click the partially masked token that you want to use and enter your HighBond account password. The unmasked token is displayed.

Item	Details
	Tip Use an existing token unless you have a reason for creating a new one. If the existing token does not work, create a new one. Using an existing token cuts down on the number of tokens you need to manage. • **Create a new token** – Click **Create token > Analytics** and enter your HighBond account password. A new Analytics token is created. **Note** If you are a Launchpad System Admin, you also have the option of creating an API token. You should reserve API tokens for their intended purpose, which is programmatic access to the HighBond platform. c. Click **Copy** to copy the token. **Tip** Do not close the dialog box containing the token until you have successfully pasted the token. d. In Analytics, do one of the following: • paste the token into the password prompt • paste the token at the appropriate point in the SET PASSWORD command syntax in a script e. In Launchpad, close the dialog box containing the token. If you created a new token, a partially masked version of the token is added to the top of your list of tokens. For more information, see Creating and managing access tokens.

How DISPLAYNAME interacts with AS when exporting to HighBond Results

The matrix below shows how the DISPLAYNAME parameter interacts with AS when exporting field names from Analytics to Results.

	Without AS	With AS
Without DISPLAYNAME	Field name and display name in Results are the field name from Analytics.	Field name and display name in Results are the display name in the AS parameter.
With DISPLAYNAME	Field name in Results is the field name from Analytics. Display name in Results is the display name from Analytics.	Field name in Results is the field name from Analytics. Display name in Results is the display name in the AS parameter.

EXTRACT command

Extracts data from an Analytics table and outputs it to a new Analytics table, or appends it to an existing Analytics table. You can extract entire records or selected fields.

Syntax

```
EXTRACT {RECORD|FIELDS field_name <AS display_name> <...n>|FIELDS ALL <EXCLUDE
field_name <...n>>}  TO table_name <LOCAL> <IF test> <WHILE test> <FIRST
range|NEXT range> <EOF> <APPEND> <OPEN>
```

Parameters

Name	Description
RECORD | FIELDS *field_name* | FIELDS ALL	The fields to include in the output: ◦ **RECORD** – use the entire record in the source data file: all fields in the table, and any undefined portions of the record Fields are used in the order that they appear in the table layout. Preserves computed fields. ◦ **FIELDS *field_name*** – use the specified fields Fields are used in the order that you list them. Converts computed fields to physical fields of the appropriate data type in the destination table – ASCII or Unicode (depending on the edition of Analytics), ACL (the native numeric data type), Datetime, or Logical. Populates the physical fields with the actual computed values. ◦ **FIELDS ALL** – use all fields in the table Fields are used in the order that they appear in the table layout. Converts computed fields to physical fields of the appropriate data type in the destination table – ASCII or Unicode (depending on the edition of Analytics), ACL (the native numeric data type), Datetime, or Logical. Populates the physical fields with the actual computed values.
AS *display_name* optional	Only valid when extracting using FIELDS *field_name*. The display name (alternate column title) for the field in the view in the new Analytics table. If you want the display name to be the same as the field name, or an existing display name in the source table, do not use AS.

Name	Description
	Specify *display_name* as a quoted string. Use a semi-colon (;) between words if you want a line break in the column title. **Note** AS works only when extracting to a new table. If you are appending to an existing table, the alternate column titles in the existing table take precedence.
EXCLUDE *field_name* optional	Only valid when extracting using FIELDS ALL. The field or fields to exclude from the command. EXCLUDE allows you to fine-tune FIELDS ALL, by excluding the specified fields. EXCLUDE must immediately follow FIELDS ALL. For example: `FIELDS ALL EXCLUDE field_1 field_2`
TO *table_name*	The location to send the results of the command to: ◦ ***table_name*** - saves the results to an Analytics table Specify *table_name* as a quoted string with a .FIL file extension. For example: `TO "Output.FIL"` By default, the table data file (.FIL) is saved to the folder containing the Analytics project. Use either an absolute or relative file path to save the data file to a different, existing folder: • `TO "C:\Output.FIL"` • `TO "Results\Output.FIL"` **Note** Table names are limited to 64 alphanumeric characters, not including the .FIL extension. The name can include the underscore character (_), but no other special characters, or any spaces. The name cannot start with a number.
LOCAL optional	Saves the output file in the same location as the Analytics project. **Note** Applicable only when running the command against a server table with an output file that is an Analytics table. The LOCAL parameter must immediately follow the TO parameter.
IF *test* optional	A conditional expression that must be true in order to process each record. The command is executed on only those records that satisfy the condition. **Note** The IF parameter is evaluated against only the records remaining in a table after any scope parameters have been applied (WHILE, FIRST, NEXT).

Name	Description
WHILE *test* optional	A conditional expression that must be true in order to process each record. The command is executed until the condition evaluates as false, or the end of the table is reached. **Note** If you use WHILE in conjunction with FIRST or NEXT, record processing stops as soon as one limit is reached.
FIRST *range* | NEXT *range* optional	The number of records to process: ◦ **FIRST** - start processing from the first record until the specified number of records is reached ◦ **NEXT** - start processing from the currently selected record until the specified number of records is reached Use *range* to specify the number of records to process. If you omit FIRST and NEXT, all records are processed by default.
EOF optional	Execute the command one more time after the end of the file has been reached. This ensures that the final record in the table is processed when inside a GROUP command. Only use EOF if all fields are computed fields referring to earlier records.
APPEND optional	Appends the command output to the end of an existing file instead of overwriting it. **Note** You must ensure that the structure of the command output and the existing file are identical: • the same fields • the same field order • matching fields are the same length • matching fields are the same data type Analytics appends output to an existing file regardless of its structure. If the structure of the output and the existing file do not match, jumbled, missing, or inaccurate data can result.
OPEN optional	Opens the table created by the command after the command executes. Only valid if the command creates an output table.

Examples

Extracting all records in a table to a new table

You create an exact duplicate of the **AR_Customer** table by extracting all the records to a new Analytics table. Any computed fields are preserved as computed fields:

```
OPEN AR_Customer
EXTRACT RECORD TO "AR_Customer_2"
```

Extracting all fields in a table to a new table

You extract all defined fields in the **AR_Customer** table to a new Analytics table. Any computed fields are converted to physical fields and populated with the actual computed values:

```
OPEN AR_Customer
EXTRACT FIELDS ALL TO "AR_Customer_2"
```

Extracting all records in a table and appending them to an existing table

You extract all the records in the **AR_Customer** table and append them as a group to the end of the **AR_Customer_Master** table:

```
OPEN AR_Customer
EXTRACT RECORD TO "AR_Customer_Master" APPEND
```

Extracting all records in a table and appending them to an existing table in a different folder

You extract all the records in the **AR_Customer** table and append them as a group to the end of the **AR_Customer_Master** table, which is in a folder other than the Analytics project folder:

```
OPEN AR_Customer
EXTRACT RECORD TO "C:\Users\Customer Data\AR_Customer_Master" APPEND
```

Extracting a subset of fields from a table to a new table

You have two options when extracting a subset of the fields in a table:

- specify the individual fields to extract
- specify FIELDS ALL and specify the fields to exclude from extraction

> **Tip**
> Use whichever method is the least labor-intensive.

The examples below refer to the **AR_Customer** table, which has seven fields:

- reference number
- customer number
- customer name
- transaction type
- invoice date
- due date
- invoice amount

Specify the fields to extract

You extract three fields from the **AR_Customer** table to a new Analytics table:

```
OPEN AR_Customer
EXTRACT FIELDS Name Due Date TO "AR_Customer_Dates.fil"
```

Specify FIELDS ALL and specify the fields to exclude

You extract all fields, except the Reference_num field, from the **AR_Customer** table to a new Analytics table:

```
OPEN AR_Customer
EXTRACT FIELDS ALL EXCLUDE Reference_num TO "AR_Customer_Dates.fil"
```

Creating display names for extracted fields

You extract three fields from the **AR_Customer** table and create display names for the fields in the new Analytics table:

```
OPEN AR_Customer
EXTRACT FIELDS Name AS "Customer;Name" Due AS "Due;Date" Date AS
"Invoice;Date" TO "AR_Customer_Dates.fil"
```

Extracting fields based on a condition

You extract three fields from the **AR_Customer** table to a new Analytics table if the date in the **Due** field is before July 1, 2014:

```
OPEN AR_Customer
EXTRACT FIELDS Name Due Date IF Due < `20140701` TO "Overdue.fil"
```

Remarks

For more information about how this command works, see "Extracting data" on page 194 or "Extracting and appending data" on page 870.

EXTRACT vs Copying a table

EXTRACT creates a new source data file (.fil) as well as a new table layout.

Copying a table using the **Navigator** (**Edit > Copy**) creates a new table layout that remains associated with the original source data file. It does not create a new data file.

FIELDSHIFT command

Shifts the start position of a field definition in a table layout.

Syntax

```
FIELDSHIFT START starting_position COLUMNS bytes_to_shift <FILTER data_filter_name> <OK>
```

Parameters

Name	Description
START *starting_position*	The starting byte position of the first field definition you want to shift. All field definitions to the right of the specified field definition are also shifted. If you specify a non-starting byte position, the next starting byte position is used. **Note** non-Unicode Analytics: 1 byte = 1 character Unicode Analytics, extended ASCII (ANSI) data: 1 byte = 1 character Unicode Analytics, Unicode data: 2 bytes = 1 character For Unicode data, typically you should specify an odd-numbered starting byte position. Specifying an even-numbered starting position can cause characters to display incorrectly.
COLUMNS *bytes_to_shift*	The number of bytes to shift the field definition. Enter a positive number to shift a field definition to the right. Enter a negative number to shift a field definition to the left.

Name	Description
	Note non-Unicode Analytics: 1 byte = 1 character Unicode Analytics, extended ASCII (ANSI) data: 1 byte = 1 character Unicode Analytics, Unicode data: 2 bytes = 1 character For Unicode data, specify an even number of bytes only. Specifying an odd number of bytes can cause characters to display incorrectly.
FILTER *data_filter_name* optional	The name of the filter that identifies field definitions associated with a particular record definition.
OK optional	Deletes or overwrites items without asking you to confirm the action.

Examples

Shifting field definitions

You shift the field definition starting at byte 11, and any subsequent field definitions, 4 bytes to the right:

```
FIELDSHIFT START 11 COLUMNS 4
```

Remarks

For more information about how this command works, see "Shifting fields in table layouts" on page 753.

Shifted field definitions must remain within the record length

When you shift one or more field definitions right or left, the fields cannot exceed the record length in either direction.

Keep in mind that FIELDSHIFT moves both the specified field definition, and any field definitions to the right of the specified definition. If the shifted block of definitions would exceed the record length in either direction, an error message appears and the command is not executed.

> **Tip**
> If the error message is appearing because you are exceeding the end of the record, try removing the final field definition to make room for the field definitions being shifted.

FIND command

Searches an indexed character field for the first value that matches the specified character string.

> **Note**
> The FIND command and the FIND( ) function are two separate Analytics features with significant differences. For information about the function, see "FIND( ) function" on page 2145.

Syntax

```
FIND search_value
```

Parameters

Name	Description
search_value	The character string to search for. *search_value* is case-sensitive and cannot include leading spaces. Do not enclose the value in quotation marks unless quotation marks are part of the data being searched.

Examples

Searching for a specific value

You want to locate the first value in the **Card_Number** character field that exactly matches or starts with "8590124".

First you index the **Card_Number** field in ascending order. Then you run FIND:

```
INDEX ON Card_Number TO "CardNum" OPEN
SET INDEX TO "CardNum"
FIND 8590124
```

Remarks

For more information about how this command works, see "Selecting the first matching record" on page 1167.

When to use FIND

Use the FIND command to move directly to the first record in a table containing the specified *search_value* in the indexed character field.

INDEX requirement

To use the command, the table you are searching must be indexed on a character field in ascending order.

If multiple character fields are indexed in ascending order, only the first field specified in the index is searched. The command cannot be used to search non-character index fields, or character fields indexed in descending order.

Partial matching

Partial matching is supported. The search value can be contained by a longer value in the indexed field. However, the search value must appear at the start of the field to constitute a match.

FIND output depending on match

The FIND command produces one of the following results, depending on whether the search value is found:

- **search value is found** – the first matching record in the table is selected
- **search value is not found** – the table is positioned at the first record with a greater value than the search value

  If there are no values in the indexed field greater than the search value, the table is positioned at the first record. In both cases, the message "No index matched key" is displayed.

The FIND command is not affected by the **Exact Character Comparisons** option (SET EXACT ON/OFF).

FUZZYDUP command

Detects nearly identical values (fuzzy duplicates) in a character field.

> **Note**
> To use fuzzy matching to combine fields from two Analytics tables into a new, single Analytics table, see "FUZZYJOIN command" on page 1729.

Syntax

```
FUZZYDUP ON key_field <OTHER field <...n>|OTHER ALL <EXCLUDE field_name
<...n>>> LEVDISTANCE value <DIFFPCT percentage> <RESULTSIZE percentage>
<EXACT> <IF test> TO table_name <LOCAL> <OPEN>
```

Parameters

Name	Description
ON *key_field*	The character field or expression to test for fuzzy duplicates.
OTHER *field <...n>* | OTHER ALL optional	One or more additional fields to include in the output. ◦ **OTHER *field <...n>*** - include the specified field or fields Fields are included in the order that you list them. ◦ **OTHER ALL** - include all fields in the table that are not specified as the key field. Fields are included in the order that they appear in the table layout.
EXCLUDE *field_name* optional	Only valid when using OTHER ALL. The field or fields to exclude from the command. EXCLUDE allows you to fine-tune OTHER ALL, by excluding the specified fields. EXCLUDE must immediately follow OTHER ALL. For example: `OTHER ALL EXCLUDE field_1 field_2`
LEVDISTANCE *value*	The maximum allowable Levenshtein distance between two strings for them to be identified as fuzzy duplicates and included in the results. The LEVDISTANCE value cannot be less than 1 or greater than 10. Increasing the

Name	Description
	LEVDISTANCE value increases the number of results by including values with a greater degree of fuzziness - that is, values that are more different from one another. For more information, see "FUZZYDUP behavior" on page 1726.
DIFFPCT *percentage* optional	A threshold that limits the 'difference percentage' or the proportion of a string that can be different. The percentage that results from an internal Analytics calculation performed on potential fuzzy duplicate pairs must be less than or equal to the DIFFPCT value for the pair to be included in the results. The DIFFPCT value cannot be less than 1 or greater than 99. If DIFFPCT is omitted the threshold is turned off and difference percentage is not considered during processing of the FUZZYDUP command. For more information, see "FUZZYDUP behavior" on page 1726.
RESULTSIZE *percentage* optional	The maximum size of the set of output results as a percentage of the number of records in the key field. For example, for a key field with 50,000 records, a RESULTSIZE of 3 would terminate processing if the results exceeded 1500 fuzzy duplicates (50,000 x 0.03). No output table is produced if processing is terminated. The RESULTSIZE value cannot be less than 1 or greater than 1000 (one thousand) percent. The limit of 1000% is to accommodate the nature of many-to-many matching, which can produce results that are more numerous than the original test data set. If RESULTSIZE is omitted the threshold is turned off and result size is not considered during processing of the FUZZYDUP command. **Caution** Omitting RESULTSIZE can produce an unduly large set of results that takes a very long time to process, or can cause available memory to be exceeded, which terminates processing. Omit RESULTSIZE only if you are confident that the results will be of a manageable size.
EXACT optional	Includes exact duplicates as well as fuzzy duplicates in the results.
IF *test* optional	A conditional expression that must be true in order to process each record. The command is executed on only those records that satisfy the condition. **Note** The IF parameter is evaluated against only the records remaining in a table after any scope parameters have been applied (WHILE, FIRST, NEXT).
TO *table_name*	The location to send the results of the command to: ◦ ***table_name*** - saves the results to an Analytics table Specify *table_name* as a quoted string with a .FIL file extension. For example: `TO`

Name	Description
	`"Output.FIL"` By default, the table data file (.FIL) is saved to the folder containing the Analytics project. Use either an absolute or relative file path to save the data file to a different, existing folder: • `TO "C:\Output.FIL"` • `TO "Results\Output.FIL"` **Note** Table names are limited to 64 alphanumeric characters, not including the .FIL extension. The name can include the underscore character (_), but no other special characters, or any spaces. The name cannot start with a number.
LOCAL optional	Saves the output file in the same location as the Analytics project. **Note** Applicable only when running the command against a server table with an output file that is an Analytics table. The LOCAL parameter must immediately follow the TO parameter.
OPEN optional	Opens the table created by the command after the command executes. Only valid if the command creates an output table.

Analytics output variables

Name	Contains
GAPDUP*n*	The total number of gaps, duplicates, or fuzzy duplicate groups identified by the command.

Examples

Test a surname field for fuzzy duplicates

You test a surname field for fuzzy duplicates (the **Last_Name** field in the **Employee_List** table in `ACL DATA\Sample Data Files\Metaphor_Employee_Data.ACL`). The results are output to a new Analytics table.

- In addition to the test field, other fields are included in the results.
- The maximum allowable Levenshtein distance is 1.
- The proportion of a string that can be different is limited to 50%.
- The size of the results is limited to 20% of the test field size.
- In addition to fuzzy duplicates, exact duplicates are included.

```
OPEN Employee_List
FUZZYDUP ON Last_Name OTHER First_Name EmpNo LEVDISTANCE 1 DIFFPCT 50
RESULTSIZE 20 EXACT TO "Fuzzy_Last_Name" OPEN
```

Remarks

How it works

The FUZZYDUP command finds nearly identical values (fuzzy duplicates), or locates inconsistent spelling in manually entered data.

Unlike the ISFUZZYDUP( ) function, which identifies an exhaustive list of fuzzy duplicates for a single character value, the FUZZYDUP command identifies all fuzzy duplicates in a field, organizes them in non-exhaustive groups, and outputs results.

For detailed information about how this command works, see "Fuzzy duplicates analysis" on page 1215.

What non-exhaustive means

Non-exhaustive means that individual fuzzy duplicate groups in the results may not contain all the fuzzy duplicates in a test field that are within the specified degree of difference of the group owner. However, if a group owner is a fuzzy duplicate of another value in the test field, the two values will appear together in a group somewhere in the results. So groups may be non-exhaustive, but the results, in total, are exhaustive.

If producing a single, exhaustive list of fuzzy duplicates for a specific value in the test field is important to your analysis, you can use the ISFUZZYDUP( ) function for this purpose.

FUZZYDUP behavior

The FUZZYDUP command has two parameters that allow you to control the degree of difference between fuzzy duplicates, and the size of the results:

- LEVDISTANCE
- DIFFPCT

You may need to try different combinations of settings for these two parameters to find out what works best for a particular data set.

LEVDISTANCE (Levenshtein distance)

When processing data, the FUZZYDUP command calculates the Levenshtein distance between each evaluated pair of strings in the test field, and calculates the difference percentage. The Levenshtein distance is a value representing the minimum number of single character edits required to make one string identical to the other string. For more information, see "LEVDIST( ) function" on page 2214.

DIFFPCT (Difference percentage)

The difference percentage is the percentage of the shorter of the two evaluated strings that is different, and is the result of the following internal Analytics calculation, which uses the Levenshtein distance between the two strings:

Levenshtein distance / number of characters in the shorter string × 100 = difference percentage

More information

For detailed information about the fuzzy duplicate difference settings, controlling result size, and fuzzy duplicate groups, see "Fuzzy duplicates analysis" on page 1215.

Case-sensitivity

The FUZZYDUP command is not case-sensitive, so "SMITH" is equivalent to "smith."

Trailing blanks automatically trimmed

The FUZZYDUP command automatically trims trailing blanks in *key_field*, so there is no need to use the TRIM( ) or ALLTRIM( ) function when specifying a single field for *key_field*.

If you concatenate fields for *key_field*, you should use ALLTRIM( ), as shown below.

Improving the effectiveness of FUZZYDUP

Three techniques can significantly improve the effectiveness of the FUZZYDUP command:

- sorting individual elements in test field values
- removing generic elements from test field values
- concatenating test fields

These techniques generate more focused sets of results with fewer false positives and more true positives. You can use the techniques separately, or in combination.

Sorting individual elements in test field values

The SORTWORDS( ) function can improve the effectiveness of the FUZZYDUP command by sorting individual elements in test field values into a sequential order.

Sorting elements, such as the components of an address, can make two strings with the same information, but a different format, more closely resemble each other. A closer resemblance improves the chances that a pair of strings are selected as fuzzy duplicates of each other.

For more information, see "SORTWORDS( ) function" on page 2378.

For a video providing an overview of SORTWORDS( ), see Fuzzy Matching Using SORTWORDS() (English only).

Removing generic elements from test field values

The OMIT( ) function can improve the effectiveness of the FUZZYDUP command by removing generic elements such as "Corporation" or "Inc.", or characters such as commas, periods, and ampersands (&), from test field values.

Removal of generic elements and punctuation focuses the FUZZYDUP string comparison on just the portion of the strings where a meaningful difference may occur.

For more information, see "OMIT( ) function" on page 2268.

Concatenating test fields

Concatenating two or more test fields can improve the effectiveness of the FUZZYDUP command by increase the degree of uniqueness of the test values.

For example, by concatenating an Address field and a City field you avoid fuzzy matches between addresses in different cities:

```
FUZZYDUP ON ALLTRIM(Address)+ALLTRIM(City) OTHER Address City Vendor_Name
LEVDISTANCE 4 DIFFPCT 50 RESULTSIZE 20 EXACT TO "Vendor_Name_Fuzzy_Dupes" OPEN
```

Other string comparison methods

- **DICECOEFFICIENT( ) function** – provides a method for comparing strings that de-emphasizes or completely ignores the relative position of characters or character blocks.
- **SOUNDSLIKE( ) and SOUNDEX( ) functions** – provide a method for comparing strings based on a phonetic comparison (sound) rather than on an orthographic comparison (spelling).

FUZZYJOIN command

Uses fuzzy matching to combine fields from two Analytics tables into a new, single Analytics table.

> **Note**
>
> To detect nearly identical values in a single character field (fuzzy duplicates), see "FUZZYDUP command" on page 1723.
>
> To join tables using exactly matching key field values, see "JOIN command" on page 1853.

Syntax

```
FUZZYJOIN {DICE PERCENT percentage NGRAM n-gram_length|LEVDISTANCE DISTANCE
value} PKEY primary_key_field SKEY secondary_key_field {FIELDS primary_
fields|FIELDS ALL <EXCLUDE primary_fields <...n>>} <WITH secondary_fields|WITH
ALL <EXCLUDE secondary_fields <...n>>> <IF test> <OPEN> TO table_name
<FIRSTMATCH> <WHILE test> <FIRST range|NEXT range> <APPEND>
```

> **Note**
>
> You cannot run the FUZZYJOIN command locally against a server table.
>
> You must specify the FUZZYJOIN command name in full. You cannot abbreviate it.

Parameters

Name	Description
DICE PERCENT *percentage* NGRAM *n-gram_length* | LEVDISTANCE DISTANCE *value*	The fuzzy matching algorithm to use. **DICE** - use the Dice coefficient algorithm ◦ **PERCENT *percentage*** - the minimum allowable Dice's coefficient of two strings for them to qualify as a fuzzy match Specify a decimal fraction, from 0.0000 to 1.0000 (for example, 0.7500). Use a maximum of four decimal places. Decreasing the value increases the number of matches by including matches with a greater degree of fuzziness - that is, strings that are more different from each other. ◦ **NGRAM *n-gram_length*** - the *n*-gram length to use

Name	Description
	Specify a whole number, 1 or greater. Increasing the *n*-gram length makes the criterion for similarity between two strings stricter. *N*-grams are overlapping substrings (character blocks) into which the comparison strings are divided as part of the Dice's coefficient calculation. **Note** When you specify DICE, the FUZZYJOIN command uses the DICECOEFFICIENT( ) function in an IF statement to conditionally join key field values. For detailed information about the function, see "DICECOEFFICIENT( ) function" on page 2116. **LEVDISTANCE** – use the Levenshtein distance algorithm ◦ **DISTANCE *value*** – the maximum allowable Levenshtein distance between two strings for them to qualify as a fuzzy match Specify a whole number, 1 or greater. Increasing the value increases the number of matches by including matches with a greater degree of fuzziness – that is, strings that are more different from each other. **Note** When you specify LEVDISTANCE, the FUZZYJOIN command uses the LEVDIST( ) function in an IF statement to conditionally join key field values. For detailed information about the function, see "LEVDIST( ) function" on page 2214. Unlike the function, the Levenshtein distance algorithm in the FUZZYJOIN command automatically trims leading and trailing blanks, and is not case-sensitive.
PKEY *primary_key_field*	The character key field, or expression, in the primary table. You can specify only one primary key field.
SKEY *secondary_key_field*	The character key field, or expression, in the secondary table. You can specify only one secondary key field.
FIELDS *primary_fields* | FIELDS ALL	The fields or expressions from the primary table to include in the joined output table. ◦ **FIELDS *primary_fields*** – include the specified field or fields Fields are included in the order that you list them. ◦ **FIELDS ALL** – include all fields from the table Fields are included in the order that they appear in the table layout. **Note** You must explicitly specify the primary key field if you want to include it in the joined table. Specifying FIELDS ALL also includes it.
EXCLUDE *primary_*	Only valid when performing a fuzzy join using FIELDS ALL.

Name	Description
fields optional	The field or fields to exclude from the command. EXCLUDE allows you to fine-tune FIELDS ALL, by excluding the specified fields. EXCLUDE must immediately follow FIELDS ALL. For example: `FIELDS ALL EXCLUDE field_1 field_2`
WITH *secondary_fields* | WITH ALL optional	The fields or expressions from the secondary table to include in the joined output table. ◦ **WITH *secondary_fields*** - include the specified field or fields Fields are included in the order that you list them. ◦ **WITH ALL** - include all fields from the table Fields are included in the order that they appear in the table layout. **Note** You must explicitly specify the secondary key field if you want to include it in the joined table. Specifying WITH ALL also includes it.
EXCLUDE *secondary_fields* optional	Only valid when performing a fuzzy join using WITH ALL. The field or fields to exclude from the command. EXCLUDE allows you to fine-tune WITH ALL, by excluding the specified fields. EXCLUDE must immediately follow WITH ALL. For example: `WITH ALL EXCLUDE field_1 field_2`
IF *test* optional	A conditional expression that must be true in order to process each record. The command is executed on only those records that satisfy the condition. **Note** The IF parameter is evaluated against only the records remaining in a table after any scope parameters have been applied (WHILE, FIRST, NEXT). **Note** The IF condition can reference the primary table, the secondary table, or both.
OPEN optional	Opens the table created by the command after the command executes. Only valid if the command creates an output table.
TO *table_name*	The location to send the results of the command to: ◦ ***table_name*** - saves the results to an Analytics table Specify *table_name* as a quoted string with a .FIL file extension. For example: `TO "Output.FIL"`

Name	Description
	By default, the table data file (.FIL) is saved to the folder containing the Analytics project. Use either an absolute or relative file path to save the data file to a different, existing folder: • `TO "C:\Output.FIL"` • `TO "Results\Output.FIL"` **Note** Table names are limited to 64 alphanumeric characters, not including the .FIL extension. The name can include the underscore character (_), but no other special characters, or any spaces. The name cannot start with a number.
FIRSTMATCH optional	Specifies that each primary key value is joined to only the first occurrence of any secondary key matches. If the first occurrence happens to be an exact match, any subsequent fuzzy matches for the primary key value are not included in the joined output table. If you omit FIRSTMATCH, the default behavior of FUZZYJOIN is to join each primary key value to all occurrences of any secondary key matches. FIRSTMATCH is useful if you only want to know if any matches, exact or fuzzy, exist between two tables, and you want to avoid the processing time required to identify all matches. You can also use FIRSTMATCH if you are certain that at most only one match exists in the secondary table for each primary key value.
WHILE *test* optional	A conditional expression that must be true in order to process each record. The command is executed until the condition evaluates as false, or the end of the table is reached. **Note** If you use WHILE in conjunction with FIRST or NEXT, record processing stops as soon as one limit is reached.
FIRST *range* | NEXT *range* optional	The number of records to process: ◦ **FIRST** - start processing from the first record until the specified number of records is reached ◦ **NEXT** - start processing from the currently selected record until the specified number of records is reached Use *range* to specify the number of records to process. If you omit FIRST and NEXT, all records are processed by default.
APPEND optional	Appends the command output to the end of an existing file instead of overwriting it.

Name	Description
	Note You must ensure that the structure of the command output and the existing file are identical: • the same fields • the same field order • matching fields are the same length • matching fields are the same data type Analytics appends output to an existing file regardless of its structure. If the structure of the output and the existing file do not match, jumbled, missing, or inaccurate data can result.
ISOLOCALE *locale_code* optional	**Note** Applicable in the Unicode edition of Analytics only. The system locale in the format *language_country*. For example, to use Canadian French, enter `fr_ca`. Use the following codes: ◦ **language** - ISO 639 standard language code ◦ **country** - ISO 3166 standard country code If you do not specify a country code, the default country for the language is used. If you do not use ISOLOCALE, the default system locale is used.

Examples

Use fuzzy matching to join two tables as a way of discovering employees who may also be vendors

The example below joins the Empmast and Vendor tables using address as the common key field (the Address and Vendor_Street fields).

The FUZZYJOIN command creates a new table with either exactly matched or fuzzy matched primary and secondary records. The result is a list of any employees and vendors with either an identical address, or a similar address.

FUZZYJOIN with the Dice coefficient algorithm

```
OPEN Empmast PRIMARY
OPEN Vendor SECONDARY
```

```
FUZZYJOIN DICE PERCENT 0.8000 NGRAM 2 PKEY Address SKEY Vendor_Street
FIELDS Employee_Number First_Name Last_Name Address WITH Vendor_Number
Vendor_Name Vendor_Street OPEN TO "Employee_Vendor_Match"
```

FUZZYJOIN with the Levenshtein distance algorithm

```
OPEN Empmast PRIMARY
OPEN Vendor SECONDARY
FUZZYJOIN LEVDISTANCE DISTANCE 5 PKEY Address SKEY Vendor_Street FIELDS
Employee_Number First_Name Last_Name Address WITH Vendor_Number Vendor_
Name Vendor_Street OPEN TO "Employee_Vendor_Match"
```

Include all fields

This version of the FUZZYJOIN command includes all fields from the primary and secondary tables in the joined output table.

```
OPEN Empmast PRIMARY
OPEN Vendor SECONDARY
FUZZYJOIN LEVDISTANCE DISTANCE 5 PKEY Address SKEY Vendor_Street FIELDS
ALL WITH ALL OPEN TO "Employee_Vendor_Match"
```

Improve the effectiveness of fuzzy matching

The example below uses the SORTWORDS( ) function to improve the effectiveness of fuzzy matching between the Address and Vendor_Street fields. Use of the UPPER( ) function ensures that case does not affect the sorting of elements in key field values.

```
OPEN Empmast PRIMARY
OPEN Vendor SECONDARY
FUZZYJOIN LEVDISTANCE DISTANCE 5 PKEY SORTWORDS(UPPER(Address)) SKEY
SORTWORDS(UPPER(Vendor_Street)) FIELDS Employee_Number First_Name Last_
Name Address WITH Vendor_Number Vendor_Name Vendor_Street OPEN TO
"Employee_Vendor_Match"
```

Remarks

For more information about how this command works, see "Fuzzy join" on page 913.

Case sensitivity

The FUZZYJOIN command is not case-sensitive, regardless of which fuzzy matching algorithm you use. So "SMITH" is equivalent to "smith."

Leading and trailing blanks

The FUZZYJOIN command automatically trims leading and trailing blanks in fields, regardless of which fuzzy matching algorithm you use. There is no need to use the TRIM( ) or ALLTRIM( ) functions when specifying the primary and secondary key fields.

Improving the effectiveness of FUZZYJOIN

Three techniques can significantly improve the effectiveness of the FUZZYJOIN command:

- sorting individual elements in primary and secondary key field values
- removing generic elements from primary and secondary key field values
- harmonizing primary and secondary key field values

These techniques allow you to use tighter fuzzy settings and still get the same fuzzy matches, while reducing the number of false positive matches. You can use the techniques separately, or in combination.

Sorting individual elements in key field values

The SORTWORDS( ) function can improve the effectiveness of the FUZZYJOIN command by sorting individual elements in primary and secondary key field values into a sequential order.

Sorting elements, such as the components of an address, can make key field values with the same information, but a different format, more closely resemble each other. A closer resemblance improves the chances that key field values are selected as fuzzy matches for each other.

For more information, see "SORTWORDS( ) function" on page 2378.

For a video providing an overview of SORTWORDS( ), see Fuzzy Matching Using SORTWORDS() (English only).

> **Note**
>
> Sorting elements in key field values is best suited for fuzzy joining using the Levenshtein distance algorithm.
>
> Sorting elements when fuzzy joining using the Dice coefficient algorithm may or may not be beneficial. Test a set of sample data before deciding whether to use SORTWORDS( ) in conjunction with the Dice coefficient algorithm in a production setting.

> **Caution**
>
> If you use SORTWORDS( ) in conjunction with the FUZZYJOIN command you must apply SORTWORDS( ) to both strings or both fields being compared.

Removing generic elements from key field values

The OMIT( ) function can improve the effectiveness of the FUZZYJOIN command by removing generic elements such as "Corporation" or "Inc.", or characters such as commas, periods, and ampersands (&), from primary and secondary key field values.

Removal of generic elements and punctuation focuses fuzzy matching on just the portion of the key field values where a meaningful difference may occur.

For more information, see "OMIT( ) function" on page 2268.

Harmonizing key field values

The REPLACE( ) or REGEXREPLACE( ) functions can improve the effectiveness of the FUZZYJOIN command by harmonizing variant forms of the same element in primary and secondary key field values. For example, you could harmonize "Street", "St.", and "St" to use the single value "St".

Harmonizing elements can make key field values with the same information, but a different format, more closely resemble each other. A closer resemblance improves the chances that key field values are selected as fuzzy matches for each other.

For more information, see "REPLACE( ) function" on page 2349 for straightforward replacements, and "REGEXREPLACE( ) function" on page 2335 for more complex replacements.

GAPS command

Detects whether a numeric or datetime field in an Analytics table contains one or more gaps in sequential data.

Syntax

```
GAPS <ON> key_field <D> <UNFORMATTED> <PRESORT> <MISSING limit>
<HEADER header_text> <FOOTER footer_text> <IF test> <WHILE test> <FIRST
range|NEXT range> <TO {SCREEN|table_name|filename|PRINT}> <LOCAL> <APPEND>
<OPEN>
```

Parameters

Name	Description
ON *key_field* D	The field or expression to check for gaps. Include D to sort the key field in descending order. The default sort order is ascending.
UNFORMATTED optional	Suppresses page headings and page breaks when the results are output to a file.
PRESORT optional	Sorts the table on the key field before executing the command. **Note** You cannot use PRESORT inside the GROUP command.
MISSING *limit* optional	The output results contain individual missing items rather than gap ranges. The *limit* value specifies the maximum number of missing items to report for each identified gap. The default value is 5. If the limit is exceeded for a particular gap, the missing items are reported as a range for that particular gap. The *limit* value does not restrict the total number of missing items reported, it only restricts the number of missing items reported within a specific gap.
HEADER *header_text* optional	The text to insert at the top of each page of a report. *header_text* must be specified as a quoted string. The value overrides the Analytics HEADER system variable.

Name	Description
FOOTER *footer_text* optional	The text to insert at the bottom of each page of a report. *footer_text* must be specified as a quoted string. The value overrides the Analytics FOOTER system variable.
IF *test* optional	A conditional expression that must be true in order to process each record. The command is executed on only those records that satisfy the condition. **Note** The IF parameter is evaluated against only the records remaining in a table after any scope parameters have been applied (WHILE, FIRST, NEXT).
WHILE *test* optional	A conditional expression that must be true in order to process each record. The command is executed until the condition evaluates as false, or the end of the table is reached. **Note** If you use WHILE in conjunction with FIRST or NEXT, record processing stops as soon as one limit is reached.
FIRST *range* | NEXT *range* optional	The number of records to process: ◦ **FIRST** - start processing from the first record until the specified number of records is reached ◦ **NEXT** - start processing from the currently selected record until the specified number of records is reached Use *range* to specify the number of records to process. If you omit FIRST and NEXT, all records are processed by default.
TO SCREEN | *table_name* | *filename* | PRINT optional	The location to send the results of the command to: ◦ **SCREEN** - displays the results in the Analytics display area **Tip** You can click any linked result value in the display area to drill down to the associated record or records in the source table. ◦ ***table_name*** - saves the results to an Analytics table Specify *table_name* as a quoted string with a .FIL file extension. For example: `TO "Output.FIL"` By default, the table data file (.FIL) is saved to the folder containing the Analytics project. Use either an absolute or relative file path to save the data file to a different, existing folder: • `TO "C:\Output.FIL"` • `TO "Results\Output.FIL"`

Name	Description
	Note Table names are limited to 64 alphanumeric characters, not including the .FIL extension. The name can include the underscore character (_), but no other special characters, or any spaces. The name cannot start with a number. ◦ ***filename*** - saves the results to a file Specify *filename* as a quoted string with the appropriate file extension. For example: `TO "Output.TXT"` By default, the file is saved to the folder containing the Analytics project. Use either an absolute or relative file path to save the file to a different, existing folder: • `TO "C:\Output.TXT"` • `TO "Results\Output.TXT"` ◦ **PRINT** - sends the results to the default printer
LOCAL optional	Saves the output file in the same location as the Analytics project. **Note** Applicable only when running the command against a server table with an output file that is an Analytics table. The LOCAL parameter must immediately follow the TO parameter.
APPEND optional	Appends the command output to the end of an existing file instead of overwriting it. **Note** You must ensure that the structure of the command output and the existing file are identical: • the same fields • the same field order • matching fields are the same length • matching fields are the same data type Analytics appends output to an existing file regardless of its structure. If the structure of the output and the existing file do not match, jumbled, missing, or inaccurate data can result.
OPEN optional	Opens the table created by the command after the command executes. Only valid if the command creates an output table.

Analytics output variables

Name	Contains
GAPDUP*n*	The total number of gaps, duplicates, or fuzzy duplicate groups identified by the command.

Examples

Testing for missing invoice numbers

You use GAPS to ensure that there are no invoice numbers missing from an **Invoices** table:

```
OPEN Invoices
GAPS ON Inv_Num PRESORT TO "Invoices_Gaps.fil"
```

Remarks

For more information about how this command works, see "Testing for gaps" on page 1195.

Using GAPS on character fields

In addition to testing numeric or datetime fields, you can also test for gaps in numeric data that appears in a character field. For example, you can test check numbers, which are typically formatted as character data.

If letters and numbers appear together in a character field, only the numbers are tested and the letters are ignored.

GROUP command

Executes one or more ACLScript commands on a record before moving to the next record in the table, with only one pass through the table. Command execution can be controlled by conditions.

Syntax

```
GROUP <IF test> <WHILE test> <FIRST range|NEXT range>
  command
  <...n>
<ELSE IF test>
  command
  <...n>
<ELSE>
  command
  <...n>
END
```

> **Note**
> Some Analytics commands cannot be used with the GROUP command. For more information, see "Commands that can be used inside the GROUP command" on page 1745.

Parameters

Name	Description
IF *test* optional	A conditional expression that must be true in order to process each record. The command is executed on only those records that satisfy the condition. **Note** The IF parameter is evaluated against only the records remaining in a table after any scope parameters have been applied (WHILE, FIRST, NEXT).
WHILE *test* optional	A conditional expression that must be true in order to process each record. The command is executed until the condition evaluates as false, or the end of the table is reached. **Note** If you use WHILE in conjunction with FIRST or NEXT, record processing stops as soon as one limit is reached.

Name	Description
FIRST *range* | NEXT *range* optional	The number of records to process: ◦ **FIRST** - start processing from the first record until the specified number of records is reached ◦ **NEXT** - start processing from the currently selected record until the specified number of records is reached Use *range* to specify the number of records to process. If you omit FIRST and NEXT, all records are processed by default.
command <...*n*>	One or more ACLScript commands to execute inside the GROUP. For a complete list of commands supported inside GROUP, see "Commands that can be used inside the GROUP command" on page 1745. If there is a preceding IF or ELSE IF, the test must evaluate to true. If the command is listed under ELSE, the command is executed if there are records that have not been processed by any of the preceding commands. You can include multiple commands, with each command starting on a separate line.
ELSE IF *test* optional	Opens an ELSE IF block for the GROUP command. The condition tests records that did not match the GROUP command test, or any previous ELSE IF tests. You can include multiple ELSE IF tests and they are evaluated from top to bottom, until the record evaluates to true and the commands that follow that ELSE IF statement are executed.
ELSE optional	Opens an ELSE block for the GROUP command. The commands that follow are executed for records that evaluated to false for all of the previous tests.
END	The end of the GROUP command.

Examples

Simple GROUP

Simple groups start with a GROUP command, are followed by a series of commands, and terminate with an END command:

```
GROUP
  COUNT
  HISTOGRAM ON Quantity MINIMUM 0 MAXIMUM 100 INTERVALS 10
  CLASSIFY ON Location SUBTOTAL Quantity
END
```

GROUP IF

Conditional groups execute commands based on whether a condition is true or false. The following GROUP command is executed only on records with a **Product_class** value less than 5:

```
GROUP IF Product_class < "05"
  COUNT
  HISTOGRAM ON Quantity MINIMUM 0 MAXIMUM 100 INTERVALS 10
  CLASSIFY ON Location SUBTOTAL Quantity
END
```

GROUP IF ...ELSE

Records that do not meet the test condition are ignored unless you include an ELSE block.

Any number of commands can follow an ELSE statement. In the following example, all records that do not meet the condition are processed by having their **Quantity** field totaled:

```
GROUP IF Product_class < "05"
  COUNT
  HISTOGRAM ON Quantity MINIMUM 0 MAXIMUM 100 INTERVALS 10
  CLASSIFY ON Location SUBTOTAL Quantity
ELSE
  TOTAL Quantity
END
```

GROUP IF...ELSE IF...ELSE

You can include multiple ELSE IF blocks within a group, as long as each ELSE IF block contains a different test. In the following example, the ELSE IF blocks, and the ELSE block, produce four totals:

```
GROUP IF Product_class < "05"
  COUNT
  HISTOGRAM ON Quantity MINIMUM 0 MAXIMUM 100 INTERVALS 10
```

```
  CLASSIFY ON Location SUBTOTAL Quantity
ELSE IF Product_class = "05"
  TOTAL Quantity
ELSE IF Product_class = "06"
  TOTAL Quantity
ELSE IF Product_class = "07"
  TOTAL Quantity
ELSE
  TOTAL Quantity
END
```

Nested GROUP commands

Nested groups refer to groups contained within other groups. Nested groups provide a powerful way for you to control which commands are executed for which records. Most applications do not require such an advanced level of functionality, but it is available, if necessary.

As with other groups, use the END command to terminate a nested group. Analytics starts processing the data only after all group commands have been terminated:

```
GROUP IF Product_class < "05"
  COUNT
  STRATIFY ON Quantity SUBTOTAL Quantity MIN 0 MAX 100 INT 10
  GROUP IF Quantity > 0
    STATISTICS ON Quantity
    HISTOGRAM ON Quantity
  END
ELSE
  TOTAL Quantity
END
```

In this example, all of the commands from COUNT up to and including the next GROUP are executed only if **Product_class** is less than 05.

The STATISTICS and HISTOGRAM commands are executed if **Quantity** is greater than zero. However, because the second GROUP command is nested, the STATISTICS and HISTOGRAM commands are executed only for records that meet the conditions **Product_class** < "05" and **Quantity** > 0.

Generating system variables inside a GROUP

You can use GROUP to create multiple system variables for a single command.

Normally, when you run a command such as TOTAL, COUNT, or STATISTICS, only one system variable is generated. Each time you run the command, you overwrite the value from the last execution of the command. Commands that run inside a GROUP create a specific variable for each instance of the command inside the GROUP.

In this example, the TOTAL command calculates the sum of the **Amount** field for each product class in the **Metaphor_Trans_2002** table. When the code runs, the following variables are generated and can be used in subsequent commands after the GROUP:

- **TOTAL2** - the sum of the **Amount** field for product class 03
- **TOTAL3** - the sum of the **Amount** field for product class 05
- **TOTAL4** - the sum of the **Amount** field for product class 08
- **TOTAL5** - the sum of the **Amount** field for product class 09

```
OPEN Metaphor_Trans_2002
GROUP
  TOTAL AMOUNT IF PRODCLS = "03"
  TOTAL AMOUNT IF PRODCLS = "05"
  TOTAL AMOUNT IF PRODCLS = "08"
  TOTAL AMOUNT IF PRODCLS = "09"
END
CLOSE Metaphor_Trans_2002
```

Remarks

> **Tip**
> For a detailed tutorial covering the GROUP and LOOP commands, see "Grouping and looping" on page 1416.

Commands that can be used inside the GROUP command

The table below lists the Analytics commands that can be used inside the GROUP command.

If a command is not listed below, it cannot be used inside GROUP.

AGE	ASSIGN	BENFORD
CLASSIFY	COMMENT	COUNT
CROSSTAB	DUPLICATES	EXPORT

EXTRACT	GAPS	GROUP
HISTOGRAM	JOIN	LIST
LOOP	MERGE	PROFILE
REPORT	SEQUENCE	STATISTICS
STRATIFY	SUMMARIZE	TOTAL
VERIFY		

Grouping and looping

The GROUP command allows you to execute several commands on a record before moving to the next record in the table, which can significantly reduce processing time.

You can use the LOOP command inside the GROUP command if you need to execute a series of commands more than once against a record.

Using variables with GROUP

User-defined variables

To use a variable inside the GROUP command, define the variable before you enter the GROUP block.

> **Note**
> While you can initialize and define a variable inside a GROUP block, it is not recommended. Variables initialized inside a GROUP may cause unexpected results when used.

Inside a GROUP, you can evaluate variables using variable substitution. The value of the variable remains the same as when the GROUP is entered.

You cannot define a variable inside a GROUP and then reference it using variable substitution:

```
ASSIGN v_test = "hello"
GROUP
  ASSIGN v_test2 = "%v_test% world"
  COMMENT this would be invalid: v_test3 = "%v_test2% again"
END
```

System-defined variables

Certain commands such as TOTAL and STATISTICS generate system variables based on calculations that the commands perform. If you use a GROUP to execute these commands, any system variables that result are numbered consecutively, starting at the line number of the command inside the GROUP (excluding empty lines) and running to *n*. The value of *n* increases by 1 for each line number in the GROUP.

> **Note**
> You must wait until the GROUP completes before using any system generated variables created inside the GROUP. The command must run against each record in the table before the variable is available. Use these variables after the closing END keyword of the GROUP.

In the following example, the first TOTAL command generates the variable TOTAL2 and the second generates TOTAL4. Both of these variables are available to use in subsequent commands once the GROUP completes:

```
GROUP
  TOTAL Discount IF Order_Priority = "Low"
  ASSIGN v_var = "test"
  TOTAL Discount IF Order_Priority = "High"
END
```

Syntax notes

- The multiline syntax listed for the GROUP command is required, and therefore the GROUP command cannot be entered in the command line.
- Each GROUP command must be terminated with an END command.
- When you use the GROUP command in your scripts, you can improve the readability of the command block by indenting the commands listed inside the group.

HELP command

Launches the Analytics Help Docs in a browser.

Syntax

```
HELP
```

HISTOGRAM command

Groups records based on values in a character or numeric field, counts the number of records in each group, and displays the groups and counts in a bar chart.

Syntax

```
HISTOGRAM {<ON> character_field|<ON> numeric_field MINIMUM value MAXIMUM value
{<INTERVALS number>|FREE interval_value <...n> last_interval}} <TO
{SCREEN|filename|GRAPH|PRINT}> <IF test> <WHILE test> <FIRST range|NEXT range>
<HEADER header_text> <FOOTER footer_text> <KEY break_field> <SUPPRESS>
<COLUMNS number> <APPEND> <OPEN>
```

Parameters

Name	Description
ON *character_field*	The character field or expression to use for the histogram.
ON *numeric_field*	The numeric field or expression to use for the histogram.
MINIMUM *value*	Applies to numeric fields only. The minimum value of the first numeric interval. MINIMUM is optional if you are using FREE, otherwise it is required.
MAXIMUM *value*	Applies to numeric fields only. The maximum value of the last numeric interval. MAXIMUM is optional if you are using FREE, otherwise it is required.
INTERVALS *number* optional	Applies to numeric fields only. The number of equal-sized intervals Analytics produces over the range specified by the MINIMUM and MAXIMUM values. If you do not specify a number of intervals, the default number is used. The default is specified by the **Intervals** number on the **Command** tab in the **Options** dialog box.
FREE *interval_value <...n> last_interval* optional	Applies to numeric fields only. Creates custom-sized intervals by specifying the start point of each interval and the end point of the last interval. If you specify MINIMUM and MAXIMUM values, those values are the start point of

Name	Description
	the first interval and the end point of the last interval, and each *interval_value* creates an additional interval within the range. The interval values you specify must be greater than the MINIMUM value, and equal to or less than the MAXIMUM value. Interval values must be in numeric sequence and cannot contain duplicate values: `FREE -1000, 0, 1000, 2000, 3000` If you specify both FREE and INTERVALS, then INTERVALS is ignored.
TO SCREEN | *filename* | GRAPH | PRINT	The location to send the results of the command to: ◦ **SCREEN** - displays the results in the Analytics display area **Tip** You can click any linked result value in the display area to drill down to the associated record or records in the source table. ◦ ***filename*** - saves the results to a file Specify *filename* as a quoted string with the appropriate file extension. For example: `TO "Output.TXT"` By default, the file is saved to the folder containing the Analytics project. Use either an absolute or relative file path to save the file to a different, existing folder: • `TO "C:\Output.TXT"` • `TO "Results\Output.TXT"` ◦ **GRAPH** - displays the results in a graph in the Analytics display area ◦ **PRINT** - sends the results to the default printer **Note** Histogram results output to a file appear as a textual representation of a bar chart.
IF *test* optional	A conditional expression that must be true in order to process each record. The command is executed on only those records that satisfy the condition. **Note** The IF parameter is evaluated against only the records remaining in a table after any scope parameters have been applied (WHILE, FIRST, NEXT).
WHILE *test* optional	A conditional expression that must be true in order to process each record. The command is executed until the condition evaluates as false, or the end of the table is reached. **Note** If you use WHILE in conjunction with FIRST or NEXT, record processing stops as soon as one limit is reached.

Name	Description
FIRST *range* | NEXT *range* optional	The number of records to process: ◦ **FIRST** - start processing from the first record until the specified number of records is reached ◦ **NEXT** - start processing from the currently selected record until the specified number of records is reached Use *range* to specify the number of records to process. If you omit FIRST and NEXT, all records are processed by default.
HEADER *header_text* optional	The text to insert at the top of each page of a report. *header_text* must be specified as a quoted string. The value overrides the Analytics HEADER system variable.
FOOTER *footer_text* optional	The text to insert at the bottom of each page of a report. *footer_text* must be specified as a quoted string. The value overrides the Analytics FOOTER system variable.
KEY *break_field* optional	The field or expression that groups subtotal calculations. A subtotal is calculated each time the value of *break_field* changes. *break_field* must be a character field or expression. You can specify only one field, but you can use an expression that contains more than one field.
SUPPRESS optional	Values above the MAXIMUM value and below the MINIMUM value are excluded from the command output.
COLUMNS *number* optional	The length of the x-axis in the textual representation of the bar chart if you output histogram results to a text file. The number value is the number of character spaces (text columns) to use for the x-axis (and the y-axis labels). If you omit COLUMNS, the default of 78 character spaces is used.
APPEND optional	Appends the command output to the end of an existing file instead of overwriting it. **Note** You must ensure that the structure of the command output and the existing file are identical: • the same fields • the same field order • matching fields are the same length • matching fields are the same data type Analytics appends output to an existing file regardless of its structure. If the structure of the output and the existing file do not match, jumbled, missing, or inaccurate data can result.
OPEN optional	Opens the table created by the command after the command executes. Only valid if the command creates an output table.

Examples

Basic histogram for hourly salary

You use HISTOGRAM to create a graph showing the distribution of wages between 0 and 100 dollars per hour:

```
HISTOGRAM ON Rate MINIMUM 0 MAXIMUM 100 TO GRAPH
```

Histogram with defined intervals for hourly salary

Continuing from the previous example, you use HISTOGRAM to specify the ranges in the graph in a more meaningful way.

Most of the wages fall between 20 and 50 dollars per hour, so the graph includes the following number of intervals:

- three in the 20 to 50 range
- one for 0-20
- one for 50-100
- one for > 100

```
HISTOGRAM ON Rate MINIMUM 0 MAXIMUM 100 FREE 20,30,40,50,100 TO GRAPH
```

Remarks

For more information about how this command works, see "Creating histograms" on page 1285.

Populating low and high values

You can run the STATISTICS or PROFILE commands on a numeric field before running the HISTOGRAM command to automatically populate the MINIMUM and MAXIMUM parameter values with the lowest and highest values in the field.

Related commands

Creating a histogram using a character field is similar to classifying. Creating a histogram using a numeric field is similar to stratifying.

Unlike the other grouping operations in Analytics, histograms do not support subtotaling numeric fields.

IF command

Specifies a condition that must evaluate to true in order to execute a command.

Syntax

```
IF test command
```

Parameters

Name	Description
test	The condition that must be met for *command* to be run.
command	Any valid ACLScript command to run if *test* evaluates to true.

Examples

Running a command conditionally

You want to use CLASSIFY on a table, but only if the *v_counter* variable is greater than ten:

```
IF v_counter > 10 CLASSIFY ON Location TO "Count_by_Location.fil" OPEN
```

Running a command based on a user decision

You want to allow the script user to decide whether to classify a table.

In your script, you include a dialog box with a check box that if selected allows the CLASSIFY command to run. The check box stores a True or False input value in the logical variable *v_classify_checkbox*.

You use an IF test to determine the value of *v_classify_checkbox*, and if the value is True, CLASSIFY executes:

```
IF v_classify_checkbox=T CLASSIFY ON Location TO "Count_by_Location.fil"
OPEN
```

Remarks

IF command versus IF parameter

The logic of the IF command differs from the IF parameter that is supported by most commands:

- **IF command** – determines whether the associated command runs or not, based on the value of the test expression
- **IF parameter** – determines whether the command runs against each record in an Analytics table based on the value of the test expression

Decision making in scripts

In a script, you can enter a series of IF command tests and run different commands based on the results. The IF command can be also be used to test the value of a variable to determine if further processing should occur.

IMPORT ACCESS command

Creates an Analytics table by defining and importing a Microsoft Access database file.

Syntax

```
IMPORT ACCESS TO table <PASSWORD num> import_filename FROM source_filename
TABLE input_tablename CHARMAX max_field_length MEMOMAX max_field_length
```

Parameters

Name	Description
TO *table*	The name of the Analytics table to import the data into. **Note** Table names are limited to 64 alphanumeric characters. The name can include the underscore character (_), but no other special characters, or any spaces. The name cannot start with a number.
PASSWORD *num* optional	Used only with password-protected Access files. The password definition to use. You do not use PASSWORD *num* to prompt for, or specify, an actual password. The password definition refers to a password previously supplied or set using the PASSWORD command, the SET PASSWORD command, or the PASSWORD analytic tag. *num* is the number of the password definition. For example, if two passwords have been previously supplied or set in a script, or when scheduling an analytic, `PASSWORD 2` specifies that password #2 is used. For more information about supplying or setting passwords, see: ◦ "PASSWORD command" on page 1893 ◦ "SET command" on page 1960 ◦ "PASSWORD tag" on page 2530
import_filename	The name of the Analytics data file to create. Specify *import_filename* as a quoted string with a .FIL file extension. For example, `"Invoices.FIL"`. By default, the data file (.FIL) is saved to the folder containing the Analytics project.

Name	Description
	Use either an absolute or relative file path to save the data file to a different, existing folder: ◦ `"C:\data\Invoices.FIL"` ◦ `"data\Invoices.FIL"`
FROM *source_filename*	The name of the source data file. *source_filename* must be a quoted string. If the source data file is not located in the same directory as the Analytics project, you must use an absolute path or a relative path to specify the file location: ◦ `"C:\data\source_filename"` ◦ `"data\source_filename"`
TABLE *input_tablename*	The name of the table in the Microsoft Access database file to import.
CHARMAX *max_field_length*	The maximum length in characters for any field in the Analytics table that originates as character data in the source from which you are importing. You can specify from 1 to 255 characters.
MEMOMAX *max_field_length*	The maximum length in characters for text, note, or memo fields you are importing. You can specify from 1 to 32767 characters (non-Unicode Analytics), or 16383 characters (Unicode Analytics).

Examples

For more information about how this command works, see "Import a Microsoft Access database file" on page 247.

Importing into a table

You have a Microsoft Access file called `Acceptable_Codes.mdb`. You need to import the **[Acceptable_Codes]** table from the file into Analytics. To do this, you use the following command and create a table called **acc_codes** in Analytics.

The length of imported character or memo fields is set to the length of the longest value in the field, or to the specified maximum number of characters, whichever is shorter:

```
SET ECHO NONE
SET PASSWORD 1 TO "qr347wx"
SET ECHO ON
IMPORT ACCESS TO acc_codes PASSWORD 1 "C:\ACL DATA\Sample Data
```

```
Files\acc_codes.fil" FROM "Acceptable_Codes.mdb" TABLE "[Acceptable_
Codes]" CHARMAX 60 MEMOMAX 70
```

IMPORT DELIMITED command

Creates an Analytics table by defining and importing a delimited text file.

Syntax

```
IMPORT DELIMITED TO table import_filename FROM source_filename <SERVER pro-
file_name> source_char_encoding SEPARATOR {char|TAB|SPACE} QUALIFIER
{char|NONE} <CONSECUTIVE> STARTLINE line_number <KEEPTITLE> <CRCLEAR>
<LFCLEAR> <REPLACENULL> <ALLCHAR> {ALLFIELDS|[field_syntax] <...n> <IGNORE
field_num> <...n>}
```

```
field_syntax ::=
FIELD name type AT start_position DEC value WID bytes PIC format AS display_
name
```

Parameters

Name	Description
TO *table*	The name of the Analytics table to import the data into. **Note** Table names are limited to 64 alphanumeric characters. The name can include the underscore character (_), but no other special characters, or any spaces. The name cannot start with a number.
import_filename	The name of the Analytics data file to create. Specify *import_filename* as a quoted string with a .FIL file extension. For example, `"Invoices.FIL"`. By default, the data file (.FIL) is saved to the folder containing the Analytics project. Use either an absolute or relative file path to save the data file to a different, existing folder: ◦ `"C:\data\Invoices.FIL"` ◦ `"data\Invoices.FIL"`
FROM *source_filename*	The name of the source data file. *source_filename* must be a quoted string.

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>If the source data file is not located in the same directory as the Analytics project, you must use an absolute path or a relative path to specify the file location:
<code>"C:\data\source_filename"</code>
<code>"data\source_filename"</code></td></tr>
<tr><td>SERVER profile_name
optional</td><td>The server profile name for the AX Server where the data you want to import is located.</td></tr>
<tr><td>source_char_encoding</td><td>The character set and encoding of the source data.
Depending on which edition of Analytics you are using, and the encoding of the source data, specify the appropriate code:<table><tr><th>Code</th><th>Analytics edition</th><th>Source data encoding</th></tr><tr><td><code>0</code></td><td>Non-Unicode edition</td><td>all data</td></tr><tr><td><code>0</code></td><td>Unicode edition</td><td>ASCII data</td></tr><tr><td><code>2</code></td><td>Unicode edition</td><td>Unicode data, UTF-16 LE encoding</td></tr><tr><td><code>3 numeric_code</code></td><td>Unicode edition</td><td>Unicode data that does not use UTF-16 LE encoding
To determine the numeric code that matches the source data encoding, perform an import using the Data Definition Wizard, select the Encoded Text option, and find the matching encoding in the accompanying drop-down list.
To specify the code, specify <code>3</code>, followed by a space, and then the numeric code.</td></tr></table></td></tr>
<tr><td>SEPARATOR char
| TAB | SPACE</td><td>The separator character (delimiter) used between fields in the source data. You must specify the character as a quoted string.
You can specify a tab or a space separator by typing the character between double quotation marks, or by using a keyword:
<code>SEPARATOR " "</code> or <code>SEPARATOR TAB</code>
<code>SEPARATOR " "</code> or <code>SEPARATOR SPACE</code></td></tr>
<tr><td>QUALIFIER char | NONE</td><td>The text qualifier character used in the source data to wrap and identify field values. You must specify the character as a quoted string.
To specify the double quotation mark character as the text qualifier, enclose the character in single quotation marks: <code>QUALIFIER '"'</code>.
You can specify that there are no text qualifiers using either of these methods:
<code>QUALIFIER ""</code>
<code>QUALIFIER NONE</code></td></tr>
<tr><td>CONSECUTIVE</td><td>Consecutive text qualifiers are treated as a single qualifier.</td></tr>
</table>

Name	Description
optional	
STARTLINE *line_number*	The line number on which to start reading the file. For example, if the first three lines of a file contain header information that you do not want, specify `STARTLINE 4` to start reading data on the fourth line.
KEEPTITLE optional	◦ **KEEPTITLE used with ALLFIELDS** - Treat the line number specified by STARTLINE as field names instead of data. If you omit KEEPTITLE, generic field names are used and the line number specified by STARTLINE is treated as data. ◦ **KEEPTITLE used with individual FIELD syntax** - Do not import the line number specified by STARTLINE. FIELD *name* specifies the field names. If you omit KEEPTITLE, the line number specified by STARTLINE is treated as data. FIELD *name* specifies the field names.
CRCLEAR optional	Replaces any CR characters (carriage return) that occur between text qualifiers with space characters. You must specify QUALIFIER with a *char* value to use CRCLEAR. If you use both CRCLEAR and LFCLEAR, CRCLEAR must come first.
LFCLEAR optional	Replaces any LF characters (line feed) that occur between text qualifiers with space characters. You must specify QUALIFIER with a *char* value to use LFCLEAR. If you use both CRCLEAR and LFCLEAR, CRCLEAR must come first.
REPLACENULL optional	Replaces any NUL characters that occur in the delimited file with space characters. The number of any replaced NUL characters is recorded in the log.
ALLCHAR optional	The Character data type is automatically assigned to all the imported fields. **Tip** Assigning the Character data type to all the imported fields simplifies the process of importing delimited text files. Once the data is in Analytics, you can assign different data types, such as Numeric or Datetime, to the fields, and specify format details. ALLCHAR is useful if you are importing a table with identifier fields automatically assigned the Numeric data type by Analytics when in fact they should use the Character data type.
ALLFIELDS	All fields in the source data file are imported. For information about how Analytics assigns data types when you use ALLFIELDS, see "Remarks" on page 1765. **Note** If you specify ALLFIELDS, do not specify any individual FIELD syntax, or IGNORE.
FIELD *name type*	The individual fields to import from the source data file, including the name and data type of the field. To exclude a field from being imported, do not specify it.

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>For information about type, see "Identifiers for field data types" on page 1765.

Note
type is ignored if you specify ALLCHAR.</td></tr>
<tr><td>AT start_position</td><td>The starting byte position of the field in the Analytics data file.

Note<table><tr><td>non-Unicode Analytics</td><td>1 byte = 1 character</td></tr><tr><td>Unicode Analytics</td><td>2 bytes = 1 character</td></tr></table>In Unicode Analytics, typically you should specify an odd-numbered starting byte position. Specifying an even-numbered starting position can cause characters to display incorrectly.</td></tr>
<tr><td>DEC value</td><td>The number of decimals for numeric fields.

Note
DEC is ignored if you specify ALLCHAR.</td></tr>
<tr><td>WID bytes</td><td>The length in bytes of the field in the Analytics table layout.

Note<table><tr><td>non-Unicode Analytics</td><td>1 byte = 1 character</td></tr><tr><td>Unicode Analytics</td><td>2 bytes = 1 character</td></tr></table>In Unicode Analytics, specify an even number of bytes only. Specifying an odd number of bytes can cause characters to display incorrectly.</td></tr>
<tr><td>PIC format</td><td>Note
Applies to numeric or datetime fields only.

◦ numeric fields - the display format of numeric values in Analytics views and reports
◦ datetime fields - the physical format of datetime values in the source data (order of date and time characters, separators, and so on)

Note
For datetime fields, format must exactly match the physical format in the source data. For example, if the source data is 12/31/2014, you must enter the format as <code>"MM/DD/YYYY"</code>.

format must be enclosed in quotation marks.</td></tr>
</table>

Name	Description
	Note PIC is ignored if you specify ALLCHAR.
AS *display_name*	The display name (alternate column title) for the field in the view in the new Analytics table. Specify *display_name* as a quoted string. Use a semi-colon (;) between words if you want a line break in the column title. AS is required when you are defining FIELD. To make the display name the same as the field name, enter a blank *display_name* value using the following syntax: `AS ""`. Make sure there is no space between the two double quotation marks.
IGNORE *field_num* <...*n*> optional	Excludes the field from the table layout. *field_num* specifies the position of the excluded field in the source data file. For example, `IGNORE 5` excludes the fifth field in the source data file from the Analytics table layout. **Note** The data in the field is still imported, but it is undefined, and does not appear in the new Analytics table. The data can be defined later, if necessary, and added to the table. To completely exclude a field from being imported, do not specify it when you specify fields individually.

Examples

Import all fields

You import all fields from a comma delimited file to an Analytics table named **Employees**. The file uses double quotation marks as the text qualifiers. Data types are automatically assigned based on the set of rules outlined in "Remarks" on page 1765:

```
IMPORT DELIMITED TO Employees "Employees.fil" FROM "Employees.csv" 0
SEPARATOR "," QUALIFIER '"' CONSECUTIVE STARTLINE 1 KEEPTITLE ALLFIELDS
```

Import all fields, automatically assign a Character data type

You import all fields from a comma delimited file to an Analytics table named **Employees**. The file uses double quotation marks as the text qualifiers. The Character data type is automatically assigned to all imported fields:

```
IMPORT DELIMITED TO Employees "Employees.fil" FROM "Employees.csv" 0
SEPARATOR "," QUALIFIER '"' CONSECUTIVE STARTLINE 1 KEEPTITLE ALLCHAR
ALLFIELDS
```

Import specified fields, automatically assign a Character data type

You import specified fields from a tab delimited file to an Analytics table named **Employees**. The file uses double quotation marks as the text qualifiers. The Character data type is automatically assigned to all imported fields:

```
IMPORT DELIMITED TO Employees "Employees.fil" FROM "Employees.csv" 0
SEPARATOR TAB QUALIFIER '"' CONSECUTIVE STARTLINE 1 KEEPTITLE ALLCHAR
FIELD "First_Name" C AT 1 DEC 0 WID 25 PIC "" AS "First Name" FIELD
"Last_Name" C AT 26 DEC 0 WID 25 PIC "" AS "Last Name" FIELD "CardNum" C
AT 51 DEC 0 WID 16 PIC "" AS "Card Num" FIELD "EmpNo" C AT 67 DEC 0 WID
6 PIC "" AS "Emp Num" FIELD "HireDate" C AT 73 DEC 0 WID 10 PIC "" AS
"Hire Date" FIELD "Salary" C AT 83 DEC 0 WID 5 PIC "" AS "" FIELD
"Bonus_2016" C AT 88 DEC 0 WID 10 PIC "" AS "Bonus 2016"
```

Import specified fields, assign data types individually

You import specified fields from a semi-colon delimited file to an Analytics table named **Employees**. The file does not use text qualifiers. You specify the data type of each imported field:

```
IMPORT DELIMITED TO Employees "Employees.fil" FROM "Employees.csv" 0
SEPARATOR ";" QUALIFIER "" CONSECUTIVE STARTLINE 1 KEEPTITLE FIELD
"First_Name" C AT 1 DEC 0 WID 25 PIC "" AS "First Name" FIELD "Last_
Name" C AT 26 DEC 0 WID 25 PIC "" AS "Last Name" FIELD "CardNum" C AT 51
DEC 0 WID 16 PIC "" AS "Card Num" FIELD "EmpNo" C AT 67 DEC 0 WID 6 PIC
"" AS "Emp Num" FIELD "HireDate" D AT 73 DEC 0 WID 10 PIC "MM/DD/YYYY"
```

```
AS "Hire Date" FIELD "Salary" N AT 83 DEC 0 WID 5 PIC "" AS "" FIELD
"Bonus_2016" N AT 88 DEC 2 WID 10 PIC "" AS "Bonus 2016"
```

Remarks

For more information about how this command works, see "Import a delimited text file" on page 248.

How Analytics assigns data types when you use ALLFIELDS

When you use the ALLFIELDS parameter, instead of defining fields individually, Analytics examines a subset of records at the beginning of the delimited file and assigns data types to fields based on the set of rules outlined below.

Once the data is in Analytics, if required you can assign different data types, such as Numeric or Datetime, to the fields, and specify format details.

Description of field values in the delimited file	Examples	Data type assigned
Values enclosed by text qualifiers	"ABC Suppliers" "6,990.75"	Character
Values include a non-numeric character anywhere in the field, with the exception of commas and periods used as numeric separators, and the negative sign (-)	$995 (995)	Character
Values include only numbers, numeric separators, or the negative sign	6,990.75 -6,990.75 995	Numeric
One or more blank values occur in a field		Character
Datetime values with separators, or alpha months	2016/12/31 31 Dec 2016	Character
Datetime values that are all numbers	20161231	Numeric

Identifiers for field data types

The table below lists the letters that you must use when specifying *type* for `FIELD`. Each letter corresponds to an Analytics data type.

For example, if you are defining a Last Name field, which requires a character data type, you would specify "C": `FIELD "Last_Name" C`.

For more information, see "Data types in Analytics" on page 739.

> **Note**
>
> When you use the **Data Definition Wizard** to define a table that includes EBCDIC, Unicode, or ASCII fields, the fields are automatically assigned the letter "C" (for the CHARACTER type).
>
> When you enter an IMPORT statement manually, or edit an existing IMPORT statement, you can substitute the more specific letters "E" or "U" for EBCDIC or Unicode fields.

Letter	Analytics Data type
A	ACL
B	BINARY
C	CHARACTER
D	DATETIME
E	EBCDIC
F	FLOAT
G	ACCPAC
I	IBMFLOAT
K	UNSIGNED
L	LOGICAL
N	PRINT
P	PACKED
Q	BASIC
R	MICRO
S	CUSTOM
T	PCASCII
U	UNICODE

Letter	Analytics Data type
V	VAXFLOAT
X	NUMERIC
Y	UNISYS
Z	ZONED

IMPORT EXCEL command

Creates an Analytics table by defining and importing a Microsoft Excel worksheet or named range.

Syntax

```
IMPORT EXCEL TO table import_filename FROM source_filename TABLE input_work-
sheet_or_named_range <KEEPTITLE> <STARTLINE line_number> <ALLCHAR>
{ALLFIELDS|CHARMAX max_field_length|[field_syntax] <...n> <IGNORE field_num>
<...n>} <OPEN>
```

```
field_syntax ::=
FIELD import_name type {PIC format|WID characters DEC value} AS display_name
```

> **Note**
> You must specify the IMPORT EXCEL parameters in exactly the same order as above, and in the table below.
>
> Analytics cannot import from an Excel workbook if Protected View is active for the workbook. You must first enable editing in the workbook, save and close the workbook, and then perform the import.

Parameters

Name	Description
TO *table*	The name of the Analytics table to import the data into. **Note** Table names are limited to 64 alphanumeric characters. The name can include the underscore character (_), but no other special characters, or any spaces. The name cannot start with a number.
import_filename	The name of the Analytics data file to create. Specify *import_filename* as a quoted string with a .FIL file extension. For example, `"Invoices.FIL"`. By default, the data file (.FIL) is saved to the folder containing the Analytics project.

Name	Description
	Use either an absolute or relative file path to save the data file to a different, existing folder: ◦ `"C:\data\Invoices.FIL"` ◦ `"data\Invoices.FIL"`
FROM *source_filename*	The name of the source data file. *source_filename* must be a quoted string. If the source data file is not located in the same directory as the Analytics project, you must use an absolute path or a relative path to specify the file location: ◦ `"C:\data\source_filename"` ◦ `"data\source_filename"`
TABLE *worksheet_or_named_range*	The worksheet or the named range to import from the Microsoft Excel source data file. Requirements: ◦ add a "$" sign at the end of a worksheet name For example, `TABLE "Corp_Credit_Cards$"` ◦ specify a named range exactly as it appears in Excel For example, `TABLE "Employees_Sales"` ◦ specify *worksheet_or_named_range* as a quoted string
KEEPTITLE optional	◦ **KEEPTITLE used with ALLFIELDS or CHARMAX** - Treat the line number specified by STARTLINE as field names instead of data. If you omit KEEPTITLE, generic field names are used and the line number specified by STARTLINE is treated as data. ◦ **KEEPTITLE used with individual FIELD syntax** - Do not import the line number specified by STARTLINE. FIELD *name* specifies the field names. If you omit KEEPTITLE, the line number specified by STARTLINE is treated as data. FIELD *name* specifies the field names.
STARTLINE *line_number* optional	The line number on which to start reading the worksheet. For example, if the first three lines of a worksheet contain header information that you do not want, specify `STARTLINE 4` to start reading data on the fourth line. If you omit STARTLINE, the start line is the first line in the worksheet. **Note** The start line for a named range is always the first line in the named range, regardless of the STARTLINE setting.
ALLCHAR optional	The Character data type is automatically assigned to all the imported fields.

Name	Description
	Tip Assigning the Character data type to all the imported fields simplifies the process of importing Excel files. Once the data is in Analytics, you can assign different data types, such as Numeric or Datetime, to the fields, and specify format details. ALLCHAR is useful if you are importing a table with identifier fields automatically assigned the Numeric data type by Analytics when in fact they should use the Character data type.
ALLFIELDS	All fields in the source data file are imported. **Note** If you specify ALLFIELDS, do not specify any individual FIELD syntax, CHARMAX, or IGNORE.
CHARMAX *max_field_length*	The maximum length in characters for any field in the Analytics table that originates as character data in the source data file. Source character data that exceeds the maximum is truncated. All fields in the source data file, regardless of data type, are imported. **Note** If you specify CHARMAX, do not specify any individual FIELD syntax, ALLFIELDS, or IGNORE.
FIELD *import_name type*	The individual fields to import from the source data file, including the name and data type of the field. *import_name* becomes the field name in the Analytics table. *import_name* does not need to be the same as the field name in the source data file, although it can be. **Tip** You can additionally use AS to specify a display name that is different from *import_name*. *type* becomes the field date type in the Analytics table. *type* does not need to be the same as the field data type in the source data file, although it can be. For more information about *type*, see "Identifiers for field data types" on page 1775. **Note** If you specify ALLCHAR, *type* is ignored. If you specify individual FIELD syntax, do not specify ALLFIELDS or CHARMAX. ## Excluding a field To exclude a field from being imported, do not specify it. You must also specify IGNORE for excluded fields.

Name	Description
PIC *format*	**Note** Applies to numeric or datetime fields only. ◦ **numeric fields** - the display format of numeric values in Analytics views and reports ◦ **datetime fields** - the physical format of datetime values in the source data (order of date and time characters, separators, and so on) **Note** For datetime fields, *format* must exactly match the physical format in the source data. For example, if the source data is 12/31/2014, you must enter the format as `"MM/DD/YYYY"`. *format* must be enclosed in quotation marks.
WID *characters*	The length in characters of the field in the Analytics table layout.
DEC *value*	The number of decimals for numeric fields.
AS *display_name*	The display name (alternate column title) for the field in the view in the new Analytics table. Specify *display_name* as a quoted string. Use a semi-colon (;) between words if you want a line break in the column title. AS is required when you are defining FIELD. To make the display name the same as the field name, enter a blank *display_name* value using the following syntax: `AS ""`. Make sure there is no space between the two double quotation marks.
IGNORE *field_num* <...*n*> optional	Excludes the field from the table layout. *field_num* specifies the position of the excluded field in the source data file. For example, `IGNORE 5` excludes the fifth field in the source data file from the Analytics table layout. **Note** Be careful to correctly align *field_num* with the position of excluded fields. If you specify *field_num* for an included field (FIELD definition), or for a field position that does not exist, the import does not work correctly. The number of FIELD and IGNORE parameters combined must equal the total number of fields in the source data table. If the total numbers do not match, the import does not work correctly. If you specify ALLFIELDS or CHARMAX, do not specify IGNORE.
OPEN optional	Opens the table created by the command after the command executes. Only valid if the command creates an output table.

Examples

Import specified fields

You perform an import that defines a new Analytics table called **Credit_Cards**. It uses the first row of Excel data as the field names.

From the twelve fields in the source table, the Analytics table defines and includes three fields and excludes nine fields:

```
IMPORT EXCEL TO Credit_Cards "Credit Cards.fil" FROM "Credit_Cards_Meta-
phor.xls" TABLE "Corp_Credit_Cards$" KEEPTITLE FIELD "CARDNUM" C WID 16
AS "Card Number" FIELD "EXPDT" D WID 10 PIC "YYYY-MM-DD" AS "Expiry
Date" FIELD "PASTDUEAMT" N WID 6 DEC 2 AS "Past Due" IGNORE 2 IGNORE 3
IGNORE 5 IGNORE 6 IGNORE 7 IGNORE 9 IGNORE 10 IGNORE 11 IGNORE 12
```

Import all fields

You perform an import that defines a new Analytics table called **May_Transactions**. It uses the first row of Excel data as the field names.

The Analytics table includes all fields from the source table and uses default field definitions.

Field length set to longest value

In the first example, fields that originate as character data in the source data file are set to the length of the longest value in the field:

```
IMPORT EXCEL TO May_Transactions "May_Transactions.fil" FROM "Trans_
May.xls" TABLE "Trans1_May$" KEEPTITLE ALLFIELDS
```

Field length constrained

In the second example, fields that originate as character data in the source data file are set to the length of the longest value in the field, or to the CHARMAX value of 50 characters, whichever is shorter:

```
IMPORT EXCEL TO May_Transactions "May_Transactions.fil" FROM "Trans_
May.xls" TABLE "Trans1_May$" KEEPTITLE CHARMAX 50
```

Import all fields as character data

You perform an import that defines a new Analytics table called **May_Transactions**. All fields, including numbers and dates, are imported as character data.

```
IMPORT EXCEL TO May_Transactions "May_Transactions.fil" FROM "Trans_
May.xls" TABLE "Trans1_May$" KEEPTITLE ALLCHAR ALLFIELDS
```

Import all fields as character data, skip header information

You perform an import that defines a new Analytics table called **Past_Due_Report**.

You skip the first two rows of the Excel file, which contain report header information, and start reading the file on the third row, which contains field names. All fields, including numbers and dates, are imported as character data.

```
IMPORT EXCEL TO Past_Due_Report "Past_Due_Report.fil" FROM "Past_Due_
Report.xlsx" TABLE "Sheet1$" KEEPTITLE STARTLINE 3 ALLCHAR ALLFIELDS
```

Remarks

For more information about how this command works, see "Import Microsoft Excel data" on page 234.

Define fields individually, or import all fields using a default definition

When you import an Excel file to an Analytics table you can use FIELD parameters to define each field individually, or you can use the ALLFIELDS parameter, or the CHARMAX parameter, to import all fields using default Analytics field definitions.

Different combinations of parameters produce different results. The table below summarizes the different possibilities.

> **Note**
> "Define" means manually specifying things like field name, data type, length, datetime format, and so on.

I want to:	Use these parameters:	Do not use these parameters:
◦ import all fields automatically with default definitions ◦ if required, define fields post-import in Analytics	ALLFIELDS	CHARMAX, FIELD
◦ import all fields automatically with default definitions ◦ if required, define fields post-import in Analytics ◦ truncate long character fields	CHARMAX	ALLFIELDS, FIELD
◦ define fields pre-import	FIELD	ALLFIELDS, CHARMAX
◦ define fields pre-import ◦ exclude some fields from being imported	FIELD IGNORE	ALLFIELDS, CHARMAX
◦ partially define fields pre-import ◦ automatically import all fields as character data	ALLCHAR FIELD	ALLFIELDS, CHARMAX
◦ omit blank rows or header information at the top of a worksheet	STARTLINE	
◦ use the first row of the worksheet as field names	KEEPTITLE	
◦ use the row of the worksheet specified by STARTLINE as field names	KEEPTITLE STARTLINE	

How Analytics assigns data types when you use ALLFIELDS or CHARMAX

When you use the ALLFIELDS or CHARMAX parameters, instead of defining fields individually, Analytics examines a subset of records at the beginning of the Excel file and assigns data types to fields based on a set of internal rules.

Once the data is in Analytics, if required you can assign different data types, such as Numeric or Datetime, to the fields, and specify format details.

Maximum size of data import

File format .xlsx or .xlsm

The maximum number of Excel columns, and the maximum number of characters in a field, that you can import from .xlsx or .xlsm files is not limited to a specific number.

Importing from these Excel file types is governed by the record length limit in Analytics data files (.fil) of 32 KB. If any record in the source Excel file would create an Analytics record longer than 32 KB, the import fails.

File format .xls

The import of .xls (Excel 97 - 2003) files uses a different type of processing, and is subject to maximums of:

- 255 columns
- 255 characters per field
- 32 KB per record
- 65,000 rows

Identifiers for field data types

The table below lists the letters that you must use when specifying *type* for `FIELD`. Each letter corresponds to an Analytics data type.

For example, if you are defining a Last Name field, which requires a character data type, you would specify "C": `FIELD "Last_Name" C`.

For more information, see "Data types in Analytics" on page 739.

> **Note**
>
> When you use the **Data Definition Wizard** to define a table that includes EBCDIC, Unicode, or ASCII fields, the fields are automatically assigned the letter "C" (for the CHARACTER type).
>
> When you enter an IMPORT statement manually, or edit an existing IMPORT statement, you can substitute the more specific letters "E" or "U" for EBCDIC or Unicode fields.

Letter	Analytics Data type
A	ACL
B	BINARY
C	CHARACTER

Letter	Analytics Data type
D	DATETIME
E	EBCDIC
F	FLOAT
G	ACCPAC
I	IBMFLOAT
K	UNSIGNED
L	LOGICAL
N	PRINT
P	PACKED
Q	BASIC
R	MICRO
S	CUSTOM
T	PCASCII
U	UNICODE
V	VAXFLOAT
X	NUMERIC
Y	UNISYS
Z	ZONED

IMPORT GRCPROJECT command

Creates an Analytics table by importing a HighBond Projects table.

Syntax

```
IMPORT GRCPROJECT TO table import_filename PASSWORD num FROM org_id/type_id
<FIELD name AS display_name <...n>>
```

Parameters

Name	Description
TO *table*	The name of the Analytics table to import the data into. **Note** Table names are limited to 64 alphanumeric characters. The name can include the underscore character (_), but no other special characters, or any spaces. The name cannot start with a number.
import_filename	The name of the Analytics data file to create. Specify *import_filename* as a quoted string with a .FIL file extension. For example, `"Invoices.FIL"`. By default, the data file (.FIL) is saved to the folder containing the Analytics project. Use either an absolute or relative file path to save the data file to a different, existing folder: ◦ `"C:\data\Invoices.FIL"` ◦ `"data\Invoices.FIL"`
PASSWORD *num*	The password definition to use. You do not use PASSWORD *num* to prompt for, or specify, an actual password. The password definition refers to a password previously supplied or set using the PASSWORD command, the SET PASSWORD command, or the PASSWORD analytic tag. *num* is the number of the password definition. For example, if two passwords have been previously supplied or set in a script, or when scheduling an analytic, `PASSWORD 2` specifies that password #2 is used. For more information about supplying or setting passwords, see:

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>

"PASSWORD command" on page 1893
"SET command" on page 1960
"PASSWORD tag" on page 2530

The required password value is a HighBond access token. For more information, see "Creating a password definition and specifying a password value" on page 1781.

Note
PASSWORD may or may not be required, depending on the environment in which the script runs:
<table>
<tr><td>Analytics
(online activation)</td><td>PASSWORD is not required.
The current user's HighBond access token is automatically used.</td></tr>
<tr><td>Analytics
(offline activation)</td><td rowspan="4">PASSWORD is required.</td></tr>
<tr><td>Robots</td></tr>
<tr><td>Analytics Exchange</td></tr>
<tr><td>Analysis App window</td></tr>
</table>
</td></tr>
<tr><td>FROM org_id/type_id</td><td>
The organization and type of information that defines the data being imported:

org_id - the Projects organization you are importing data from
type_id - the type of information you are importing

The org_id value and the type_id value must be separated by a slash, with no intervening spaces: <code>FROM "125@eu/audits"</code>.

The entire string must be enclosed in quotation marks.
<h2>Organization ID</h2>
org_id must include the organization ID number, and if you are importing from a data center other than North America (US), the data center code. The organization ID number and the data center code must be separated by the at sign (@): <code>FROM "125@eu"</code>.

The data center code specifies which regional HighBond server you are importing the data from.

<code>af</code> - Africa (South Africa)
<code>ap</code> - Asia Pacific (Singapore)
<code>au</code> - Asia Pacific (Australia)
<code>ca</code> - North America (Canada)
<code>eu</code> - Europe (Germany)
<code>sa</code> - South America (Brazil)
<code>us</code> - North America (US)

You can use only the data center code or codes authorized for your organization's
</td></tr>
</table>

Name	Description
	instance of HighBond. The North America (US) data center is the default, so specifying `@us` is optional. If you do not know the organization ID number, use the Analytics user interface to import a table from Projects. The organization ID number is contained in the command in the log. For more information, see "Import HighBond Projects data" on page 688. ## Type ID *type_id* specifies the type of information you are importing. Information in Projects is contained in a series of related tables. For *type_id*, use one of the values listed below. Enter the value exactly as it appears and include underscores, if applicable: ◦ `audits` - Projects ◦ `control_test_plans` - Control Test Plans ◦ `control_tests` - Control Test ◦ `controls` - Controls ◦ `finding_actions` - Actions ◦ `findings` - Issues ◦ `mitigations` - Risk Control Associations ◦ `narratives` - Narratives ◦ `objectives`- Objectives ◦ `risks` - Risks ◦ `walkthroughs` - Walkthroughs **Tip** For information about how the tables in Projects are related, and the key fields that you can use to join the tables once you have imported them to Analytics, see "Import HighBond Projects data" on page 688.
FIELD *name* AS *display_name* <...*n*> optional	Individual fields in the source data to import. Specify the name. If you omit FIELD, all fields are imported. ◦ *name* must exactly match the physical field name in the Projects table, including matching the case ◦ *display_name* (alternate column title) is the display name for the field in the view in the new Analytics table. You must specify a display name for each FIELD *name*. Specify *display_name* as a quoted string. Use a semi-colon (;) between words if you want a line break in the column title. Unlike some other IMPORT commands in Analytics, you cannot specify a blank *display_name* as a way of using the FIELD name as the display name. **Tip** To get the physical field names, use the Analytics user interface to import the appropriate table from Projects. The physical field names are contained in the command in the log. Subsequent imports can be scripted.

Examples

Importing all fields from the Projects table

You import all fields from the **Projects** table for all active projects belonging to organization 286 to an Analytics table named **All_Projects**. You include a numbered password definition to authenticate the connection:

```
IMPORT GRCPROJECT TO All_Projects "C:\HighBond Projects Data\All_Pro-
jects.fil" PASSWORD 1 FROM "286@us/audits"
```

Importing specified fields from the Projects table

You import specified fields from the **Projects** table for all active projects belonging to organization 286 to an Analytics table named **All_Projects**:

```
IMPORT GRCPROJECT TO All_Projects "C:\HighBond Projects Data\All_Pro-
jects.fil" FROM "286@us/audits" FIELD "id" AS "Id" FIELD "description"
AS "Description" FIELD "name" AS "Name" FIELD "start_date" AS "Start
date" FIELD "status" AS "Status" FIELD "created_at" AS "Created at"
```

Importing all fields from the Issues table

You import all fields from the **Issues** table for all active projects belonging to organization 286 to an Analytics table named **All_Issues**:

```
IMPORT GRCPROJECT TO All_Issues "C:\HighBond Projects Data\All_Issues.-
fil" FROM "286@us/findings"
```

Remarks

For more information about how this command works, see "Import HighBond Projects data" on page 688.

Creating a password definition and specifying a password value

PASSWORD command

If you use the PASSWORD command to create the numbered password definition for connecting to HighBond, no password value is specified, so a password prompt is displayed when the script attempts to connect.

For more information, see "PASSWORD command" on page 1893.

SET PASSWORD command

If you use the SET PASSWORD command to create the numbered password definition for connecting to HighBond, a password value is specified, so no password prompt is displayed, which is appropriate for scripts designed to run unattended.

For more information, see SET PASSWORD command.

Acquire a HighBond access token

Regardless of which method you use to create the password definition, the required password value is a HighBond access token, which users can generate in Launchpad.

> **Caution**
> The generated access token matches the account used to sign in to Launchpad. As a scriptwriter, specifying your own access token in a script may not be appropriate if the script will be used by other people.

1. Do one of the following:
   - From the Analytics main menu, select **Tools > HighBond Access Token**.
   - In the **Script Editor**, right-click and select **Insert > HighBond Token**.

   The **Manage API tokens** page opens in your browser. You may be required to first sign in to Launchpad.
2. Do one of the following:
   - **Use an existing token** – In the **Token** column, click the partially masked token that you want to use and enter your HighBond account password. The unmasked token is displayed.

> **Tip**
> Use an existing token unless you have a reason for creating a new one. If the existing token does not work, create a new one.
> Using an existing token cuts down on the number of tokens you need to manage.

- **Create a new token** – Click **Create token > Analytics** and enter your HighBond account password.

  A new Analytics token is created.

> **Note**
> If you are a Launchpad System Admin, you also have the option of creating an API token. You should reserve API tokens for their intended purpose, which is programmatic access to the HighBond platform.

3. Click **Copy** to copy the token.

> **Tip**
> Do not close the dialog box containing the token until you have successfully pasted the token.

4. In Analytics, do one of the following:
   - paste the token into the password prompt
   - paste the token at the appropriate point in the SET PASSWORD command syntax in a script
5. In Launchpad, close the dialog box containing the token.

   If you created a new token, a partially masked version of the token is added to the top of your list of tokens.

   For more information, see Creating and managing access tokens.

Import debug capability

A simple debug capability exists for imports from HighBond.

The imported data is temporarily stored in a JSON intermediary file in the folder containing the target Analytics project. In any folder containing an Analytics project you can create a text file that causes the JSON file to be retained, instead of being deleting after the data is imported into Analytics.

- **JSON file is present** – If the import from HighBond is failing, but the JSON file is present on your computer, you know that the problem is on the Analytics side, not on the HighBond side.
- **JSON file is not present** – If the import from HighBond is failing, and the JSON file is not present on your computer, you know that the problem is on the HighBond side.

This information can help with troubleshooting.

Configure retention of the JSON intermediary file

In the folder containing the target Analytics project, create an empty text file with exactly this name: `_grc_import_debug.txt`

When you import from either Results or Projects in HighBond, the JSON intermediary file is retained with the name `results.json`. The file is overwritten with each subsequent import from HighBond.

IMPORT GRCRESULTS command

Creates an Analytics table by importing a HighBond Results table or interpretation.

Syntax

```
IMPORT GRCRESULTS TO table import_filename PASSWORD num FROM Results_resource_
path <FIELD name AS display_name <...n>>
```

Parameters

Name	Description
TO *table*	The name of the Analytics table to import the data into. **Note** Table names are limited to 64 alphanumeric characters. The name can include the underscore character (_), but no other special characters, or any spaces. The name cannot start with a number.
import_filename	The name of the Analytics data file to create. Specify *import_filename* as a quoted string with a .FIL file extension. For example, `"Invoices.FIL"`. By default, the data file (.FIL) is saved to the folder containing the Analytics project. Use either an absolute or relative file path to save the data file to a different, existing folder: ◦ `"C:\data\Invoices.FIL"` ◦ `"data\Invoices.FIL"`
PASSWORD *num*	The password definition to use. You do not use PASSWORD *num* to prompt for, or specify, an actual password. The password definition refers to a password previously supplied or set using the PASSWORD command, the SET PASSWORD command, or the PASSWORD analytic tag. *num* is the number of the password definition. For example, if two passwords have been previously supplied or set in a script, or when scheduling an analytic, `PASSWORD 2` specifies that password #2 is used. For more information about supplying or setting passwords, see:

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>

"PASSWORD command" on page 1893
"SET command" on page 1960
"PASSWORD tag" on page 2530

The required password value is a HighBond access token. For more information, see "Creating a password definition and specifying a password value" on page 1789.

Note
PASSWORD may or may not be required, depending on the environment in which the script runs:
<table>
<tr><td>Analytics
(online activation)</td><td>PASSWORD is not required.
The current user's HighBond access token is automatically used.</td></tr>
<tr><td>Analytics
(offline activation)</td><td rowspan="4">PASSWORD is required.</td></tr>
<tr><td>Robots</td></tr>
<tr><td>Analytics Exchange</td></tr>
<tr><td>Analysis App window</td></tr>
</table>
</td></tr>
<tr><td>FROM Results_resource_path</td><td>
The path to the data you are importing.

The form of the path varies depending on the data you are importing. For details about the form of the path, see "Results path" on page 1787.

Note
The form of the Results path is supplied by an API, and is subject to change. The easiest and most reliable way to acquire the correct and current syntax for the path is to perform a manual import of the target data, and copy the path from the command log.
</td></tr>
<tr><td>FIELD name AS display_name <...n>

optional</td><td>
Individual fields in the source data to import. Specify the name.

If you omit FIELD, all fields are imported.

<h2>Name</h2>
name must exactly match the physical field name in the Results table, including matching the case. To view the physical field name, do one of the following:

In Results, click a column header in the Table View. The physical field name appears after Field Name.
In Analytics, when you import a Results table, the physical field name appears in parentheses after the display name in the dialog box that allows you to select fields.

</td></tr>
</table>

Name	Description
	Note The Results physical field name is not the display name used for column headers in the **Table View**. Also see "Field name considerations when importing and exporting Results data" on page 1789. ### Display name *display_name* (alternate column title) is the display name for the field in the view in the new Analytics table. You must specify a display name for each FIELD *name*. Specify *display_name* as a quoted string. Use a semi-colon (;) between words if you want a line break in the column title. Unlike some other IMPORT commands in Analytics, you cannot specify a blank *display_name* as a way of using the FIELD name as the display name.

Examples

Importing specified fields from a table in Results

You import specified fields from a table in Results to an Analytics table named **T and E exceptions**:

```
IMPORT GRCRESULTS TO T_and_E_exceptions "C:\Secondary Analysis\T_and_E_
exceptions.fil" PASSWORD 1 FROM "results/api/orgs/11594/control_test-
s/185699/exceptions" FIELD "metadata.status" AS "Status" FIELD "EmpNo"
AS "Employee Number" FIELD "DATE" AS "Date" FIELD "CARDNUM" AS "Card Num-
ber" FIELD "CODES" AS "MC Codes" FIELD "AMOUNT" AS "Amount" FIELD
"DESCRIPTION" AS "Description"
```

Importing all fields from a table in Results

You import all fields from a table in Results to an Analytics table named **T and E exceptions**:

```
IMPORT GRCRESULTS TO T_and_E_exceptions "C:\Secondary Analysis\T_and_E_
exceptions.fil" PASSWORD 1 FROM "results/api/orgs/11594/control_test-
s/185699/exceptions"
```

Importing data from an interpretation in Results

You import an interpretation in Results to an Analytics table named **T and E exceptions filtered**:

```
IMPORT GRCRESULTS TO T_and_E_exceptions_filtered "C:\Secondary Ana-
lysis\T_and_E_exceptions_filtered.fil" FROM "res-
ults/api/orgs/11594/control_
tests/185699/interpretations/22699/exceptions"
```

Remarks

For more information about how this command works, see "Import HighBond Results data" on page 692.

Preserving sort order and filters

When you import data from Results, any data customization, such as sorting or filter, is retained or ignored in the resulting Analytics table depending on how you import the data:

- **import a table** - data customization is ignored. All the data in the table is imported, with the exception of any fields you choose to omit.
- **import an interpretation** - data customization is retained

Results path

> **Note**
> The form of the Results path is supplied by an API, and is subject to change. The easiest and most reliable way to acquire the correct and current syntax for the path is to perform a manual import of the target data, and copy the path from the command log.

The Results path in the `FROM` parameter takes the following general form:

`FROM "results <-region code>/api/orgs/<org ID>/control_tests/<control test ID>/exceptions`

For example: `FROM "results/api/orgs/11594/control_tests/4356/exceptions"`

The org ID is displayed in the browser address bar when you log in to Launchpad. The control test ID, and the interpretation ID, are displayed in the address bar when you are viewing those tables in Results.

The table below provides all the variations of the Results path.

To import:	Use this form of Results path:
Control test (table) data	`FROM "results/api/orgs/11594/control_tests/4356/exceptions"`
Control test (table) audit trail	`FROM "results/api/orgs/11594/control_tests/4356/audit_trail"`
Control test (table) comments	`FROM "results/api/orgs/11594/control_tests/4356/comments"`
Interpretation	`FROM "results/api/orgs/11594/control_tests/4356/interpretations/1192/exceptions"`
Data from a HighBond region other than the default region (us)	◦ **Asia Pacific** - `FROM "results-ap/api/orgs/11594/control_tests/4356/exceptions"` ◦ **Australia** - `FROM "results-au/api/orgs/11594/control_tests/4356/exceptions"` ◦ **Canada** - `FROM "results-ca/api/orgs/11594/control_tests/4356/exceptions"` ◦ **Europe** - `FROM "results-eu/api/orgs/11594/control_tests/4356/exceptions"`

System-generated information columns

When you import data from Results, you have the option of also importing one or more of the system-generated information columns listed below.

The system-generated columns are either:

- part of Results tables, and contain processing information related to individual records
- additional information – collection name, table name, or record ID number

You must specify the field names of the system-generated columns exactly as they appear below. The default display names apply when you import from Results through the Analytics user interface. You are free to change the display names if you are scripting the import process.

Field name	Default display name
metadata.priority	Priority
metadata.status	Status
metadata.publish_date	Published

Field name	Default display name
metadata.publisher	Publisher Name
metadata.assignee	Assignee
metadata.group	Group
metadata.updated_at	Updated
metadata.closed_at	Closed
extras.collection	Collection
extras.results_table	Results Table
extras.record_id	Record ID

Field name considerations when importing and exporting Results data

If you are round-tripping data between Results and Analytics, you need to ensure that all field names in the Results table meet the more stringent Analytics field name requirements. If you do not, you risk misaligning your Analytics and Results data.

For example, any special characters in Results field names are automatically converted to underscores when they are imported into Analytics, which means the field names no longer match the original names in Results. If you then export the Analytics data back to the original table in Results, fields are no longer correctly matched.

To avoid this problem with data that you intend to round-trip, make sure that before you upload the data to Results from CSV or Excel files it meets these Analytics field name requirements:

- no special characters or spaces
- does not start with a number
- contains only alphanumeric characters, or the underscore character (_)

Creating a password definition and specifying a password value

PASSWORD command

If you use the PASSWORD command to create the numbered password definition for connecting to HighBond, no password value is specified, so a password prompt is displayed when the script attempts to connect.

For more information, see "PASSWORD command" on page 1893.

SET PASSWORD command

If you use the SET PASSWORD command to create the numbered password definition for connecting to HighBond, a password value is specified, so no password prompt is displayed, which is appropriate for scripts designed to run unattended.

For more information, see SET PASSWORD command.

Acquire a HighBond access token

Regardless of which method you use to create the password definition, the required password value is a HighBond access token, which users can generate in Launchpad.

> **Caution**
> The generated access token matches the account used to sign in to Launchpad. As a scriptwriter, specifying your own access token in a script may not be appropriate if the script will be used by other people.

1. Do one of the following:
   - From the Analytics main menu, select **Tools > HighBond Access Token**.
   - In the **Script Editor**, right-click and select **Insert > HighBond Token**.

   The **Manage API tokens** page opens in your browser. You may be required to first sign in to Launchpad.
2. Do one of the following:
   - **Use an existing token** – In the **Token** column, click the partially masked token that you want to use and enter your HighBond account password. The unmasked token is displayed.

     > **Tip**
     > Use an existing token unless you have a reason for creating a new one. If the existing token does not work, create a new one.
     >
     > Using an existing token cuts down on the number of tokens you need to manage.

   - **Create a new token** – Click **Create token > Analytics** and enter your HighBond account password.

     A new Analytics token is created.

     > **Note**
     > If you are a Launchpad System Admin, you also have the option of creating an API token. You should reserve API tokens for their intended purpose, which is programmatic access to the HighBond platform.

3. Click **Copy** to copy the token.

   > **Tip**
   > Do not close the dialog box containing the token until you have successfully pasted the token.

4. In Analytics, do one of the following:
   - paste the token into the password prompt
   - paste the token at the appropriate point in the SET PASSWORD command syntax in a script
5. In Launchpad, close the dialog box containing the token.

   If you created a new token, a partially masked version of the token is added to the top of your list of tokens.

   For more information, see Creating and managing access tokens.

Import debug capability

A simple debug capability exists for imports from HighBond.

The imported data is temporarily stored in a JSON intermediary file in the folder containing the target Analytics project. In any folder containing an Analytics project you can create a text file that causes the JSON file to be retained, instead of being deleting after the data is imported into Analytics.

- **JSON file is present** – If the import from HighBond is failing, but the JSON file is present on your computer, you know that the problem is on the Analytics side, not on the HighBond side.
- **JSON file is not present** – If the import from HighBond is failing, and the JSON file is not present on your computer, you know that the problem is on the HighBond side.

This information can help with troubleshooting.

Configure retention of the JSON intermediary file

In the folder containing the target Analytics project, create an empty text file with exactly this name: `_grc_import_debug.txt`

When you import from either Results or Projects in HighBond, the JSON intermediary file is retained with the name `results.json`. The file is overwritten with each subsequent import from HighBond.

Importing large tables

Tables that have a large number of fields may not successfully import using a single IMPORT GRCRESULTS command. If you need to work with a single table containing a large number of fields outside of Results, use one of the following approaches:

- **Split the table** – use two or more IMPORT GRCRESULTS commands to import a subset of fields and then join the resulting tables in Analytics using the JOIN command
- **Export the table to file** – use the export to CSV format and then import the resulting file into Analytics using the IMPORT DELIMITED command

IMPORT LAYOUT command

Imports an external table layout file (.layout) to an Analytics project.

> **Note**
> Prior to version 11 of Analytics, external table layout files used an .fmt file extension. You can still import a table layout file with an .fmt extension by manually specifying the extension.

Syntax

```
IMPORT LAYOUT external_layout_file TO table_layout_name
```

Parameters

Name	Description
external_layout_file	The name of the external table layout file. If the file name or path includes any spaces it must be enclosed in quotation marks – for example, `"Ap Trans.layout"`. The .layout file extension is used by default and does not need to be specified. If required, you can use another file extension, such as .fmt. If the layout file is not located in the same folder as the Analytics project, you must use an absolute path or a relative path to specify the file location – for example, `"C:\Saved layouts\Ap_Trans.layout"` or `"Saved layouts\Ap_Trans.layout"`.
TO *table_layout_name*	The name of the imported table layout in the Analytics project – for example, `"Ap Trans May"`. You must specify the *table_layout_name* as a quoted string if it contains any spaces. You can specify a *table_layout_name* that is different from the name of the *external_layout_file*.

Example

Importing an external table layout file

You import an external table layout file called **Ap_Trans.layout** and create a new table layout called **Ap_Trans _May** in the Analytics project:

```
IMPORT LAYOUT "C:\Saved layouts\Ap_Trans.layout" TO "Ap_Trans_May"
```

Remarks

When to use IMPORT LAYOUT

Importing an external table layout file and associating it with a data file can save you the labor of creating a new table layout from scratch:

- If the imported table layout specifies an association with a particular Analytics data file (.fil), and a data file of the same name exists in the folder containing the project, the imported table layout is automatically associated with the data file in the folder.
- If there is no data file with the same name in the project folder, you need to link the imported table layout to a new data source.

Table layouts and source data files must match

The imported table layout and the data file it is associated with must match. The structure of the data in the data file must match the field definitions specified by the table layout metadata.

Data structure refers to the data elements (fields) contained in a data file, the number and order of the fields, and the data type and length of the fields. If the table layout and the data file do not match, jumbled or missing data results.

IMPORT MULTIDELIMITED command

Creates multiple Analytics tables by defining and importing multiple delimited files.

Syntax

```
IMPORT MULTIDELIMITED <TO import_folder> FROM {source_filename|source_folder}
source_char_encoding SEPARATOR {char|TAB|SPACE} QUALIFIER {char|NONE}
<CONSECUTIVE> STARTLINE line_number <KEEPTITLE> <CRCLEAR> <LFCLEAR>
<REPLACENULL> <ALLCHAR>
```

> **Note**
>
> You must specify the IMPORT MULTIDELIMITED parameters in exactly the same order as above, and in the table below.
>
> To import multiple delimited files cleanly, the structure of all the files must be consistent before importing.
>
> For more information, see "Consistent file structure required" on page 1800.

Parameters

Name	Description
TO *import_folder* optional	The folder to import the data into. To specify the folder, use an absolute file path, or a file path relative to the folder containing the Analytics project. Specify *import_folder* as a quoted string. **Example** `TO "C:\Point of sale audit\Data\Transaction working data"`

Name	Description
	``` TO "Data\Transaction working data" ``` If you omit `TO`, the data is imported to the folder containing the Analytics project.

FROM *source_filename* | *source_folder*	The name of the source data files, or the folder containing the source data files.
	Specify *source_filename* or *source_folder* as a quoted string.
	The command supports importing four types of delimited file:
	◦ `*.csv`
	◦ `*.dat`
	◦ `*.del`
	◦ `*.txt`
	Source data files in the root Analytics project folder
	To specify multiple files, use a wildcard character (*) in place of unique characters in file names. The wildcard character stands for zero (0) or more occurrences of any letter, number, or special character.
	Example
	``
	FROM "Transactions_FY*.csv"
	``
	selects:
	`Transactions_FY18.csv`
	`Transactions_FY17.csv`
	You can use a wildcard in more than one location in a file name, and in a file extension.
	Example
	``
	FROM "Transactions_FY*.*"
	``
	selects:
	`Transactions_FY18.txt`
	`Transactions_FY17.csv`
	Source data files not in the root Analytics project folder

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>If the source data files are not located in the same folder as the Analytics project, you must use an absolute file path, or a file path relative to the folder containing the project, to specify the location of the files.

Example

<pre>FROM "C:\Point of sale audit\Data\Transaction master files\Transactions_FY*.csv"</pre>
<pre>FROM "Data\Transaction master files\Transactions_FY*.csv"</pre>
Folder containing source data files

Instead of specifying file names, you can just specify the name of the folder containing source data files. All supported delimited files in the folder are imported (<code>*.csv</code>, <code>*.dat</code>, <code>*.del</code>, <code>*.txt</code>).

To specify a source data folder, use an absolute file path, or a file path relative to the folder containing the Analytics project.

Example

<pre>FROM "C:\Point of sale audit\Data\Transaction master files"</pre>
<pre>FROM "Data\Transaction master files"</pre></td></tr>
<tr><td>source_char_encoding</td><td>The character set and encoding of the source data.

Depending on which edition of Analytics you are using, and the encoding of the source data, specify the appropriate code:

<table><tr><th>Code</th><th>Analytics edition</th><th>Source data encoding</th></tr><tr><td><code>0</code></td><td>Non-Unicode edition</td><td>all data</td></tr><tr><td><code>0</code></td><td>Unicode edition</td><td>ASCII data</td></tr></table></td></tr>
</table>

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Code</th><th>Analytics edition</th><th>Source data encoding</th></tr>
<tr><td><code>2</code></td><td>Unicode edition</td><td>Unicode data, UTF-16 LE encoding</td></tr>
<tr><td><code>3 numeric_code</code></td><td>Unicode edition</td><td>Unicode data that does not use UTF-16 LE encoding
To determine the numeric code that matches the source data encoding, perform an import using the Data Definition Wizard, select the Encoded Text option, and find the matching encoding in the accompanying drop-down list.
To specify the code, specify <code>3</code>, followed by a space, and then the numeric code.</td></tr>
</table>
Note
If you do not specify a code, Non-Unicode Analytics automatically uses <code>0</code>, and Unicode Analytics automatically uses <code>2</code>.
</td></tr>
<tr><td>SEPARATOR char | TAB | SPACE</td><td>The separator character (delimiter) used between fields in the source data. You must specify the character as a quoted string.
You can specify a tab or a space separator by typing the character between double quotation marks, or by using a keyword:
◦ <code>SEPARATOR " "</code> or <code>SEPARATOR TAB</code>
◦ <code>SEPARATOR " "</code> or <code>SEPARATOR SPACE</code></td></tr>
<tr><td>QUALIFIER char | NONE</td><td>The text qualifier character used in the source data to wrap and identify field values. You must specify the character as a quoted string.
To specify the double quotation mark character as the text qualifier, enclose the character in single quotation marks: <code>QUALIFIER '"'</code>.
You can specify that there are no text qualifiers using either of these methods:
◦ <code>QUALIFIER ""</code>
◦ <code>QUALIFIER NONE</code></td></tr>
<tr><td>CONSECUTIV-E
optional</td><td>Consecutive text qualifiers are treated as a single qualifier.</td></tr>
<tr><td>STARTLINE line_number</td><td>The line the data begins on.
For example, if the first four lines of data contain header information that you do not want, specify <code>5</code> for line_number.
Note
Ideally, the start line of the data should be the same in all the delimited files that you import with a single execution of IMPORT MULTIDELIMITED.
If start lines are different, see "Consistent file structure required" on page 1800.</td></tr>
</table>

Name	Description
KEEPTITLE optional	Treat the line number specified by STARTLINE as field names instead of data. If you omit KEEPTITLE, generic field names are used. **Note** The field names must be on the same line number in all the delimited files that you import with a single execution of IMPORT MULTIDELIMITED. If field names are on different line numbers, see "Consistent file structure required" on page 1800.
CRCLEAR optional	Replaces any CR characters (carriage return) that occur between text qualifiers with space characters. You must specify QUALIFIER with a *char* value to use CRCLEAR. If you use both CRCLEAR and LFCLEAR, CRCLEAR must come first.
LFCLEAR optional	Replaces any LF characters (line feed) that occur between text qualifiers with space characters. You must specify QUALIFIER with a *char* value to use LFCLEAR. If you use both CRCLEAR and LFCLEAR, CRCLEAR must come first.
REPLACENUL-L optional	Replaces any NUL characters that occur in the delimited file with space characters. The number of any replaced NUL characters is recorded in the log.
ALLCHAR optional	The Character data type is automatically assigned to all the imported fields. **Tip** Assigning the Character data type to all the imported fields simplifies the process of importing delimited text files. Once the data is in Analytics, you can assign different data types, such as Numeric or Datetime, to the fields, and specify format details. ALLCHAR is useful if you are importing a table with identifier fields automatically assigned the Numeric data type by Analytics when in fact they should use the Character data type.

Examples

The examples below assume that you have monthly transaction data stored in 12 delimited files:

- `Transactions_Jan.csv` to `Transactions_Dec.csv`

Note
A separate Analytics table is created for each delimited file that you import.

Import all the delimited files

You want to import all 12 delimited files. You use the wildcard symbol (*) where the month occurs in each file name.

Analytics attempts to assign the appropriate data type to each field.

```
IMPORT MULTIDELIMITED FROM "Transactions_*.csv" 0 SEPARATOR ","
QUALIFIER '"' CONSECUTIVE STARTLINE 1 KEEPTITLE
```

Import all the delimited files as character data

This example is the same as the one above, except Analytics automatically assigns the Character data type to all the imported fields.

```
IMPORT MULTIDELIMITED FROM "Transactions_*.csv" 0 SEPARATOR ","
QUALIFIER '"' CONSECUTIVE STARTLINE 1 KEEPTITLE ALLCHAR
```

Import all the delimited files from the specified folder

You want to import all the delimited files in the `C:\Point of sale audit\Data\Transaction master files` folder.

```
IMPORT MULTIDELIMITED FROM "C:\Point of sale audit\Data\Transaction mas-
ter files" 0 SEPARATOR "," QUALIFIER '"' CONSECUTIVE STARTLINE 1
KEEPTITLE
```

Import all the delimited files from the specified folder, and save the Analytics tables to another folder

This example is the same as the one above, but instead of saving the Analytics tables in the root project folder, you want to save them in the `C:\Point of sale audit\Data\Transaction working data` folder.

```
IMPORT MULTIDELIMITED TO "C:\Point of sale audit\Data\Transaction work-
ing data" FROM "C:\Point of sale audit\Data\Transaction master files" 0
SEPARATOR "," QUALIFIER '"' CONSECUTIVE STARTLINE 1 KEEPTITLE
```

Remarks

Consistent file structure required

To import a group of delimited files cleanly using IMPORT MULTIDELIMITED, the structure of all the files in the group must be consistent.

You can import inconsistently structured delimited files, and subsequently perform data cleansing and standardizing in Analytics. However, this approach can be labor intensive. In many cases, it is easier to make the delimited files consistent before importing.

To import multiple delimited files cleanly, the following items need to be consistent across all files:

Item	ACLScript keyword	Problem	Solution
The character set and encoding of the source data	*numeric code*	(Unicode edition of Analytics only) Source delimited files use different character encodings. For example, some files have ASCII encoding and some files have Unicode encoding.	Group source files by encoding type, and do a separate import for each group.
Delimiter character	SEPARATOR	Source delimited files use a different separator character (delimiter) between fields.	Do one of the following: ◦ Standardize the separator character in the source files before importing them. ◦ Group source files by separator character, and do a separate import for each group.
Text qualifier character	QUALIFIER	Source delimited files use a different text qualifier character to wrap and identify field values.	Do one of the following: ◦ Standardize the qualifier character in the source files before importing them. ◦ Group source files by qualifier character, and do a separate import for each group.
Start line of the data	STARTLINE	Source delimited files have different start lines for the data.	Do one of the following: ◦ Standardize the start line in the source files before importing

Item	ACLScript keyword	Problem	Solution
			them. ◦ Group source files that have the same start line, and do a separate import for each group. ◦ Make *line_number* equal to the lowest start line across all the files. Once the files have been imported to Analytics tables, you can use the "EXTRACT command" on page 1712 to extract only the records from any table with unwanted header information.
Field names	KEEPTITLE	Source delimited files have field names on different line numbers.	Do one of the following: ◦ Standardize the line number with the field names in the source files before importing them. ◦ Group source files that have field names on the same line number, and do a separate import for each group.
Field names	KEEPTITLE	Some source delimited files have field names and some do not.	Do one of the following: ◦ Add field names to the source files that require them before importing all files. ◦ Group source files that have field names, and files that do not have field names, and do a separate import for each group. ◦ Omit KEEPTITLE to import all files using generic field names. Once the files have been imported to Analytics tables, you can use the "EXTRACT command" on page 1712 to extract only the data you want from any table.

Multiple IMPORT DELIMITED commands

The IMPORT MULTIDELIMITED command actually performs multiple individual IMPORT DELIMITED commands – one for each file imported. If you double-click the IMPORT MULTIDELIMITED entry in the log, the individual IMPORT DELIMITED commands are displayed in the display area.

Combining multiple delimited files after importing them

After you import multiple delimited files into individual Analytics tables you might want to combine them into a single Analytics table. For example, you could combine the data from twelve monthly tables into a single annual table containing all the data.

For information about combining multiple Analytics tables, see "APPEND command" on page 1565.

IMPORT MULTIEXCEL command

Creates multiple Analytics tables by defining and importing multiple Microsoft Excel worksheets or named ranges.

Syntax

```
IMPORT MULTIEXCEL <TO import_folder> FROM {source_filename|source_folder}
TABLE input_worksheets_or_named_ranges <PREFIX> <KEEPTITLE> <CHARMAX max_
field_length>
```

> **Note**
>
> You must specify the IMPORT MULTIEXCEL parameters in exactly the same order as above, and in the table below.
>
> Analytics cannot import from an Excel workbook if Protected View is active for the workbook. You must first enable editing in the workbook, save and close the workbook, and then perform the import.

Parameters

Name	Description

TO *import_folder*
optional
	The folder to import the data into.
	To specify the folder, use an absolute file path, or a file path relative to the folder containing the Analytics project. Specify *import_folder* as a quoted string.
	Example
	`TO "C:\Point of sale audit\Data\Transaction working data"`
	`TO "Data\Transaction working data"`
	If you omit TO, the data is imported to the folder containing the Analytics project.

Name	Description

FROM *source_filename* | *source_folder*	The name of the source data file or files, or the folder containing the source data file or files.
	Specify *source_filename* or *source_folder* as a quoted string.
	Source data file or files in the root Analytics project folder
	◦ **single Excel file**
	Specify the compete file name and extension.
	Example
	`FROM "Transactions_FY18.xlsx"`
	◦ **multiple Excel files**
	To specify multiple files, use a wildcard character (*) in place of unique characters in file names. The wildcard character stands for zero (0) or more occurrences of any letter, number, or special character.
	Example
	`FROM "Transactions_FY*.xlsx"`
	selects:
	`Transactions_FY18.xlsx`
	`Transactions_FY17.xlsx`
	You can use a wildcard in more than one location in a file name, and in a file extension.
	Example
	`FROM "Transactions_FY*.*"`
	selects:
	`Transactions_FY18.xlsx`

Name	Description

	`Transactions_FY17.xls`
	
	### Source data file or files not in the root Analytics project folder
	
	If the source data file or files are not located in the same folder as the Analytics project, you must use an absolute file path, or a file path relative to the folder containing the project, to specify the file location.
	
	#### Example
	
	`FROM "C:\Point of sale audit\Data\Transaction master files\Transactions_FY18.xlsx"`
	
	`FROM "Data\Transaction master files\Transactions_FY*.xlsx"`
	
	### Folder containing source data file or files
	
	Instead of specifying a file name, you can just specify the name of the folder containing a source data file or files.
	
	To specify a source data folder, use an absolute file path, or a file path relative to the folder containing the Analytics project.
	
	#### Example
	
	`FROM "C:\Point of sale audit\Data\Transaction master files"`
	
	`FROM "Data\Transaction master files"`
	
	Note
	When you specify a folder, any worksheet in any Excel file in the folder is imported if the worksheet name matches the TABLE value.

TABLE *input_worksheets_*	The name of the worksheets or named ranges to import. A separate Analytics table is created for each imported worksheet or named range. Specify *input_worksheets_or_named_ranges* as a quoted string.

Name	Description
or_named_ranges	Use a wildcard (*) in place of unique characters in the names of worksheets or ranges. For example, `"Trans_*$"` selects the following worksheets: ◦ Trans_Jan ◦ Trans_Feb ◦ Trans_Mar ◦ and so on **Note** The wildcard character (*) stands for zero (0) or more occurrences of any letter, number, or special character. You can use a wildcard in more than one location. For example, `*Trans*$` selects: • `Trans_Jan` • `Jan_Trans` ## The meaning of the dollar sign ($) In an Excel file, worksheets are identified with a dollar sign ($) appended to the worksheet name (`Trans_Jan$`). The dollar sign is not visible in Excel. Named ranges are identified by the absence of a dollar sign (`Trans_Jan_commercial`). Specifying the dollar sign is not required when using IMPORT MULTIEXCEL. However, you should include it, or exclude it, in the following situations: ◦ **Include "$"** - if you want to import only worksheets, and no named ranges, include the dollar sign at the end of the worksheet name ◦ **Exclude "$"** - if you want to import named ranges, or worksheets and named ranges in a single import operation, do not include the dollar sign
PREFIX optional	Prepend the Excel file name to the name of the Analytics tables. **Tip** If worksheets in different files have the same name, prepending the Excel file name allows you to avoid table name conflicts.
KEEPTITLE optional	Treat the first row of data as field names instead of data. If omitted, generic field names are used. **Note** All first rows in the worksheets and named ranges that you import should use a consistent approach. First rows should be either field names, or data, across all data sets. Avoid mixing the two approaches in a single import operation. If the data sets have an inconsistent approach to first rows, use two separate import operations.
CHARMAX *max_field_length* optional	The maximum length in characters for any field in an Analytics table that originates as character data in a source data file.

Examples

The examples below assume that you have monthly transaction data for three years stored in three Excel files:

- `Transactions_FY18.xlsx`
- `Transactions_FY17.xlsx`
- `Transactions_FY16.xlsx`

Each Excel file has 12 worksheets – one for each month of the year. The worksheets also include some named ranges identifying various subsets of transactions.

> **Note**
> A separate Analytics table is created for each worksheet or named range that you import.

Import worksheets

Import all FY18 worksheets

You want to import all 12 monthly worksheets from the FY18 Excel file, and ignore any named ranges.

- you use the wildcard symbol (*) where the month occurs in each worksheet name
- you include the dollar sign ($) at the end of the worksheet name so that only worksheets are selected, and no named ranges

```
IMPORT MULTIEXCEL FROM "Transactions_FY18.xlsx" TABLE "Trans_*$"
```

Import all FY18 worksheets, keep field names, and specify maximum character field length

This example is the same as the one above, but you want to keep the field names from the Excel files, and also limit the length of character fields.

- you include KEEPTITLE to use the first row of Excel data as the field names
- you include CHARMAX 50 so that fields that originate as character data in the Excel file are limited to 50 characters in the resulting Analytics table

```
IMPORT MULTIEXCEL FROM "Transactions_FY18.xlsx" TABLE "Trans_*$"
KEEPTITLE CHARMAX 50
```

Import all worksheets from all three files

You want to import all 36 monthly worksheets from the three Excel files, and ignore any named ranges.

- you use the wildcard symbol (*) where the month occurs in each worksheet name
- you include the dollar sign ($) at the end of the worksheet name so that only worksheets are selected, and no named ranges
- you use the wildcard symbol (*) where the year occurs in each Excel file name
- as a way of reducing the chance of naming conflicts, you use PREFIX to prepend the name of the source Excel file to each Analytics table name

```
IMPORT MULTIEXCEL FROM "Transactions_FY*.xlsx" TABLE "Trans_*$" PREFIX
```

Import named ranges

Import all FY18 "Commercial_transaction" named ranges

You want to import all "Commercial_transaction" named ranges from the FY18 Excel file, and ignore worksheets, and other named ranges.

- you use the wildcard symbol (*) where a unique identifier occurs in the names of the different ranges
- you exclude the dollar sign ($) so that named ranges can be selected

```
IMPORT MULTIEXCEL FROM "Transactions_FY18.xlsx" TABLE "Commercial_trans-
action_*"
```

Import worksheets and named ranges

Import all FY18 worksheets and named ranges

You want to import all 12 monthly worksheets, and all named ranges, from the FY18 Excel file.

- with TABLE, you use only the wildcard symbol (*) so that all worksheets and named ranges in the file are selected
- you exclude the dollar sign ($) so that named ranges can be selected

```
IMPORT MULTIEXCEL FROM "Transactions_FY18.xlsx" TABLE "*"
```

Manage directories

Import all worksheets from all Excel files in the specified folder

You want to import all worksheets from all Excel files in the `C:\Point of sale audit\Data\Transaction master files` folder.

- with TABLE, you use only the wildcard symbol (*) so that all worksheets in each file are selected, and the dollar sign ($) so that only worksheets are selected, and no named ranges
- as a way of reducing the chance of naming conflicts, you use PREFIX to prepend the name of the source Excel file to each Analytics table name

```
IMPORT MULTIEXCEL FROM "C:\Point of sale audit\Data\Transaction master
files" TABLE "*$" PREFIX
```

Import all worksheets from all Excel files in the specified folder, and save the Analytics tables to another folder

This example is the same as the one above, but instead of saving the Analytics tables in the root project folder, you want to save them in the `C:\Point of sale audit\Data\Transaction working data` folder.

```
IMPORT MULTIEXCEL TO "C:\Point of sale audit\Data\Transaction working
data" FROM "C:\Point of sale audit\Data\Transaction master files" TABLE
"*$" PREFIX
```

Remarks

Multiple IMPORT EXCEL commands

The IMPORT MULTIEXCEL command actually performs multiple individual IMPORT EXCEL commands - one for each worksheet imported. If you double-click the IMPORT MULTIEXCEL entry in the log, the individual IMPORT EXCEL commands are displayed in the display area.

Last table imported is automatically opened

IMPORT MULTIEXCEL does not support the OPEN keyword. However, after the command executes, the last table imported is automatically opened.

Combining multiple worksheets after importing them

After you import multiple worksheets into individual Analytics tables you might want to combine them into a single Analytics table. For example, you could combine the data from twelve monthly tables into a single annual table containing all the data.

For information about combining multiple Analytics tables, see "APPEND command" on page 1565.

IMPORT ODBC command

Creates an Analytics table by defining and importing data from an ODBC data source.

ODBC stands for Open Database Connectivity, a standard method for accessing databases.

Syntax

```
IMPORT ODBC SOURCE source_name TABLE table_name <QUALIFIER data_qualifier>
<OWNER user_name> <USERID user_id> <PASSWORD num> <WHERE where_clause>
<TO table_name> <WIDTH max_field_length> <MAXIMUM max_field_length>
<FIELDS field <,...n>>
```

Parameters

Name	Description
SOURCE *source_name*	The data source name (DSN) of the ODBC data source to connect to. The DSN must already exist and be correctly configured. **Note** You are limited to data sources that use the Windows ODBC drivers that are installed on your computer. The Analytics native data connectors that can be used with the ACCESSDATA command may not be available with IMPORT ODBC.
TABLE *table_name*	The table name in the ODBC data source to import data from. *table_name* usually refers to a database table in the source data, but it can refer to anything Analytics imports as a table. For example, if you use the Microsoft Text Driver, *table_name* refers to the text file you want to import data from.
QUALIFIER *data_qualifier* optional	The character to use as the text qualifier to wrap and identify field values. You must specify the character as a quoted string. Use single quotation marks to specify the double quotation character: `'"'`.
OWNER *user_name* optional	The name of the database user account that owns the table you are connecting to.
USERID *user_id* optional	The username to access the data source.
PASSWORD *num*	The password definition to use.

Name	Description
optional	You do not use PASSWORD *num* to prompt for, or specify, an actual password. The password definition refers to a password previously supplied or set using the PASSWORD command, the SET PASSWORD command, or the PASSWORD analytic tag. *num* is the number of the password definition. For example, if two passwords have been previously supplied or set in a script, or when scheduling an analytic, `PASSWORD 2` specifies that password #2 is used. For more information about supplying or setting passwords, see: ◦ "PASSWORD command" on page 1893 ◦ "SET command" on page 1960 ◦ "PASSWORD tag" on page 2530
WHERE *where_clause* optional	An SQL WHERE clause that limits the records returned based on a criteria you specify. Must be a valid SQL statement and must be entered as a quoted string: `WHERE "SALARY > 50000".`
TO *table_name* optional	The name of the Analytics data file to create. Specify *table_name* as a quoted string with a .FIL file extension. For example, `TO "Invoices.FIL"`. By default, the data file (.FIL) is saved to the folder containing the Analytics project. Use either an absolute or relative file path to save the data file to a different, existing folder: ◦ `TO "C:\data\Invoices.FIL"` ◦ `TO "data\Invoices.FIL"`.
WIDTH *max_field_length* optional	The maximum length in characters for any field in the Analytics table that originates as character data in the source from which you are importing. You can enter any value between 1 and 254. The default value is 50. Data that exceeds the maximum field length is truncated when imported to Analytics.
MAXIMUM *max_field_length* optional	The maximum length in characters for text, note, or memo fields you are importing. You can enter any value between 1 and 1100. The default value is 100. Data that exceeds the maximum field length is truncated when imported to Analytics.
FIELDS *field* <,...*n*> optional	Individual fields in the source data to import. Specify the name. If you specify multiple fields, each field must be separated by a comma. If you omit FIELDS, all fields are imported. Enclosing the field names in quotation marks makes them case-sensitive. If you use quotation marks, the case of field names must exactly match between FIELDS and the ODBC data source. If you use quotation marks and the case of the field names does not match, the fields are not imported.

Name	Description
	Note FIELDS must be positioned last among the IMPORT ODBC parameters. If FIELDS is not positioned last, the command fails.

Examples

Importing data from SQL Server

You import data from a SQL Server database to an Analytics table named **Trans_Dec11**:

```
IMPORT ODBC SOURCE "SQLServerAudit" TABLE "Transactions" OWNER "audit"
TO "C:\ACL DATA\Trans_Dec11.FIL" WIDTH 100 MAXIMUM 200 FIELDS
"CARDNUM","CREDLIM","CUSTNO","PASTDUEAMT"
```

Remarks

Older method of connecting to ODBC data sources

The IMPORT ODBC command is the older method of connecting to ODBC-compliant data sources from Analytics. The new method of connecting to ODBC data sources uses the Data Access window and the ACCESSDATA command.

You can continue to use IMPORT ODBC in Analytics. However, this method of connecting is now available only in scripts and from the Analytics command line. You can no longer access this connection method in the **Data Definition Wizard**.

Suppress the time portion of datetime values

When using the IMPORT ODBC command to define an Analytics table you can suppress the time portion of datetime values by prefacing the command with the SET SUPPRESSTIME ON command.

This capability allows retrofitting Analytics scripts written prior to version 10.0 of Analytics, when the time portion of datetime values was automatically truncated. If SET SUPPRESSTIME ON is not added to these scripts, they do not run in the datetime-enabled version of Analytics.

For more information, see the "SET SUPPRESSTIME" section in "SET command" on page 1960.

IMPORT PDF command

Creates an Analytics table by defining and importing an Adobe PDF file.

Syntax

```
IMPORT PDF TO table <PASSWORD num> import_filename FROM source_filename
<SERVER profile_name> skip_length <PARSER "VPDF"> <PAGES page_range> {[record_
syntax] [field_syntax] <...n>} <...n>
```

```
record_syntax ::=
RECORD record_name record_type lines_in_record transparent [test_syntax]
<...n>
```

```
test_syntax ::=
TEST include_exclude match_type AT start_line,start_position,range logic text
```

```
field_syntax ::=
FIELD name type AT start_line,start_position SIZE length,lines_in_field DEC
value WID bytes PIC format AS display_name
```

Parameters

General parameters

Name	Description
TO *table*	The name of the Analytics table to import the data into. **Note** Table names are limited to 64 alphanumeric characters. The name can include the underscore character (_), but no other special characters, or any spaces. The name cannot start with a number.

Name	Description
PASSWORD *num* optional	Used for password-protected PDF files. The password definition to use. You do not use PASSWORD *num* to prompt for, or specify, an actual password. The password definition refers to a password previously supplied or set using the PASSWORD command, the SET PASSWORD command, or the PASSWORD analytic tag. *num* is the number of the password definition. For example, if two passwords have been previously supplied or set in a script, or when scheduling an analytic, `PASSWORD 2` specifies that password #2 is used. For more information about supplying or setting passwords, see: ◦ "PASSWORD command" on page 1893 ◦ "SET command" on page 1960 ◦ "PASSWORD tag" on page 2530
import_filename	The name of the Analytics data file to create. Specify *import_filename* as a quoted string with a .FIL file extension. For example, `"Invoices.FIL"`. By default, the data file (.FIL) is saved to the folder containing the Analytics project. Use either an absolute or relative file path to save the data file to a different, existing folder: ◦ `"C:\data\Invoices.FIL"` ◦ `"data\Invoices.FIL"`
FROM *source_filename*	The name of the source data file. *source_filename* must be a quoted string. If the source data file is not located in the same directory as the Analytics project, you must use an absolute path or a relative path to specify the file location: ◦ `"C:\data\source_filename"` ◦ `"data\source_filename"`
SERVER *profile_name* optional	The profile name for the server that contains the data that you want to import.
skip_length optional	The number of bytes to skip at the start of the file. For example, if the first 32 bytes contains header information, specify a skip length value of 32 to omit this information. **Note** For Unicode data, specify an even number of bytes only. Specifying an odd number of bytes can cause problems with subsequent processing of the imported data.
PARSER "VPDF" optional	Use the VeryPDF parser to parse the PDF file during the file definition process. If you omit PARSER, the default Xpdf parser is used. If you are importing the PDF file for the first time, and you have no reason to do otherwise, use the default Xpdf parser. If you have already encountered data

Name	Description
	alignment issues when using Xpdf, use the VeryPDF parser to see if the parsing results are better.
PAGES *page_range* optional	The pages to include if you do not want to import all of the pages in the PDF file. *page_range* must be specified as a quoted string. You can specify: ◦ individual pages separated by commas (1,3,5) ◦ page ranges (2-7) ◦ a combination of pages and ranges (1, 3, 5-7, 11) If you omit PAGES, all pages in the PDF file are imported.

RECORD parameter

General record definition information.

> **Note**
> Some of the record definition information is specified using numeric codes that map to options in the Data Definition Wizard.
> **In scripts, specify the numeric code, not the option name.**

Name	Description
RECORD *record_name*	The name of the record in the Data Definition Wizard. Specifying *record_name* is required in the IMPORT PDF command, but the *record_name* value does not appear in the resulting Analytics table. In the Data Definition Wizard, Analytics provides default names based on the type of record: ◦ `Detail` ◦ `Header`*n* ◦ `Footer`*n* You can use the default names, or specify different names.
record_type	The three possible record types when defining a PDF file: ◦ `0` - detail ◦ `1` - header ◦ `2` - footer > **Note** > You can define multiple sets of header and footer records in a single execution of IMPORT PDF, but only one set of detail records.
lines_in_record	The number of lines occupied by a record in the PDF file. You can define single-line or multiline records to match the data in the PDF file.

Name	Description
transparent	The transparency setting for a header record. **Note** Applies to header records only. ◦ `0` - not transparent ◦ `1` - transparent Transparent header records do not split multiline detail records. If a header record splits a multiline detail record in the source PDF file, which can happen at a page break, specifying `1` (transparent) unifies the detail record in the resulting Analytics table.

TEST parameter

The criteria for defining a set of records in the PDF file. You can have one or more occurrences of TEST (up to 8) for each occurrence of RECORD.

> **Note**
> Some of the criteria are specified using numeric codes that map to options in the Data Definition Wizard (option names are shown in parentheses below).
> **In scripts, specify the numeric code, not the option name.**

Name	Description
TEST *include_exclude*	How to treat matching data: ◦ `0` - (**Include**) data meeting the criteria is included in the set of records ◦ `1` - (**Exclude**) data meeting the criteria is excluded from the set of records
match_type	The type of matching to perform: ◦ `0` - (**Exact Match**) matching records must contain the specified character, or string of characters, in the specified start line, starting at the specified position ◦ `2` - (**Alpha**) matching records must contain one or more alpha characters, in the specified start line, at the specified start position, or in all positions of the specified range ◦ `3` - (**Numeric**) matching records must contain one or more numeric characters, in the specified start line, at the specified start position, or in all positions of the specified range ◦ `4` - (**Blank**) matching records must contain one or more blank spaces, in the specified start line, at the specified start position, or in all positions of the specified range ◦ `5` - (**Non-Blank**) matching records must contain one or more non-blank characters (includes special characters), in the specified start line, at the specified start position, or in all positions of the specified range ◦ `7` - (**Find in Line**) matching records must contain the specified character, or string of characters, anywhere in the specified start line ◦ `8` - (**Find in Range**) matching records must contain the specified character, or

Name	Description
	string of characters, in the specified start line, anywhere in the specified range ◦ `10` - (**Custom Map**) matching records must contain characters that match the specified character pattern, in the specified start line, starting at the specified position
AT *start_line*, *start_position*, *range*	◦ ***start_line*** - the line of a record that the criteria apply to For example, if you create a custom map to match zip codes, and the zip codes appear on the third line of a three-line address record, you must specify `3` in *start_line*. **Note** For single-line records, the *start_line* value is always `1`. ◦ ***start_position*** - the starting byte position in the PDF file for the comparison against the criteria ◦ ***range*** - the number of bytes from the starting byte position in the PDF file to use in the comparison against the criteria If you are using starting byte position only, without a range, specify `0` for *range*. **Note** non-Unicode Analytics: 1 byte = 1 character Unicode Analytics: 2 bytes = 1 character
logic	The logical relations between criteria: ◦ `0` - (**And**) the current and the next criteria are related with a logical AND ◦ `1` - (**Or**) the current and the next criteria are related with a logical OR ◦ `4` - (**New Group** > **And**) the current criterion is the last in a group of logical criteria, and the current group and the next group are related with a logical AND ◦ `5` - (**New Group** > **Or**) the current criterion is the last in a group of logical criteria, and the current group and the next group are related with a logical OR ◦ `7` - (**End**) the current criterion is the last in a group of logical criteria
text	Literal or wildcard characters to match against: ◦ **For Exact Match, Find in Line, or Find in Range** - specifies the character, or string of characters, that uniquely identifies the set of records in the PDF file ◦ **For Custom Map** - specifies the character pattern that uniquely identifies the set of records in the PDF file The **Custom Map** option uses the same syntax as the "MAP( ) function" on page 2224. For other match types, *text* is an empty string `""`.

FIELD parameters

Field definition information.

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>FIELD name type</td><td>The individual fields to import from the source data file, including the name and data type of the field. To exclude a field from being imported, do not specify it.
For information about type, see "Identifiers for field data types" on page 1821.</td></tr>
<tr><td>AT start_line, start_position</td><td>◦ start_line - the start line of the field in the record in the PDF file
For multiline records in a PDF file, start_line allows you to start a field at any line of the record. start_line is always 1 if lines_in_record is 1.
◦ start_position - the starting byte position of the field in the PDF file
Note<table><tr><td>non-Unicode Analytics</td><td>1 byte = 1 character</td></tr><tr><td>Unicode Analytics</td><td>2 bytes = 1 character</td></tr></table>In Unicode Analytics, typically you should specify an odd-numbered starting byte position. Specifying an even-numbered starting position can cause characters to display incorrectly.</td></tr>
<tr><td>SIZE length, lines_in_field</td><td>◦ length - the length in bytes of the field in the Analytics table layout
Note<table><tr><td>non-Unicode Analytics</td><td>1 byte = 1 character</td></tr><tr><td>Unicode Analytics</td><td>2 bytes = 1 character</td></tr></table>In Unicode Analytics, specify an even number of bytes only. Specifying an odd number of bytes can cause characters to display incorrectly.
◦ lines_in_field - the number of lines occupied by a single field value in the PDF file
You can define single-line or multiline fields to match the data in the file.
Note
The number of lines specified for a field cannot exceed the number of lines specified for the record containing the field.</td></tr>
<tr><td>DEC value</td><td>The number of decimals for numeric fields.</td></tr>
<tr><td>WID bytes</td><td>The display width of the field in bytes.
The specified value controls the display width of the field in Analytics views and reports. The display width never alters data, however it can hide data if it is shorter than the field length.</td></tr>
<tr><td>PIC format</td><td>Note
Applies to numeric or datetime fields only.</td></tr>
</table>

Name	Description
	◦ **numeric fields** - the display format of numeric values in Analytics views and reports ◦ **datetime fields** - the physical format of datetime values in the source data (order of date and time characters, separators, and so on) **Note** For datetime fields, *format* must exactly match the physical format in the source data. For example, if the source data is 12/31/2014, you must enter the format as `"MM/DD/YYYY"`. *format* must be enclosed in quotation marks.
AS *display_name*	The display name (alternate column title) for the field in the view in the new Analytics table. Specify *display_name* as a quoted string. Use a semi-colon (;) between words if you want a line break in the column title. AS is required when you are defining FIELD. To make the display name the same as the field name, enter a blank *display_name* value using the following syntax: `AS ""`. Make sure there is no space between the two double quotation marks.

Examples

Importing data from a specific page of a PDF file

You import data from page 1 of the password-protected PDF file, `Vendors.pdf`.

One set of detail records, with three fields, is created in the resulting Analytics table, **Vendor_List**:

```
IMPORT PDF TO Vendor_List PASSWORD 1 "Vendor_List.FIL" FROM "Ven-
dors.pdf" 2 PAGES "1" RECORD "Detail" 0 1 0 TEST 0 3 AT 1,1,0 7 "" FIELD
"Vendor_Number" C AT 1,1 SIZE 10,1 DEC 0 WID 10  PIC "" AS "" FIELD
"Vendor_Name" C AT 1,33 SIZE 58,1 DEC 0 WID 58  PIC "" AS "" FIELD
"Last_Active_Date" D AT 1,277 SIZE 20,1 DEC 0 WID 20  PIC "DD/MM/YYYY"
AS ""
```

Remarks

For more information about how this command works, see "Defining and importing print image (report) files and PDF files" on page 261.

Troubleshooting PDF imports in the Unicode edition of Analytics

If you encounter issues when you import a PDF file using the Unicode edition of Analytics, the problem may be related to length specifications:

- If foreign language characters are appearing unexpectedly, or the layout in the resulting Analytics table is skewed, check that SIZE *length* is set to an even number.

  Specifying an odd number of bytes for SIZE *length* can cause problems with processing of the imported data.
- If the Analytics table is created, but contains zero records, trying setting *skip_length* to `2`, or some other even number if there is header data at the beginning of the file that you want to skip.

Identifiers for field data types

The table below lists the letters that you must use when specifying *type* for `FIELD`. Each letter corresponds to an Analytics data type.

For example, if you are defining a Last Name field, which requires a character data type, you would specify "C": `FIELD "Last_Name" C`.

For more information, see "Data types in Analytics" on page 739.

> **Note**
>
> When you use the **Data Definition Wizard** to define a table that includes EBCDIC, Unicode, or ASCII fields, the fields are automatically assigned the letter "C" (for the CHARACTER type).
>
> When you enter an IMPORT statement manually, or edit an existing IMPORT statement, you can substitute the more specific letters "E" or "U" for EBCDIC or Unicode fields.

Letter	Analytics Data type
A	ACL
B	BINARY

Letter	Analytics Data type
C	CHARACTER
D	DATETIME
E	EBCDIC
F	FLOAT
G	ACCPAC
I	IBMFLOAT
K	UNSIGNED
L	LOGICAL
N	PRINT
P	PACKED
Q	BASIC
R	MICRO
S	CUSTOM
T	PCASCII
U	UNICODE
V	VAXFLOAT
X	NUMERIC
Y	UNISYS
Z	ZONED

IMPORT PRINT command

Creates an Analytics table by defining and importing a Print Image (Report) file.

Syntax

```
IMPORT PRINT TO table import_filename FROM source_filename <SERVER profile_
name> character_set_value <code_page_number> {[record_syntax] [field_syntax]
<...n>} <...n>
```

```
record_syntax ::=
RECORD record_name record_type lines_in_record transparent [test_syntax]
<...n>
```

```
test_syntax ::=
TEST include_exclude match_type AT start_line,start_position,range logic text
```

```
field_syntax ::=
FIELD name type AT start_line,start_position SIZE length,lines_in_field DEC
value WID bytes PIC format AS display_name
```

Parameters

General parameters

Name	Description
TO *table*	The name of the Analytics table to import the data into. **Note** Table names are limited to 64 alphanumeric characters. The name can include the underscore character (_), but no other special characters, or any spaces. The name cannot start with a number.

Name	Description
import_filename	The name of the Analytics data file to create. Specify *import_filename* as a quoted string with a .FIL file extension. For example, `"Invoices.FIL"`. By default, the data file (.FIL) is saved to the folder containing the Analytics project. Use either an absolute or relative file path to save the data file to a different, existing folder: ◦ `"C:\data\Invoices.FIL"` ◦ `"data\Invoices.FIL"`
FROM *source_filename*	The name of the source data file. *source_filename* must be a quoted string. If the source data file is not located in the same directory as the Analytics project, you must use an absolute path or a relative path to specify the file location: ◦ `"C:\data\source_filename"` ◦ `"data\source_filename"`
SERVER *profile_name* optional	The profile name for the server that contains the data that you want to import.
character_set_value	The character set used to encode the Print Image (Report) file. The following values are supported: ◦ `0` - ASCII ◦ `1` - EBCDIC ◦ `2` - Unicode ◦ `3` - Encoded text
code_page_number optional	If you specified `3` (Encoded text) for *character_set_value*, you must also enter a code page number.

RECORD parameter

General record definition information.

> **Note**
> Some of the record definition information is specified using numeric codes that map to options in the Data Definition Wizard.
> **In scripts, specify the numeric code, not the option name.**

Name	Description
RECORD *record_name*	The name of the record in the Data Definition Wizard. Specifying *record_name* is required in the IMPORT PRINT command, but the *record_name* value does not appear in the resulting Analytics table. In the Data Definition Wizard, Analytics provides default names based on the type of

Name	Description
	record: ◦ `Detail` ◦ `Header`*n* ◦ `Footer`*n* You can use the default names, or specify different names.
record_type	The three possible record types when defining a Print Image file: ◦ `0` - detail ◦ `1` - header ◦ `2` - footer **Note** You can define multiple sets of header and footer records in a single execution of IMPORT PRINT, but only one set of detail records.
lines_in_record	The number of lines occupied by a record in the Print Image file. You can define single-line or multiline records to match the data in the file.
transparent	The transparency setting for a header record. **Note** Applies to header records only. ◦ `0` - not transparent ◦ `1` - transparent Transparent header records do not split multiline detail records. If a header record splits a multiline detail record in the source Print Image file, which can happen at a page break, specifying `1` (transparent) unifies the detail record in the resulting Analytics table.

TEST parameter

The criteria for defining a set of records in the Print Image file. You can have one or more occurrences of TEST (up to 8) for each occurrence of RECORD.

> **Note**
> Some of the criteria are specified using numeric codes that map to options in the Data Definition Wizard (option names are shown in parentheses below).
> **In scripts, specify the numeric code, not the option name.**

Name	Description
TEST *include_exclude*	How to treat matching data:

Name	Description
	◦ `0` - (**Include**) data meeting the criteria is included in the set of records ◦ `1` - (**Exclude**) data meeting the criteria is excluded from the set of records
match_type	The type of matching to perform: ◦ `0` - (**Exact Match**) matching records must contain the specified character, or string of characters, in the specified start line, starting at the specified position ◦ `2` - (**Alpha**) matching records must contain one or more alpha characters, in the specified start line, at the specified start position, or in all positions of the specified range ◦ `3` - (**Numeric**) matching records must contain one or more numeric characters, in the specified start line, at the specified start position, or in all positions of the specified range ◦ `4` - (**Blank**) matching records must contain one or more blank spaces, in the specified start line, at the specified start position, or in all positions of the specified range ◦ `5` - (**Non-Blank**) matching records must contain one or more non-blank characters (includes special characters), in the specified start line, at the specified start position, or in all positions of the specified range ◦ `7` - (**Find in Line**) matching records must contain the specified character, or string of characters, anywhere in the specified start line ◦ `8` - (**Find in Range**) matching records must contain the specified character, or string of characters, in the specified start line, anywhere in the specified range ◦ `10` - (**Custom Map**) matching records must contain characters that match the specified character pattern, in the specified start line, starting at the specified position
AT *start_line*, *start_position*, *range*	◦ ***start_line*** - the line of a record that the criteria apply to For example, if you create a custom map to match zip codes, and the zip codes appear on the third line of a three-line address record, you must specify `3` in *start_line*. **Note** For single-line records, the *start_line* value is always `1`. ◦ ***start_position*** - the starting byte position in the Print Image file for the comparison against the criteria ◦ ***range*** - the number of bytes from the starting byte position in the Print Image file to use in the comparison against the criteria If you are using starting byte position only, without a range, specify `0` for *range*.

Name	Description
	Note non-Unicode Analytics: 1 byte = 1 character Unicode Analytics, extended ASCII (ANSI) data: 1 byte = 1 character Unicode Analytics, Unicode data: 2 bytes = 1 character For Unicode data, *range* must be an even number of bytes. For example, `50,59` (10 bytes). Specifying an odd number of bytes can prevent correct matching against criteria.
logic	The logical relations between criteria: ◦ `0` - (**And**) the current and the next criteria are related with a logical AND ◦ `1` - (**Or**) the current and the next criteria are related with a logical OR ◦ `4` - (**New Group** > **And**) the current criterion is the last in a group of logical criteria, and the current group and the next group are related with a logical AND ◦ `5` - (**New Group** > **Or**) the current criterion is the last in a group of logical criteria, and the current group and the next group are related with a logical OR ◦ `7` - (**End**) the current criterion is the last in a group of logical criteria
text	Literal or wildcard characters to match against: ◦ **For Exact Match, Find in Line, or Find in Range** - specifies the character, or string of characters, that uniquely identifies the set of records in the Print Image file ◦ **For Custom Map** - specifies the character pattern that uniquely identifies the set of records in the Print Image file The **Custom Map** option uses the same syntax as the "MAP( ) function" on page 2224. For other match types, *text* is an empty string `""`.

FIELD parameters

Field definition information.

Name	Description
FIELD *name type*	The individual fields to import from the source data file, including the name and data type of the field. To exclude a field from being imported, do not specify it. For information about *type*, see "Identifiers for field data types" on page 1830.
AT *start_line*, *start_position*	◦ ***start_line*** - the start line of the field in the record in the Print Image file For multiline records in a Print Image file, *start_line* allows you to start a field at any line of the record. *start_line* is always `1` if *lines_in_record* is `1`. ◦ ***start_position*** - the starting byte position of the field in the Print Image file

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>
Note
<table>
<tr><td>non-Unicode Analytics</td><td>1 byte = 1 character</td></tr>
<tr><td>Unicode Analytics, extended ASCII (ANSI) data</td><td>1 byte = 1 character</td></tr>
<tr><td>Unicode Analytics, Unicode data</td><td>2 bytes = 1 character</td></tr>
</table>
</td></tr>
<tr><td>SIZE length, lines_in_field</td><td>

length - the length in bytes of the field in the Analytics table layout
Note
<table>
<tr><td>non-Unicode Analytics</td><td>1 byte = 1 character</td></tr>
<tr><td>Unicode Analytics, extended ASCII (ANSI) data</td><td>1 byte = 1 character</td></tr>
<tr><td>Unicode Analytics, Unicode data</td><td>2 bytes = 1 character</td></tr>
</table>
For Unicode data, specify an even number of bytes only. Specifying an odd number of bytes can cause characters to display incorrectly.
lines_in_field - the number of lines occupied by a single field value in the Print Image file
You can define single-line or multiline fields to match the data in the file.
Note
The number of lines specified for a field cannot exceed the number of lines specified for the record containing the field.

</td></tr>
<tr><td>DEC value</td><td>The number of decimals for numeric fields.</td></tr>
<tr><td>WID bytes</td><td>The display width of the field in bytes.
The specified value controls the display width of the field in Analytics views and reports. The display width never alters data, however it can hide data if it is shorter than the field length.</td></tr>
<tr><td>PIC format</td><td>
Note
Applies to numeric or datetime fields only.

numeric fields - the display format of numeric values in Analytics views and reports
datetime fields - the physical format of datetime values in the source data (order of date and time characters, separators, and so on)

</td></tr>
</table>

Name	Description
	Note For datetime fields, *format* must exactly match the physical format in the source data. For example, if the source data is 12/31/2014, you must enter the format as `"MM/DD/YYYY"`. *format* must be enclosed in quotation marks.
AS *display_name*	The display name (alternate column title) for the field in the view in the new Analytics table. Specify *display_name* as a quoted string. Use a semi-colon (;) between words if you want a line break in the column title. AS is required when you are defining FIELD. To make the display name the same as the field name, enter a blank *display_name* value using the following syntax: `AS ""`. Make sure there is no space between the two double quotation marks.

Examples

Importing data from a Print Image (Report) file

You import data from the Print Image (Report) file, `Report.txt`.

One header record, and one set of detail records, with five fields, is created in the resulting Analytics table, **Inventory_report**:

```
IMPORT PRINT TO Inventory_report "Inventory_report.FIL" FROM
"Report.txt" 0 RECORD "Header1" 1 1 0 TEST 0 0 AT 1,17,0 7 ":" FIELD
"Field_1" C AT 1,19 SIZE 2,1 DEC 0 WID 2 PIC "" AS "Prod Class" FIELD
"Field_2" C AT 1,24 SIZE 31,1 DEC 0 WID 31 PIC "" AS "Prod Description"
RECORD "Detail" 0 1 0 TEST 0 0 AT 1,59,59 7 "." FIELD "Field_3" X AT 1,6
SIZE 9,1 DEC 0 WID 9 PIC "" AS "Item ID" FIELD "Field_4" C AT 1,16 SIZE
24,1 DEC 0 WID 24 PIC "" AS "Item Desc." FIELD "Field_5" N AT 1,40 SIZE
10,1 DEC 0 WID 10 PIC "" AS "On Hand" FIELD "Field_6" N AT 1,50 SIZE
12,1 DEC 2 WID 12 PIC "" AS "Cost" FIELD "Field_7" N AT 1,62 SIZE 12,1
DEC 2 WID 12 PIC "" AS "Total"
```

Remarks

For more information about how this command works, see "Defining and importing print image (report) files and PDF files" on page 261.

Identifiers for field data types

The table below lists the letters that you must use when specifying *type* for `FIELD`. Each letter corresponds to an Analytics data type.

For example, if you are defining a Last Name field, which requires a character data type, you would specify "C": `FIELD "Last_Name" C`.

For more information, see "Data types in Analytics" on page 739.

> **Note**
>
> When you use the **Data Definition Wizard** to define a table that includes EBCDIC, Unicode, or ASCII fields, the fields are automatically assigned the letter "C" (for the CHARACTER type).
>
> When you enter an IMPORT statement manually, or edit an existing IMPORT statement, you can substitute the more specific letters "E" or "U" for EBCDIC or Unicode fields.

Letter	Analytics Data type
A	ACL
B	BINARY
C	CHARACTER
D	DATETIME
E	EBCDIC
F	FLOAT
G	ACCPAC
I	IBMFLOAT
K	UNSIGNED
L	LOGICAL
N	PRINT
P	PACKED
Q	BASIC
R	MICRO

Letter	Analytics Data type
S	CUSTOM
T	PCASCII
U	UNICODE
V	VAXFLOAT
X	NUMERIC
Y	UNISYS
Z	ZONED

IMPORT SAP command

Creates an Analytics table by importing data from an SAP system using Direct Link.

> **Note**
> The IMPORT SAP command is only supported if Direct Link is installed and configured on your local computer and on your organization's SAP system.

Syntax

```
IMPORT SAP PASSWORD num TO table_name SAP SOURCE "SAP AGENT" import_details
```

Parameters

Name	Description
PASSWORD *num*	The password definition to use. You do not use PASSWORD *num* to prompt for, or specify, an actual password. The password definition refers to a password previously supplied or set using the PASSWORD command, the SET PASSWORD command, or the PASSWORD analytic tag. *num* is the number of the password definition. For example, if two passwords have been previously supplied or set in a script, or when scheduling an analytic, `PASSWORD 2` specifies that password #2 is used. For more information about supplying or setting passwords, see: ◦ "PASSWORD command" on page 1893 ◦ "SET command" on page 1960 ◦ "PASSWORD tag" on page 2530 **Note** The password is used to access the SAP system.
TO *table_name*	The name of the Analytics table to import the data into. **Note** Table names are limited to 64 alphanumeric characters. The name can include the underscore character (_), but no other special characters, or any spaces. The name cannot start with a number.
SAP SOURCE "SAP AGENT"	Required for importing SAP data. "SAP AGENT" is the only available option.

Name	Description
import_details	The details of the query. Must be enclosed by the <q></q> tags, and uses the tags listed in "Direct Link query tags" on page 1835 to define the query. The physical size of this value can be up to 16 KB.

Examples

Performing a multi-table query

This example performs a multi-table query using the IMPORT SAP command.

Correct order and nesting of the tags is necessary to create a valid query string. The tags in the example are ordered and nested correctly. Use this example to determine the required order and nesting of IMPORT SAP query tags.

> **Note**
> To assist readability, this example is formatted using multiple lines. In your script, the command and the query string must be entered without any line breaks.

> **Tip**
> The syntax of an IMPORT SAP query string is typically complex. The best way to add IMPORT SAP commands with query strings to your scripts is to copy an existing IMPORT SAP command from the **Log** tab in Analytics, then edit the query tags as necessary.

```
IMPORT SAP PASSWORD 1 TO Purchasing_doc SAP SOURCE "SAP AGENT"
<q version="6.0">
  <s>0</s>
  <d>IDES</d>
  <u>mzunini</u>
  <c>800</c>
  <lg>en</lg>
  <cf>C:\ACL Data\Purchasing_doc.fil</cf>
  <sf>E:\Data\DL_JSMITH111107.DAT</sf>
  <jcount>11110701</jcount>
  <jname>DL_JSMITH111107.DAT</jname>
  <dl>75</dl>
  <m>2</m>
  <dt>20140321</dt>
```

```
<tm>033000</tm>
<r>500</r>
<ar>0</ar>
<e>500</e>
<ts>
  <t>
    <n>EKKO</n>
    <a>T00001</a>
    <td>Purchasing Document Header</td>
    <fs>
      <f>EBELN</f>
      <f>BUKRS</f>
      <f>BSTYP</f>
      <f>BSART</f>
      <f>STATU</f>
      <f>WKURS</f>
    </fs>
    <wc>
      <w>
         <f>BUKRS</f>
         <o>0</o>
         <l>1000</l>
         <h></h>
      </w>
    </wc>
  </t>
  <t>
    <n>EKPO</n>
    <a>T00002</a>
    <td>Purchasing Document Item</td>
    <fs>
      <f>EBELP</f>
      <f>WERKS</f>
      <f>MENGE</f>
      <f>BRTWR</f>
    </fs>
    <wc></wc>
  </t>
</ts>
<js>
  <jc>
    <pt>
      <pa>T00001</pa>
      <pf>EBELN</pf>
    </pt>
```

```
      <ct>
        <ca>T00002</ca>
        <cf>EBELN</cf>
      </ct>
    </jc>
  </js>
</q>
```

Remarks

The table in "Direct Link query tags" below lists the tags that can be included in the *import_details* parameter. The **Required** column uses the following values to indicate when tags must be present:

- **Y** - Required
- **N** - Optional
- **M** - Required for multi-table queries only
- **B** - Required, but no value should be passed
- **W** - Optional when filters are used
- **S** - Required when scheduled mode is specified

Direct Link query tags

Name	Tag	Required	Description
Table Alias	<a>	M	The alias that uniquely identifies the table within the query. This allows the same table to be used more than once. The maximum length is 6 characters.
All Rows	<ar>	Y	Indicates that all matching rows should be returned as part of the query's result set. Valid values are: **1** - Overrides the number of records specified in the <r> tag (Maximum Rows) **0** - Returns the number of records specified in the <r> tag (Maximum Rows) This tag always appears after the <r></r> tag.
Client	<c>	N	The client within the SAP system.
Child Table Alias	<ca>	M	The alias of the child table.
Child Table Field	<cf>	M	The field in the child table that the join condition is based on.

Name	Tag	Required	Description
Client Filename	<cf>	Y	Identifies the target file on the client system where the results of the query will be stored.
Child Table	<ct>	M	The child table in the join condition.
Destination	<d>	N	Identifies a destination in the SAP RFC library file (`sapnwrfc.ini`) that is used to locate an SAP system.
Data Length	<dl>	B	The number of characters in each row, including carriage return and line feed characters indicating the end of the record (CR+LF, or the hexadecimal characters 0D+0A).
Date	<dt>	S	Required when using scheduled mode. Specifies the time to run the SAP job. Must be formatted as YYYYMMDD. For example, December 31, 2014 must be specified as 20141231.
Expected Rows	<e>	B	The expected number of rows the query will return.
Field Name	<f>	Y	The native field name.
Filter Field	<f>	W	The native field name that the filter applies to.
Fields	<fs>	Y	The list of fields in the table that will be returned as part of the query results.
High Value	<h>	W	Contains the high value when using the Between operator. Ignored when using any other operator.
Join Condition	<jc>	M	The join condition.
Job Count	<jcount>	B	Used internally by SAP to identify a Background mode query.
Job Name	<jname>	B	Used internally by SAP to identify a Background mode query.
Join Relation-ships	<js>	Y	The list of join conditions that link tables within the query.
Join Switch	<jw>	N	Numeric equivalent of the join switch enumerated type. Valid values are: **0** - Inner Join **1** - Left Outer Join
Low Value	<l>	W	Contains either the lowest value when using the Between operator or the value when using any other operator.

Name	Tag	Required	Description
Language	<lg>	Y	Language identifier used to determine the locale of fields in the SAP database.
Mode	<m>	Y	Numeric equivalent of the submission mode enumerated type. Valid values are: **0** - Extract Now **1** - Background **2** - Scheduled
Table Name	<n>	Y	The native table name.
Operator	<o>	W	Numeric equivalent of the operator enumerated type. Valid values are: **0** - Equal to (=) **1** - Not equal to (<>) **2** - Less than (<) **3** - Less than or equal to (<=) **4** - Greater than (>) **5** - Greater than or equal to (>=) **6** - Between **7** - Contains
Parent Table Alias	<pa>	M	The alias of the parent table.
Parent Table Field	<pf>	M	The field in the parent table the join condition is based on.
Parent Table	<pt>	M	The parent table in the join condition.
Query	<q>	Y	Encloses a query.
Maximum Rows	<r>	Y	The maximum number of rows the query should return.
Selected	<s>	Y	If the <s> tag appears below the <f> tag, it indicates whether the field will be returned as part of the query's result set.
System	<s>	Y	If the <s> tag appears below the <q> tag, it identifies the type of system this query is used against (currently only SAP is supported).
Server Filename	<sf>	B	Identifies the file on the server that holds the results of a Background mode query.

Name	Tag	Required	Description
Server Group Name	<sg>	N	The name of the server group. Maximum 20 characters.
Server Name	<sn>	N	The name of the server. Maximum 20 characters.
Table	<t>	Y	The table.
Table Description	<td>	Y	The table description from the SAP data dictionary. It should always appear below the <a> tag.
Time	<tm>	S	Required when using scheduled mode. Specifies the time to run the SAP job. Must be formatted as hhmmss. For example, 2:30 pm must be specified as 143000.
Tables	<ts>	Y	The list of tables from which the query will extract data.
Table Type	<tt>	Y	The type of SAP table. Valid values are: **0** - clustered **1** - transparent **2** - pooled **3** - view
Username	<u>	N	The user's logon name.
Filter	<w>	W	The filter applied to the table's data.
Filters	<wc>	W	The list of filters that will be applied to the data contained within the table.
Filter Switch	<ws>	N	Numeric equivalent of the filter switch enumerated type. Valid values are: **0** - (Or) And (Or) **1** - (And) Or (And)

IMPORT XBRL command

Creates an Analytics table by defining and importing an XBRL file.

Syntax

```
IMPORT XBRL TO table import_filename FROM source_filename CONTEXT context_name
<...n> [field_syntax] <...n> <IGNORE field_num> <...n>
```

```
field_syntax ::=
FIELD name type AT start_position DEC value WID bytes PIC format AS display_
name
```

Parameters

Name	Description
TO *table*	The name of the Analytics table to import the data into. **Note** Table names are limited to 64 alphanumeric characters. The name can include the underscore character (_), but no other special characters, or any spaces. The name cannot start with a number.
import_filename	The name of the Analytics data file to create. Specify *import_filename* as a quoted string with a .FIL file extension. For example, `"Invoices.FIL"`. By default, the data file (.FIL) is saved to the folder containing the Analytics project. Use either an absolute or relative file path to save the data file to a different, existing folder: ◦ `"C:\data\Invoices.FIL"` ◦ `"data\Invoices.FIL"`
FROM *source_filename*	The name of the source data file. *source_filename* must be a quoted string. If the source data file is not located in the same directory as the Analytics project, you must use an absolute path or a relative path to specify the file location: ◦ `"C:\data\source_filename"`

Name	Description
	◦ `"data\source_filename"`
CONTEXT *context_name*	The XBRL context to define the table from. If you specify more than one context, all contexts must be of the same type (instant, period, or forever).
FIELD *name type*	The individual fields to import from the source data file, including the name and data type of the field. To exclude a field from being imported, do not specify it. For information about *type*, see "Identifiers for field data types" on the facing page.
AT *start_position*	The starting byte position of the field in the Analytics data file. **Note** non-Unicode Analytics: 1 byte = 1 character Unicode Analytics: 2 bytes = 1 character In Unicode Analytics, typically you should specify an odd-numbered starting byte position. Specifying an even-numbered starting position can cause characters to display incorrectly.
DEC *value*	The number of decimals for numeric fields.
WID *bytes*	The length in bytes of the field in the Analytics table layout. **Note** non-Unicode Analytics: 1 byte = 1 character Unicode Analytics: 2 bytes = 1 character In Unicode Analytics, specify an even number of bytes only. Specifying an odd number of bytes can cause characters to display incorrectly.
PIC *format*	**Note** Applies to numeric or datetime fields only. ◦ **numeric fields** - the display format of numeric values in Analytics views and reports ◦ **datetime fields** - the physical format of datetime values in the source data (order of date and time characters, separators, and so on) **Note** For datetime fields, *format* must exactly match the physical format in the source data. For example, if the source data is 12/31/2014, you must enter the format as `"MM/DD/YYYY"`. *format* must be enclosed in quotation marks.

Name	Description
AS *display_name*	The display name (alternate column title) for the field in the view in the new Analytics table. Specify *display_name* as a quoted string. Use a semi-colon (;) between words if you want a line break in the column title. AS is required when you are defining FIELD. To make the display name the same as the field name, enter a blank *display_name* value using the following syntax: `AS ""`. Make sure there is no space between the two double quotation marks.
IGNORE *field_num* optional	Excludes a field from the table layout. *field_num* specifies the position of the field in the source data. For example, `IGNORE 5` excludes the fifth field in the source data from the Analytics table layout.

Examples

Importing an XBRL file to an Analytics table

You import data from the **Current_AsOf** context in an XBRL file to an Analytics table called **Financials**:

```
IMPORT XBRL TO Financials "Financials.fil" FROM "Fin-
ancialStatemenXBRL.xml" CONTEXT "Current_AsOf" FIELD "Item" C AT 1 DEC 0
WID 57 PIC "" AS "" FIELD "Value" X AT 58 DEC 0 WID 7 PIC "" AS ""
IGNORE 1 IGNORE 3
```

Remarks

For more information about how this command works, see "Import an XBRL file" on page 335.

Identifiers for field data types

The table below lists the letters that you must use when specifying *type* for `FIELD`. Each letter corresponds to an Analytics data type.

For example, if you are defining a Last Name field, which requires a character data type, you would specify "C": `FIELD "Last_Name" C`.

For more information, see "Data types in Analytics" on page 739.

> **Note**
>
> When you use the **Data Definition Wizard** to define a table that includes EBCDIC, Unicode, or ASCII fields, the fields are automatically assigned the letter "C" (for the CHARACTER type).
>
> When you enter an IMPORT statement manually, or edit an existing IMPORT statement, you can substitute the more specific letters "E" or "U" for EBCDIC or Unicode fields.

Letter	Analytics Data type
A	ACL
B	BINARY
C	CHARACTER
D	DATETIME
E	EBCDIC
F	FLOAT
G	ACCPAC
I	IBMFLOAT
K	UNSIGNED
L	LOGICAL
N	PRINT
P	PACKED
Q	BASIC
R	MICRO
S	CUSTOM
T	PCASCII
U	UNICODE
V	VAXFLOAT
X	NUMERIC
Y	UNISYS

Letter	Analytics Data type
Z	ZONED

IMPORT XML command

Creates an Analytics table by defining and importing an XML file.

Syntax

```
IMPORT XML TO table import_filename FROM source_filename [field_syntax] <...n>
```

```
field_syntax ::=
FIELD name type AT start_position DEC value WID bytes PIC format AS display_
name RULE xpath_expression
```

Parameters

Name	Description
TO *table*	The name of the Analytics table to import the data into. **Note** Table names are limited to 64 alphanumeric characters. The name can include the underscore character (_), but no other special characters, or any spaces. The name cannot start with a number.
import_filename	The name of the Analytics data file to create. Specify *import_filename* as a quoted string with a .FIL file extension. For example, `"Invoices.FIL"`. By default, the data file (.FIL) is saved to the folder containing the Analytics project. Use either an absolute or relative file path to save the data file to a different, existing folder: ◦ `"C:\data\Invoices.FIL"` ◦ `"data\Invoices.FIL"`
FROM *source_filename*	The name of the source data file. *source_filename* must be a quoted string. If the source data file is not located in the same directory as the Analytics project, you must use an absolute path or a relative path to specify the file location: ◦ `"C:\data\source_filename"` ◦ `"data\source_filename"`

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>FIELD name type</td><td>The individual fields to import from the source data file, including the name and data type of the field. To exclude a field from being imported, do not specify it.
For information about type, see "Identifiers for field data types" on the next page.</td></tr>
<tr><td>AT start_position</td><td>The starting byte position of the field in the Analytics data file.
Note<table><tr><td>non-Unicode Analytics</td><td>1 byte = 1 character</td></tr><tr><td>Unicode Analytics</td><td>2 bytes = 1 character</td></tr></table>In Unicode Analytics, typically you should specify an odd-numbered starting byte position. Specifying an even-numbered starting position can cause characters to display incorrectly.</td></tr>
<tr><td>DEC value</td><td>The number of decimals for numeric fields.</td></tr>
<tr><td>WID bytes</td><td>The length in bytes of the field in the Analytics table layout.
Note<table><tr><td>non-Unicode Analytics</td><td>1 byte = 1 character</td></tr><tr><td>Unicode Analytics</td><td>2 bytes = 1 character</td></tr></table>In Unicode Analytics, specify an even number of bytes only. Specifying an odd number of bytes can cause characters to display incorrectly.</td></tr>
<tr><td>PIC format</td><td>Note
Applies to numeric or datetime fields only.
◦ numeric fields - the display format of numeric values in Analytics views and reports
◦ datetime fields - the physical format of datetime values in the source data (order of date and time characters, separators, and so on)
Note
For datetime fields, format must exactly match the physical format in the source data. For example, if the source data is 12/31/2014, you must enter the format as <code>"MM/DD/YYYY"</code>.
format must be enclosed in quotation marks.</td></tr>
<tr><td>AS display_name</td><td>The display name (alternate column title) for the field in the view in the new Analytics table.
Specify display_name as a quoted string. Use a semi-colon (;) between words if you want a line break in the column title.</td></tr>
</table>

Name	Description
	AS is required when you are defining FIELD. To make the display name the same as the field name, enter a blank *display_name* value using the following syntax: `AS ""`. Make sure there is no space between the two double quotation marks.
RULE *xpath_expression*	The XPath expression used to select the field contents from the XML file. XPath is a standard way of accessing data from XML files. For example, `acct/title/text()` retrieves the text within the `<title>` tag in the XML file.

Examples

Importing data from an XML file to an Analytics table

You import data from an XML file to an Analytics table named **Employees**:

```
IMPORT XML TO Employees "Employees.fil" FROM "emp.XML" FIELD "Empno" C
AT 1 DEC 0 WID 6 PIC "" AS "" RULE "/RECORDS/RECORD/Empno/text()" FIELD
"First" C AT 7 DEC 0 WID 13 PIC "" AS "" RULE "/RECORDS/RECORD/First/-
text()" FIELD "Last" C AT 20 DEC 0 WID 20 PIC "" AS "" RULE
"/RECORDS/RECORD/Last/text()" FIELD "HireDate" D AT 40 DEC 0 WID 10 PIC
"YYYY-MM-DD" AS "" RULE "/RECORDS/RECORD/HireDate/text()" FIELD "Salary"
N AT 50 DEC 2 WID 8 PIC "" AS "" RULE "/RECORDS/RECORD/Salary/text()"
```

Remarks

For more information about how this command works, see "Import an XML file" on page 326.

Identifiers for field data types

The table below lists the letters that you must use when specifying *type* for `FIELD`. Each letter corresponds to an Analytics data type.

For example, if you are defining a Last Name field, which requires a character data type, you would specify "C": `FIELD "Last_Name" C`.

For more information, see "Data types in Analytics" on page 739.

> **Note**
>
> When you use the **Data Definition Wizard** to define a table that includes EBCDIC, Unicode, or ASCII fields, the fields are automatically assigned the letter "C" (for the CHARACTER type).
>
> When you enter an IMPORT statement manually, or edit an existing IMPORT statement, you can substitute the more specific letters "E" or "U" for EBCDIC or Unicode fields.

Letter	Analytics Data type
A	ACL
B	BINARY
C	CHARACTER
D	DATETIME
E	EBCDIC
F	FLOAT
G	ACCPAC
I	IBMFLOAT
K	UNSIGNED
L	LOGICAL
N	PRINT
P	PACKED
Q	BASIC
R	MICRO
S	CUSTOM
T	PCASCII
U	UNICODE
V	VAXFLOAT
X	NUMERIC
Y	UNISYS

Letter	Analytics Data type
Z	ZONED

INDEX command

Creates an index for an Analytics table that allows access to the records in a sequential order rather than a physical order.

Syntax

```
INDEX {<ON> key_field <D> <...n>|<ON> ALL <EXCLUDE field_name <...n>>} TO
file_name <IF test> <WHILE test> <FIRST range|NEXT range> <OPEN> <ISOLOCALE
locale_code>
```

Parameters

Name	Description
ON *key_field* D <...*n*> | ON ALL	The key field or fields, or the expression, to use for indexing. You can index by any type of field, including computed fields and ad hoc expressions, regardless of data type. ◦ **ON *key_field*** - use the specified field or fields If you index by more than one field, you create nested indexing in the table. The order of nesting follows the order in which you specify the fields. Include D to index the key field in descending order. The default index order is ascending. ◦ **ON ALL** - use all fields in the table If you index by all the fields in a table you create nested indexing. The order of nesting follows the order in which the fields appear in the table layout. An ascending index order is the only option for ON ALL.
EXCLUDE *field_name* optional	Only valid when indexing using ON ALL. The field or fields to exclude from the command. EXCLUDE allows you to fine-tune ON ALL, by excluding the specified fields. EXCLUDE must immediately follow ON ALL. For example: `ON ALL EXCLUDE field_1 field_2`
TO *file_name*	The name of the index and the associated index file. The index file is created with

Name	Description
	an .INX extension. **Note** In the Analytics user interface, index names are limited to 64 alphanumeric characters. The name can include the underscore character (_), but no other special characters, or any spaces. The name cannot start with a number.
IF *test* optional	A conditional expression that must be true in order to process each record. The command is executed on only those records that satisfy the condition. **Note** The IF parameter is evaluated against only the records remaining in a table after any scope parameters have been applied (WHILE, FIRST, NEXT).
WHILE *test* optional	A conditional expression that must be true in order to process each record. The command is executed until the condition evaluates as false, or the end of the table is reached. **Note** If you use WHILE in conjunction with FIRST or NEXT, record processing stops as soon as one limit is reached.
FIRST *range* | NEXT *range* optional	The number of records to process: ◦ **FIRST** - start processing from the first record until the specified number of records is reached ◦ **NEXT** - start processing from the currently selected record until the specified number of records is reached Use *range* to specify the number of records to process. If you omit FIRST and NEXT, all records are processed by default.
OPEN optional	Open the table and apply the index to the table.
ISOLOCALE *locale_code* optional	**Note** Applicable in the Unicode edition of Analytics only. The system locale in the format *language_country*. For example, to use Canadian French, enter `fr_ca`. Use the following codes: ◦ **language** - ISO 639 standard language code ◦ **country** - ISO 3166 standard country code If you do not specify a country code, the default country for the language is used. If you do not use ISOLOCALE, the default system locale is used.

Examples

Create an index and open the table

In the Vendor table, you create an index on the **Vendor City** field and open the table:

```
OPEN Vendor
INDEX ON Vendor_City to "CityIndex" OPEN
```

Create an index and apply it to a table

In the Vendor table, you create an index on the **Vendor City** field. Later, you apply the index to the table:

```
OPEN Vendor
INDEX ON Vendor_City to "CityIndex"
.
.
.
SET INDEX TO "CityIndex"
```

Remarks

For more information about how this command works, see "Indexing records" on page 1135.

The sort sequence used by the INDEX command

The INDEX command uses whatever sort sequence is specified in the **Sort Order** option (**Tools > Options > Table**). The default sort sequences are shown below.

For detailed information, see "The Sort Order option and sort sequences " on page 1123.

Analytics Edition	Sort Order default	Associated sort sequence
non-Unicode	System Default	Numbers, then uppercase, then lowercase:

Analytics Edition	Sort Order default	Associated sort sequence
	(ASCII)	`0, 1, 2... A, B, C... a, b, c...` For example, “Z” sorts before “a”.
Unicode	Mix Languages (UCA) (Unicode collation algorithm)	Numbers, then lowercase and uppercase intermixed: `0, 1, 2... a, A, b, B, c, C...` For example, “a” sorts before “Z”.

Case sensitivity

INDEX is case sensitive. Depending on which edition of Analytics you are using (non-Unicode or Unicode), casing in strings may affect indexing.

You can use the UPPER( ) function in conjunction with INDEX if you do not want case to affect indexing:

```
INDEX ON UPPER(key_field) TO "Index_file"
```

JOIN command

Combines fields from two Analytics tables into a new, single Analytics table.

> **Note**
> To use fuzzy matching to join tables, see "FUZZYJOIN command" on page 1729.

Syntax

```
JOIN {PKEY primary_key_fields|PKEY ALL <EXCLUDE field_name <...n>>} {FIELDS
primary_fields|FIELDS ALL <EXCLUDE field_name <...n>>} {SKEY secondary_key_
fields|SKEY ALL <EXCLUDE field_name <...n>>} <WITH secondary_fields|WITH ALL
<EXCLUDE field_name <...n>>> {no_
keyword|MANY|UNMATCHED|PRIMARY|SECONDARY|PRIMARY SECONDARY} <IF test> TO
table_name <LOCAL> <OPEN> <WHILE test> <FIRST range|NEXT range> <APPEND>
<PRESORT> <SECSORT> <ISOLOCALE locale_code>
```

Parameters

Name	Description
PKEY *primary_key_fields* | PKEY ALL	The key field or fields, or expression, in the primary table. ◦ **PKEY *primary_key_fields*** - use the specified field or fields Fields are used in the order that you list them. ◦ **PKEY ALL** - use all fields in the table Fields are used in the order that they appear in the table layout.
EXCLUDE *field_name* optional	Only valid when performing a join using PKEY ALL. The field or fields to exclude from the command. EXCLUDE allows you to fine-tune PKEY ALL, by excluding the specified fields. EXCLUDE must immediately follow PKEY ALL. For example: `PKEY ALL EXCLUDE field_1 field_2`
FIELDS *primary_fields* | FIELDS ALL	The fields or expressions from the primary table to include in the joined output table. ◦ **FIELDS *primary_fields*** - include the specified field or fields

Name	Description
	Fields are included in the order that you list them. ◦ **FIELDS ALL** – include all fields from the table Fields are included in the order that they appear in the table layout. **Note** You must explicitly specify the primary key field or fields if you want to include them in the joined table. Specifying FIELDS ALL also includes them.
EXCLUDE *primary_fields* optional	Only valid when performing a join using FIELDS ALL. The field or fields to exclude from the command. EXCLUDE allows you to fine-tune FIELDS ALL, by excluding the specified fields. EXCLUDE must immediately follow FIELDS ALL. For example: `FIELDS ALL EXCLUDE field_1 field_2`
SKEY *secondary_key_fields* | SKEY ALL	The key field or fields, or expression, in the secondary table. ◦ **SKEY *secondary_key_fields*** – use the specified field or fields Fields are used in the order that you list them. ◦ **SKEY ALL** – use all fields in the table Fields are used in the order that they appear in the table layout.
EXCLUDE *field_name* optional	Only valid when performing a join using SKEY ALL. The field or fields to exclude from the command. EXCLUDE allows you to fine-tune SKEY ALL, by excluding the specified fields. EXCLUDE must immediately follow SKEY ALL. For example: `SKEY ALL EXCLUDE field_1 field_2`
WITH *secondary_fields* | WITH ALL optional	The fields or expressions from the secondary table to include in the joined output table. ◦ **WITH *secondary_fields*** – include the specified field or fields Fields are included in the order that you list them. ◦ **WITH ALL** – include all fields from the table Fields are included in the order that they appear in the table layout. **Note** You must explicitly specify the secondary key field or fields if you want to include them in the joined table. Specifying WITH ALL also includes them. You cannot specify WITH if you are using the UNMATCHED join type.

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>EXCLUDE field_name

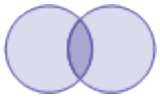
optional</td><td>Only valid when performing a join using WITH ALL.

The field or fields to exclude from the command. EXCLUDE allows you to fine-tune WITH ALL, by excluding the specified fields.

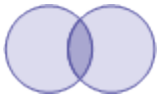
EXCLUDE must immediately follow WITH ALL. For example:

<code>WITH ALL EXCLUDE field_1 field_2</code></td></tr>
<tr><td>no_keyword | MANY | UNMATCHED | PRIMARY | SECONDARY | PRIMARY SECONDARY</td><td>The type of join to perform.

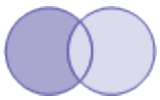
no_keyword (omit all join-type keywords)

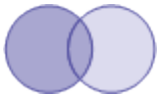
<table><tr><th>The joined output table contains:</th><th>Corresponding option in the Join dialog box</th></tr><tr><td>◦ all matched primary records and the first matched secondary record</td><td>Matched primary and secondary (1st secondary match)</td></tr></table>
MANY

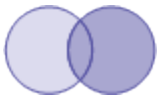
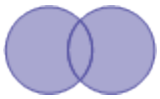
 

<table><tr><th>The joined output table contains:</th><th>Corresponding option in the Join dialog box</th></tr><tr><td>◦ all matched primary records and all matched secondary records
◦ one record for each match between the primary and secondary tables</td><td>Matched primary and secondary (all secondary matches)</td></tr></table>
UNMATCHED

<table><tr><th>The joined output table contains:</th><th>Corresponding option in the Join dialog box</th></tr><tr><td>◦ unmatched primary records</td><td>Unmatched primary</td></tr></table>
PRIMARY

  </td></tr>
</table>

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>The joined output table contains:</th><th>Corresponding option in the Join dialog box</th></tr>
<tr><td>◦ all primary records (matched and unmatched) and the first matched secondary record</td><td>All primary and matched secondary</td></tr>
</table>
<blockquote>Note
The keyword BOTH is the same as specifying PRIMARY.</blockquote>
SECONDARY

<table>
<tr><th>The joined output table contains:</th><th>Corresponding option in the Join dialog box</th></tr>
<tr><td>◦ all secondary records (matched and unmatched) and all matched primary records
Only the first instance of any duplicate secondary matches is joined to a primary record.</td><td>All secondary and matched primary</td></tr>
</table>
PRIMARY SECONDARY

<table>
<tr><th>The joined output table contains:</th><th>Corresponding option in the Join dialog box</th></tr>
<tr><td>◦ all primary and all secondary records, matched and unmatched
Only the first instance of any duplicate secondary matches is joined to a primary record.</td><td>All primary and secondary</td></tr>
</table>
</td></tr>
<tr><td>IF test
optional</td><td>A conditional expression that must be true in order to process each record. The command is executed on only those records that satisfy the condition.<blockquote>Note
The IF parameter is evaluated against only the records remaining in a table after any scope parameters have been applied (WHILE, FIRST, NEXT).</blockquote></td></tr>
</table>

Name	Description
	Note For most join types, an IF condition applies only to the primary table. The one exception is a many-to-many join, in which the IF condition can also reference the secondary table. To reference the secondary table you must specify a fully qualified field name (*table_name.field_name*). For example: `IF Customer.State="NY"`
TO *table_name*	The location to send the results of the command to: ◦ ***table_name*** - saves the results to an Analytics table Specify *table_name* as a quoted string with a .FIL file extension. For example: `TO "Output.FIL"` By default, the table data file (.FIL) is saved to the folder containing the Analytics project. Use either an absolute or relative file path to save the data file to a different, existing folder: • `TO "C:\Output.FIL"` • `TO "Results\Output.FIL"` **Note** Table names are limited to 64 alphanumeric characters, not including the .FIL extension. The name can include the underscore character (_), but no other special characters, or any spaces. The name cannot start with a number.
LOCAL optional	Saves the output file in the same location as the Analytics project. **Note** Applicable only when running the command against a server table with an output file that is an Analytics table. The LOCAL parameter must immediately follow the TO parameter.
OPEN optional	Opens the table created by the command after the command executes. Only valid if the command creates an output table.
WHILE *test* optional	A conditional expression that must be true in order to process each record. The command is executed until the condition evaluates as false, or the end of the table is reached. **Note** If you use WHILE in conjunction with FIRST or NEXT, record processing stops as soon as one limit is reached.
FIRST *range* | NEXT *range*	The number of records to process: ◦ **FIRST** - start processing from the first record until the specified number of

Name	Description
optional	records is reached ◦ **NEXT** - start processing from the currently selected record until the specified number of records is reached Use *range* to specify the number of records to process. If you omit FIRST and NEXT, all records are processed by default.
APPEND optional	Appends the command output to the end of an existing file instead of overwriting it. **Note** You must ensure that the structure of the command output and the existing file are identical: • the same fields • the same field order • matching fields are the same length • matching fields are the same data type Analytics appends output to an existing file regardless of its structure. If the structure of the output and the existing file do not match, jumbled, missing, or inaccurate data can result.
PRESORT optional	Sorts the primary table on the primary key field before executing the command. **Note** You cannot use PRESORT inside the GROUP command. ## Indexing instead of sorting The primary table can be indexed instead of sorted. With large tables, indexing instead of sorting may reduce the time required to join the tables. If you are joining two tables using an indexed common key field, omit PRESORT and SECSORT.
SECSORT optional	Sorts the secondary table on the secondary key field before executing the command. **Note** You cannot use SECSORT inside the GROUP command. ## Indexing instead of sorting The secondary table can be indexed instead of sorted. With large tables, indexing instead of sorting may reduce the time required to join the tables. If you are joining two tables using an indexed common key field, omit PRESORT and SECSORT.
ISOLOCALE *locale_code* optional	**Note** Applicable in the Unicode edition of Analytics only.

Name	Description
	The system locale in the format *language_country*. For example, to use Canadian French, enter `fr_ca`. Use the following codes: ◦ **language** - ISO 639 standard language code ◦ **country** - ISO 3166 standard country code If you do not specify a country code, the default country for the language is used. If you do not use ISOLOCALE, the default system locale is used.

Examples

Join two tables as a way of discovering employees who may also be vendors

The examples below join the Empmast and Vendor tables using address as the common key field (the Address and Vendor_Street fields).

The JOIN command creates a new table with matched primary and secondary records, which results in a list of any employees and vendors with the same address.

```
OPEN Empmast PRIMARY
OPEN Vendor SECONDARY
JOIN PKEY Address FIELDS Empno First Last Address SKEY Vendor_Street
WITH Vendor_No Vendor_Name Vendor_Street TO "Employee_Vendor_Match" OPEN
PRESORT SECSORT
```

This version of the JOIN command includes all fields from the primary and secondary tables in the joined output table.

```
OPEN Empmast PRIMARY
OPEN Vendor SECONDARY
JOIN PKEY Address FIELDS ALL SKEY Vendor_Street WITH ALL TO "Employee_
Vendor_Match" OPEN PRESORT SECSORT
```

This version of the JOIN command uses an IF condition to restrict the joined output table to employees and vendors with addresses in California.

Note that the join type is `MANY`, which is required if you want an IF condition to reference a secondary table. The name of the field in the secondary table must be fully qualified (`Vendor.Vendor_State`).

```
OPEN Empmast PRIMARY
OPEN Vendor SECONDARY
JOIN PKEY Address FIELDS ALL SKEY Vendor_Street WITH ALL IF State="CA"
AND Vendor.Vendor_State="CA" TO "Employee_Vendor_Match" OPEN PRESORT
MANY SECSORT
```

Join two tables as a way of discovering accounts receivable records with no matching customer

The example below joins the Ar and Customer tables using Customer Number (CustNo) as the common key field.

The JOIN command uses the UNMATCHED join type to create a new table with unmatched primary records, which results in a list of Ar records that are not associated with any Customer record.

```
OPEN Ar PRIMARY
OPEN Customer SECONDARY
JOIN PKEY CustNo FIELDS CustNo Due Amount SKEY CustNo UNMATCHED TO "Cus-
tomerNotFound.fil" OPEN PRESORT SECSORT
```

Remarks

For more information about how this command works, see "Joining tables" on page 889.

LIST command

Outputs the data in one or more fields in an Analytics table to a display formatted in columns.

Syntax

```
LIST {FIELDS field_name <AS display_name> <...n>|FIELDS ALL} <LINE number
field_list> <TO {SCREEN|filename|PRINT}> <UNFORMATTED> <IF test> <WHILE test>
<FIRST range|NEXT range> <HEADER header_text> <FOOTER footer_text> <SKIP
lines> <EOF> <APPEND>
```

Parameters

Name	Description
FIELDS *field_name* <...*n*> | FIELDS ALL	The fields to include in the output. ◦ **FIELDS *field_name*** - include the specified field or fields Fields are included in the order that you list them. ◦ **FIELDS ALL** - include all fields in the table Fields are included in the order that they appear in the table layout.
AS *display_name* optional	Only used when listing data using FIELDS *field_name*. The display name (alternate column title) for the field in the output. If you want the display name to be the same as the field name, or an existing display name in the source table, do not use AS. Specify *display_name* as a quoted string. Use a semi-colon (;) between words if you want a line break in the column title.
LINE *number field_list* optional	More than one line is used in the output for each record: ◦ ***number*** - the line number, must be between 2 and 60 inclusive ◦ ***field_list*** - the fields to include on that line
TO SCREEN | *filename* | PRINT optional	The location to send the results of the command to: ◦ **SCREEN** - displays the results in the Analytics display area **Tip** You can click any linked result value in the display area to drill down to the associated record or records in the source table. ◦ ***filename*** - saves the results to a file

Name	Description
	Specify *filename* as a quoted string with the appropriate file extension. For example: `TO "Output.TXT"` By default, the file is saved to the folder containing the Analytics project. Use either an absolute or relative file path to save the file to a different, existing folder: • `TO "C:\Output.TXT"` • `TO "Results\Output.TXT"` ◦ **PRINT** - sends the results to the default printer
UNFORMATTED optional	The output is displayed as unformatted text. Output is identical to that created by the EXPORT ASCII command. Unformatted data can be output to a file for further processing by other software programs.
IF *test* optional	A conditional expression that must be true in order to process each record. The command is executed on only those records that satisfy the condition. **Note** The IF parameter is evaluated against only the records remaining in a table after any scope parameters have been applied (WHILE, FIRST, NEXT).
WHILE *test* optional	A conditional expression that must be true in order to process each record. The command is executed until the condition evaluates as false, or the end of the table is reached. **Note** If you use WHILE in conjunction with FIRST or NEXT, record processing stops as soon as one limit is reached.
FIRST *range* | NEXT *range* optional	The number of records to process: ◦ **FIRST** - start processing from the first record until the specified number of records is reached ◦ **NEXT** - start processing from the currently selected record until the specified number of records is reached Use *range* to specify the number of records to process. If you omit FIRST and NEXT, all records are processed by default.
HEADER *header_text* optional	The text to insert at the top of each page of a report. *header_text* must be specified as a quoted string. The value overrides the Analytics HEADER system variable.
FOOTER *footer_text* optional	The text to insert at the bottom of each page of a report. *footer_text* must be specified as a quoted string. The value overrides the Analytics FOOTER system variable.
SKIP *lines* optional	Inserts the specified number of blank lines between each record in the list. For example, `LIST ALL SKIP 1` produces a double spaced list (one blank line between each record).

Name	Description
EOF optional	Execute the command one more time after the end of the file has been reached. This ensures that the final record in the table is processed when inside a GROUP command. Only use EOF if all fields are computed fields referring to earlier records.
APPEND optional	Appends the command output to the end of an existing file instead of overwriting it.

Examples

Listing exceptions and saving to a text file

You use LIST to create a report listing exceptions identified in an inventory table. The report is saved as a text file:

```
LIST Product_number Description Quantity Unit_cost Value IF Quantity < 0
OR Unit_cost < 0 HEADER "Negative Values" TO "Exceptions.txt"
```

Remarks

When to use LIST

Use LIST to print data, display data on screen, or export it to a text file.

Formatting and totals

Unless you specify UNFORMATTED, the following information is included automatically:

- page numbers
- date
- time
- user identification
- column headings

Numeric columns are also automatically totaled.

LOCATE command

Searches for the first record that matches the specified value or condition, or moves to the specified record number.

Syntax

```
LOCATE {IF test <WHILE test> <FIRST range|NEXT range>|RECORD num}
```

Parameters

Name	Description
IF *test*	The value or condition to search for. You must enclose character literal values in quotation marks, and datetime values in backquotes.
WHILE *test* optional	A conditional expression that must be true in order to process each record. The command is executed until the condition evaluates as false, or the end of the table is reached. **Note** If you use WHILE in conjunction with FIRST or NEXT, record processing stops as soon as one limit is reached.
FIRST *range* | NEXT *range* optional	The number of records to process: ◦ **FIRST** - start processing from the first record until the specified number of records is reached ◦ **NEXT** - start processing from the currently selected record until the specified number of records is reached Use *range* to specify the number of records to process. If you omit FIRST and NEXT, all records are processed by default.
RECORD *num*	The record number to locate.

Examples

Locate the first record that matches a specified value

The following examples illustrate using LOCATE to find the first occurrence of a specific value in a table:

```
LOCATE IF Vendor_Name = "United Equipment"
```

```
LOCATE IF Vendor_Name = "Uni"
```

```
LOCATE IF Invoice_Amount > 1000
```

```
LOCATE IF Invoice_Date = `20141231`
```

Locate the first record that matches a specified condition or expression

The following examples illustrate using LOCATE to find the first occurrence of a specific condition or expression in a table:

```
LOCATE IF Vendor_Name = "United Equipment" AND Invoice_Amount > 1000 AND
Invoice_Date > `20140930`
```

```
LOCATE IF Vendor_City = v_city
```

Locate a record by record number

The following example illustrates using LOCATE to move to a particular record in a table:

```
LOCATE RECORD 50
```

Remarks

For more information about how this command works, see "Selecting the first matching record" on page 1167.

How it works

Use the LOCATE command to move directly to the first record in a table matching the specified value or condition.

If the specified value or condition is found, the first matching record in the table is selected. If the specified value or condition is not found, the table is positioned at the first record.

You can also use LOCATE to move directly to a specific record number

LOCATE compared to FIND and SEEK

Unlike the FIND and SEEK commands, the LOCATE command is not restricted to searching an indexed table, or a single character field. Using LOCATE, you can search for any type of literal, or for an expression that uses any data type, or mix of data types.

When used to search an unindexed table, the LOCATE command may be significantly slower than FIND or SEEK because it must process each record in the table sequentially. The required processing time depends on the size of the table, the location of a matching record, and whether you reduce the scope of the search by using WHILE, FIRST, or NEXT.

Partial matching supported

Partial matching is supported for character searches. The search value can be contained by a longer value in the field or fields being searched. However, search values must appear at the start of fields to constitute a match.

Enabling or disabling partial matching

You can enable or disable partial matching using either the SET command, or a setting in the **Options** dialog box:

Enable partial matching	Disable partial matching
Specify: `SET EXACT OFF` or **Deselect: Exact Character Comparisons** in the **Options** dialog box (**Tools > Options > Table**) **Result:** The search value can be contained by a longer value in the field or fields being searched. The search value must appear at the start of a field to constitute a match.	**Specify:** `SET EXACT ON` or **Select: Exact Character Comparisons** in the **Options** dialog box (**Tools > Options > Table**) **Result:** The search value must exactly match a value in a field to constitute a match.

For more information about SET EXACT, see "SET command" on page 1960.

For more information about the **Exact Character Comparisons** option, see "Table options" on page 122.

LOOP command

Executes a series of ACLScript commands repeatedly on a record while a specified condition evaluates to true.

> **Note**
> The LOOP command must be enclosed inside the GROUP command.

Syntax

```
LOOP WHILE test
  command
  <...n>
END
```

Parameters

Name	Description
WHILE *test*	The test that must evaluate to true for the commands inside the LOOP command to be executed. If the test evaluates to true the commands are executed repeatedly until the test evaluates to false.
command <...*n*>	One or more commands to execute. You can enter multiple commands inside the LOOP command. Each command must start on a new line.
END	The end of the LOOP command.

Examples

Splitting a comma-delimited field

You have a table containing invoice data and you need to isolate specific information for invoice amounts per department. One invoice may be related to more than one department,

and department codes are stored in comma-delimited format in the table.

To extract the invoice amounts per department, you:

1. Use a GROUP command to process the table record by record.
2. Calculate the number of departments (n) associated with each record.
3. Use the LOOP command to iterate n times over the record to extract data for each department associated with the record.

```
COMMENT
use GROUP to count commas in each department code field as a way of
identifying how many departments are associated with the record
"LOOP" over each record for each code in the field, with each iteration
of the loop extracting the record with a single code to the result1
table
END
GROUP
  v_department_count = OCCURS(Dept_Code,',')
  v_counter = 0
  LOOP WHILE v_counter <= v_department_count
    v_dept = SPLIT(Dept_Code, ',', (v_counter + 1))
    EXTRACT FIELDS Invoice_Number, Amount, v_dept AS "Department" TO res-
ult1
    v_counter = v_counter + 1
  END
END
```

Remarks

> **Tip**
> For a detailed tutorial covering the LOOP and GROUP commands, see "Grouping and looping" on page 1416.

When to use LOOP

Loops are frequently used when a record contains repeated segments of data that you want to process.

How it works

Each LOOP command must specify a WHILE condition to test, and be closed with an END statement. The commands between LOOP and END are executed repeatedly for the current record as long as

the specified test is true.

If the test is initially false, the commands are not executed.

Avoiding infinite loops

To avoid creating an infinite loop, make sure that the test you specify eventually returns false. You can also use the SET LOOP command to prevent infinite looping.

MERGE command

Combines records from two sorted Analytics tables with an identical structure into a new Analytics table that uses the same sort order as the original tables.

Syntax

```
MERGE {{ON key_fields|ON ALL <EXCLUDE field_name <...n>>}|{PKEY primary_key_
fields|PKEY ALL <EXCLUDE field_name <...n>>} {SKEY secondary_key_
fields|SKEY ALL <EXCLUDE field_name <...n>>}} <IF test> TO table_name <LOCAL>
<OPEN> <WHILE test> <FIRST range|NEXT range> <APPEND> <PRESORT> <ISOLOCALE loc-
ale_code>
```

> **Note**
>
> Only character fields, or character computed fields, can be used as key fields in MERGE.
>
> The key fields in the primary and secondary tables must both be sorted in ascending order. If one or both key fields are unsorted, or sorted in descending order, the MERGE command fails.
>
> You can use PRESORT to sort the primary key field. If the secondary key field is unsorted, you must first sort it in a separate sort operation before performing the merge.
>
> The primary and secondary tables can be indexed instead of sorted. With large tables, indexing instead of sorting may reduce the time required to merge the tables.

Parameters

Name	Description
ON *key_fields* | ON ALL	**Note** You can only use ON if the corresponding key fields in the primary and the secondary tables have the same name. If the corresponding fields have different names, or if they are expressions rather than actual physical fields, you must use PKEY and SKEY. The key field or fields in both the primary and the secondary tables. ◦ **ON *key_fields*** - use the specified field or fields

Name	Description
	Fields are used in the order that you list them. ◦ **ON ALL** - use all fields in the table Fields are used in the order that they appear in the table layout.
EXCLUDE *field_name* optional	Only valid when merging using ON ALL. The field or fields to exclude from the command. EXCLUDE allows you to fine-tune ON ALL, by excluding the specified fields. EXCLUDE must immediately follow ON ALL. For example: `ON ALL EXCLUDE field_1 field_2`
PKEY *primary_key_fields* | PKEY ALL	The key field or fields, or expression, in the primary table. ◦ **PKEY *primary_key_fields*** - use the specified field or fields Fields are used in the order that you list them. ◦ **PKEY ALL** - use all fields in the table Fields are used in the order that they appear in the table layout.
EXCLUDE *field_name* optional	Only valid when merging using PKEY ALL. The field or fields to exclude from the command. EXCLUDE allows you to fine-tune PKEY ALL, by excluding the specified fields. EXCLUDE must immediately follow PKEY ALL. For example: `PKEY ALL EXCLUDE field_1 field_2`
SKEY *secondary_key_fields* | SKEY ALL	The key field or fields, or expression, in the secondary table. ◦ **SKEY *secondary_key_fields*** - use the specified field or fields Fields are used in the order that you list them. ◦ **SKEY ALL** - use all fields in the table Fields are used in the order that they appear in the table layout.
EXCLUDE *field_name* optional	Only valid when merging using SKEY ALL. The field or fields to exclude from the command. EXCLUDE allows you to fine-tune SKEY ALL, by excluding the specified fields. EXCLUDE must immediately follow SKEY ALL. For example: `SKEY ALL EXCLUDE field_1 field_2`
IF *test* optional	A conditional expression that must be true in order to process each record. The command is executed on only those records that satisfy the condition.

Name	Description
	Note The IF parameter is evaluated against only the records remaining in a table after any scope parameters have been applied (WHILE, FIRST, NEXT).
TO *table_name*	The location to send the results of the command to: ◦ ***table_name*** - saves the results to an Analytics table Specify *table_name* as a quoted string with a .FIL file extension. For example: `TO "Output.FIL"` By default, the table data file (.FIL) is saved to the folder containing the Analytics project. Use either an absolute or relative file path to save the data file to a different, existing folder: • `TO "C:\Output.FIL"` • `TO "Results\Output.FIL"` **Note** Table names are limited to 64 alphanumeric characters, not including the .FIL extension. The name can include the underscore character (_), but no other special characters, or any spaces. The name cannot start with a number.
LOCAL optional	Saves the output file in the same location as the Analytics project. **Note** Applicable only when running the command against a server table with an output file that is an Analytics table. The LOCAL parameter must immediately follow the TO parameter.
OPEN optional	Opens the table created by the command after the command executes. Only valid if the command creates an output table.
WHILE *test* optional	A conditional expression that must be true in order to process each record. The command is executed until the condition evaluates as false, or the end of the table is reached. **Note** If you use WHILE in conjunction with FIRST or NEXT, record processing stops as soon as one limit is reached.
FIRST *range* | NEXT *range* optional	The number of records to process: ◦ **FIRST** - start processing from the first record until the specified number of records is reached ◦ **NEXT** - start processing from the currently selected record until the specified number of records is reached Use *range* to specify the number of records to process. If you omit FIRST and NEXT, all records are processed by default.

Name	Description
APPEND optional	Appends the command output to the end of an existing file instead of overwriting it. **Note** You must ensure that the structure of the command output and the existing file are identical: • the same fields • the same field order • matching fields are the same length • matching fields are the same data type Analytics appends output to an existing file regardless of its structure. If the structure of the output and the existing file do not match, jumbled, missing, or inaccurate data can result.
PRESORT optional	Sorts the primary table on the primary key field before executing the command. **Note** You cannot use PRESORT inside the GROUP command. Omit PRESORT: ◦ If the primary key field is already sorted ◦ If you are merging two tables using an indexed common key field
ISOLOCALE *locale_code* optional	**Note** Applicable in the Unicode edition of Analytics only. The system locale in the format *language_country*. For example, to use Canadian French, enter `fr_ca`. Use the following codes: ◦ **language** - ISO 639 standard language code ◦ **country** - ISO 3166 standard country code If you do not specify a country code, the default country for the language is used. If you do not use ISOLOCALE, the default system locale is used.

Examples

Merge tables with identical key field names

The following example merges two tables with identical key field names:

```
OPEN Employees_Location_1 PRIMARY
OPEN Employees_Location_2 SECONDARY
MERGE ON Last_Name TO "AllEmployees" PRESORT
```

Merge tables with different key field names

The following example merges two tables with different key field names:

```
OPEN Employees_Location_1 PRIMARY
OPEN Employees_Location_2 SECONDARY
MERGE PKEY Last_Name SKEY Surname TO "AllEmployees" PRESORT
```

Remarks

For more information about how this command works, see "Merging tables" on page 881.

Alternatives to merging

Merging can be tricky to perform correctly. You can get the same result by appending, or by extracting and appending, and then sorting.

For more information, see "APPEND command" on page 1565, and "EXTRACT command" on page 1712.

If the two source tables are already sorted, merging is more efficient and can execute more quickly.

NOTES command

Creates, modifies, or removes a note associated with an individual record in an Analytics table.

Syntax

```
NOTES <IF test> <TEXT note_text> <APPEND> <CLEAR>
```

Parameters

Name	Description
IF *test* optional	A conditional expression that must be true in order to process each record. The command is executed on only those records that satisfy the condition. **Note** The IF parameter is evaluated against only the records remaining in a table after any scope parameters have been applied (WHILE, FIRST, NEXT). ◦ If you do not specify an IF test, the note text is added to each record in the table ◦ If you specify an IF test and CLEAR, the notes for those records that meet the condition are deleted
TEXT *note_text* optional	The text to add as a note. *note_text* must be either a string enclosed in quotation marks, or a character expression.
APPEND optional	The note text is added to the end of any existing notes. If omitted, any existing notes are overwritten.
CLEAR optional	Notes are deleted. Even if all record notes in a table are deleted, the auto-generated **RecordNote** field is not deleted from the table layout.

Examples

Adding the same note to multiple records

Any existing notes for the specified records are overwritten:

```
NOTES IF MATCH(RECNO(),1,3,5,7) TEXT "note text"
```

Adding or append the same note to multiple records

Any existing notes for the specified records have the new note text appended:

```
NOTES IF MATCH(RECNO(),1,3,5,7) TEXT "note text" APPEND
```

Deleting notes from multiple records

All record notes in the table are deleted:

```
NOTES CLEAR
```

Notes for the specified records are deleted:

```
NOTES IF MATCH(RECNO(),1,3,5,7) CLEAR
```

Notes for records 1-100 are deleted:

```
NOTES IF RECNO() <= 100 CLEAR
```

Remarks

Deleting the RecordNote field

To delete the **RecordNote** field from the table layout, and all notes in the table, use the `DELETE NOTES` command without any of the options specified.

NOTIFY command

Sends an email notification message.

Syntax

```
NOTIFY USER username <PASSWORD pwd> MAILBOX pathname ADDRESS recipient <CC cc_recipient> <BCC bcc_recipient> <SUBJECT subject> MESSAGE message <ATTACHMENT pathname>
```

Parameters

Name	Description
USER *username*	The email address of the sender.
PASSWORD *pwd* optional	The password for the mail server.
MAILBOX *pathname*	The SMTP server name to use to send the email message. For example: `MAILBOX "mailserver.example.com"`
ADDRESS *recipient*	The email address of one or more recipients. Separate multiple email addresses with a comma. Enter a maximum of 1020 characters.
CC *cc_recipient* optional	The email address of one or more carbon copy recipients. Separate multiple email addresses with a comma. Enter a maximum of 1000 characters.
BCC *bcc_recipient* optional	The email address of one or more blind carbon copy recipients. Separate multiple email addresses with a comma.
SUBJECT *subject* optional	The subject line of the email message.
MESSAGE *message*	The body text of the email message. The message is plain text and does not support HTML. If you want to insert a line break in your message, use two carat characters: `^^`.

Name	Description
ATTACHMENT *pathname* optional	The path and filename of one or more attachments. Must be a quoted string. Specify multiple attachments by entering a comma separated list of files for *pathname*: `ATTACHMENT "result1,result2"`

Examples

Sending an error report email

You are running a script, and you want to send a notification email if the script fails. Using NOTIFY, you define the email message and include two attachments:

- the log file
- a .fil file containing recorded errors

```
NOTIFY USER "support@company.com" MAILBOX "mail.company.com" ADDRESS
"script_admin@example.com" SUBJECT "Error Report" MESSAGE "Failed to pro-
cess script. Details attached." ATTACHMENT "Errors.fil,ACL_Demo.log"
```

Remarks

Recipients and attachments

You can use the NOTIFY command to send email notification messages to one or more recipients. Messages can include attached data files and Analytics projects.

The NOTIFY command can be used to notify the appropriate personnel when a script fails unexpectedly.

Protocols and ports

The command can be used with any mail server that supports SMTP (Simple Mail Transfer Protocol), which is used by Microsoft Exchange and many other mail servers. The NOTIFY command can also be used with older email applications, from Microsoft and others, that send mail locally.

NOTIFY uses port 25, so this port must be open on the mail server or the command fails. The port number used by the command is not configurable. If NOTIFY fails with an error message, contact your IT department to find out if port 25 is blocked on your network.

Error handling

If Analytics is unable to connect with the mail server, it makes five additional attempts to connect, with a 10-second pause between each attempt. If all connection attempts are unsuccessful, the NOTIFY command is canceled, with a message written to the log, but the script continues processing.

You can use the SET command to change this default behavior. You can specify a different number of connection attempts and a different amount of time between attempts, or you can turn off additional connection attempts. You can also specify that Analytics stops processing a script if the NOTIFY command is canceled. For more information, see "SET command" on page 1960.

An invalid email recipient is not considered a failure of the NOTIFY command and does not cause a script to stop regardless of the associated setting.

OPEN command

Opens an Analytics table and the associated data file.

Syntax

```
OPEN {table_name|data_file <FORMAT layout_name>} <BUFFERLENGTH length> <CRLF>
<DBASE> <INDEX index_file> <PRIMARY|SECONDARY> <SKIP bytes> <RELATION key_
field>
```

Parameters

Name	Description
table_name	The name of the Analytics table to open.
data_file	The data file to associate with the table specified by FORMAT *layout_name*. Analytics assumes a file extension of .fil if no extension is specified. To open a file with no extension, insert a period (.) at the end of the file name.
FORMAT *layout_name* optional	The Analytics table layout to apply to the data file that you open as a table.
BUFFERLENGTH *n* optional	The length in bytes of the input buffer area to be allocated to the table. The default value is 33,000 bytes. Larger buffer areas may improve processing speed at the expense of RAM available for storing Analytics commands. If any IBM variable length blocks are read which exceed the buffer length, Analytics displays an error message and stops processing. The default value is set in the **Buffer Size** field in the **Table** tab in the **Options** dialog box. You will seldom need to change BUFFERLENGTH *n*, because the default is sufficient to handle almost all situations.
CRLF optional	Specifies that a variable length ASCII file is to be read. Analytics automatically adjusts for the varying record lengths. By default, files are assumed to be fixed-length files.
DBASE optional	Specifies that the data source is a dBASE file. Analytics recognizes the type of dBASE file and automatically creates a table from the file description. Can be omitted for dBASE files with a .dbf extension.

Name	Description
INDEX *index_file* optional	The index file to apply to the table when it is opened. The file extension for the index file name is assumed to be .inx when none is specified. You can specify INDEX with either primary or secondary tables.
PRIMARY | SECONDARY optional	Specifies that a table is opened as either a primary table or a secondary table. If omitted, the table is opened as a primary table.
SKIP *bytes* optional	The number of bytes to bypass at the physical start of the table. SKIP can be used to ignore table header records or leading portions of the table that do not follow the layout of the remainder of the table. If omitted, the table is read starting at the first byte. **Note** non-Unicode Analytics: 1 byte = 1 character Unicode Analytics, extended ASCII (ANSI) data: 1 byte = 1 character Unicode Analytics, Unicode data: 2 bytes = 1 character For Unicode data, specify an even number of bytes only. Specifying an odd number of bytes can cause characters to display incorrectly.
RELATION *key_field* optional	Specifies that the table is to be opened as an ad hoc related table. Analytics does not retain this relation when the table is closed. You must also specify the INDEX parameter when you use RELATION. *key_field* is the key field or expression used to create the relation between two tables.

Examples

Opening a table while specifying a table layout

You open the **April_2012** table using the **March_2012** table layout:

```
OPEN April_2012 FORMAT March_2012
```

Opening a dBASE file

You open a dBASE file named `Inventory.dbf` for which there is no existing table:

```
OPEN Inventory
```

Opening a table and apply a pre-existing index

To open either a primary or a secondary table, and apply an index that already exists for the table, use the following syntax:

```
OPEN Accounts_receivable INDEX Customer_number_AR
```

```
OPEN Customer SECONDARY INDEX Customer_number
```

Opening a table and establish an ad hoc relation to another table

You need to establish a temporary relation between an open table named **Customers** (the primary table) and a table named **Accounts_receivable** (the secondary table).

You use an index named **Customer_index** and a key field in the primary table named **Last_name**:

```
OPEN Accounts_receivable INDEX Customer_index RELATION Last_name
```

OUTLIERS command

Identifies statistical outliers in a numeric field. Outliers can be identified for the field as a whole, or for separate groups based on identical values in one or more character, numeric, or datetime key fields.

Syntax

```
OUTLIERS {AVERAGE|MEDIAN} {PKEY key_field <...n>|PKEY ALL <EXCLUDE field_name
<...n>>|NOKEY} ON numeric_field <OTHER field <...n>|OTHER ALL <EXCLUDE field_
name <...n>>> NUMSTDEV number_of_std_devs <IF test> <TO {SCREEN|table_name}>
<PRESORT> <WHILE test> <FIRST range|NEXT range> <OPEN>
```

> **Note**
> You cannot run the OUTLIERS command locally against a server table.
> You must specify the OUTLIERS command name in full. You cannot abbreviate it.

Parameters

Name	Description
AVERAGE | MEDIAN	The method for calculating the center point of the values in *numeric_field* (the outlier field). ◦ **AVERAGE** - calculate the average (mean) of the values ◦ **MEDIAN** - calculate the median of the values The center point is calculated for either: ◦ the numeric field as a whole ◦ the numeric values for each key field group The center point is subsequently used in calculating the standard deviation of the numeric field, or of each group. **Note** If you specify MEDIAN, *numeric_field* must be sorted. Use PRESORT if *numeric_field* is not already sorted. **Tip** If the data you are examining for outliers is significantly skewed, MEDIAN might produce results that are more representative of the bulk of the data.

Name	Description
PKEY *key_field <...n>* | PKEY ALL | NOKEY	One or more character, numeric, or datetime fields to use for grouping the data in the table. If you specify NOKEY, the data is not grouped, and outliers are identified at the field level. **Note** Key fields must be sorted. Use PRESORT if one or more fields are not already sorted. ◦ **PKEY *key_field*** - use the specified field or fields to group the data in the table Multiple fields must be separated by spaces, and can be different data types. If you specify more than one field, you created nested groups in the output table. The nesting follows the order in which you specify the fields. For each group, a standard deviation is calculated for the group's numeric values in *numeric_field*. The group standard deviation is used as the basis for identifying group outliers. ◦ **PKEY ALL** - use all fields in the table to group the data in the table If you specify all fields, you create nested groups in the output table. The nesting follows the order in which the fields appear in the table layout. For each group, a standard deviation is calculated for the group's numeric values in *numeric_field*. The group standard deviation is used as the basis for identifying group outliers. **Note** Grouping by all fields includes *numeric_field*, which may not make sense. You can use EXCLUDE to exclude *numeric_field* from grouping. ◦ **NOKEY** - do not group the data in the table A standard deviation is calculated for *numeric_field* as a whole. The field standard deviation is used as the basis for identifying field outliers.
EXCLUDE *field_name* optional	Only valid when grouping table data using PKEY ALL. The field or fields to exclude from the command. EXCLUDE allows you to fine-tune PKEY ALL, by excluding the specified fields. EXCLUDE must immediately follow PKEY ALL. For example: `PKEY ALL EXCLUDE field_1 field_2`
ON *numeric_field*	The numeric field to examine for outliers. You can examine only one field at a time. Outliers are values that fall outside the upper and lower boundaries established by the field or group standard deviation, or by a specified multiple of the standard deviation.
OTHER *field <...n>* |	One or more additional fields to include in the output.

Name	Description
OTHER ALL optional	◦ **OTHER *field <...n>*** - include the specified field or fields Fields are included in the order that you list them. ◦ **OTHER ALL** - include all fields in the table that are not specified as key fields or the outlier field Fields are included in the order that they appear in the table layout. **Note** Key fields and the outlier field are automatically included in the output table, and do not need to be specified using OTHER.
EXCLUDE *field_name* optional	Only valid when using OTHER ALL. The field or fields to exclude from the command. EXCLUDE allows you to fine-tune OTHER ALL, by excluding the specified fields. EXCLUDE must immediately follow OTHER ALL. For example: `OTHER ALL EXCLUDE field_1 field_2`

NUMSTDEV *number_of_std_devs*	In *numeric_field*, the number of standard deviations from the mean or the median to the upper and lower outlier boundaries. You can specify any positive integer or decimal numeral (0.5, 1, 1.5, 2 . . .)
	The formula for creating outlier boundaries is:
	`mean/median ± (number_of_std_devs * standard deviation)`
	Note
	Standard deviation is a measure of the dispersion of a data set - that is, how spread out the values are. The outliers calculation uses population standard deviation.
	Example of outlier boundaries
	`NUMSTDEV 2`
	establishes, for *numeric_field* as a whole, or for each key field group:
	• an upper outlier boundary 2 standard deviations greater than the mean or the median
	`mean/median + (2 * SD)`
	• a lower outlier boundary 2 standard deviations less than the mean or the median
	`mean/median – (2 * SD)`

Name	Description
	Any value greater than the upper boundary, or less than the lower boundary, is included as an outlier in the output results. **Note** For the same set of data, as you increase the value in *number_of_std_devs* you potentially decrease the number of outliers returned.
IF *test* optional	A conditional expression that must be true in order to process each record. The command is executed on only those records that satisfy the condition. **Note** The IF parameter is evaluated against only the records remaining in a table after any scope parameters have been applied (WHILE, FIRST, NEXT).
TO SCREEN | *table_name* optional	The location to send the results of the command to: ◦ **SCREEN** - displays the results in the Analytics display area **Tip** You can click any linked result value in the display area to drill down to the associated record or records in the source table. ◦ ***table_name*** - saves the results to an Analytics table Specify *table_name* as a quoted string with a .FIL file extension. For example: `TO "Output.FIL"` By default, the table data file (.FIL) is saved to the folder containing the Analytics project. Use either an absolute or relative file path to save the data file to a different, existing folder: • `TO "C:\Output.FIL"` • `TO "Results\Output.FIL"` **Note** Table names are limited to 64 alphanumeric characters, not including the .FIL extension. The name can include the underscore character (_), but no other special characters, or any spaces. The name cannot start with a number.
PRESORT optional	Performs a sorting operation before executing the command. **Tip** If the appropriate field or fields in the input table are already sorted, you can save processing time by not specifying PRESORT.

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>If you specify PRESORT and:</th><th>Sorts by:</th></tr>
<tr><td>PKEY, AVERAGE</td><td>

key field or fields
key field or fields, then by numeric_field (if numeric_field is computed)

Note
Sorting a computed numeric_field is an internal, technical requirement of Analytics.
</td></tr>
<tr><td>PKEY, MEDIAN</td><td>key field or fields, then by numeric_field</td></tr>
<tr><td>NOKEY, AVERAGE</td><td>no sorting</td></tr>
<tr><td>NOKEY, MEDIAN</td><td>numeric_field</td></tr>
</table>
</td></tr>
<tr><td>WHILE test
optional</td><td>A conditional expression that must be true in order to process each record. The command is executed until the condition evaluates as false, or the end of the table is reached.

Note
If you use WHILE in conjunction with FIRST or NEXT, record processing stops as soon as one limit is reached.</td></tr>
<tr><td>FIRST range | NEXT range
optional</td><td>The number of records to process:
FIRST - start processing from the first record until the specified number of records is reached
NEXT - start processing from the currently selected record until the specified number of records is reached
Use range to specify the number of records to process.
If you omit FIRST and NEXT, all records are processed by default.</td></tr>
<tr><td>OPEN
optional</td><td>Opens the table created by the command after the command executes. Only valid if the command creates an output table.</td></tr>
</table>

Examples

Identifying transaction amounts that are out of the ordinary

You want to identify transaction amounts that are out of the ordinary across the entire **Ar** table in `Sample Project.acl`.

You decide to set the outlier boundaries at 3 times the standard deviation of the **Amount** field. The test returns 16 outliers in the table of 772 records.

```
OPEN Ar
OUTLIERS AVERAGE NOKEY ON Amount NUMSTDEV 3 PRESORT TO "Outliers_AR.fil"
OPEN
```

You repeat the test, but increase the standard deviation multiple to 3.5. The test now returns only 6 outliers because the outlier boundaries are farther away from the center point of the values in the **Amount** field.

```
OPEN Ar
OUTLIERS AVERAGE NOKEY ON Amount NUMSTDEV 3.5 PRESORT TO "Outliers_AR.-
fil" OPEN
```

Identifying transaction amounts that are out of the ordinary for each customer

For each customer in the **Ar** table in `Sample Project.acl`, you want to identify any transaction amounts that are out of the ordinary.

You decide to set the outlier boundaries at 3 times the standard deviation of each customer's group of transactions.

```
OPEN Ar
OUTLIERS AVERAGE PKEY No ON Amount NUMSTDEV 3 PRESORT TO "Outliers_Cus-
tomer_AR.fil" OPEN
```

The test returns 7 outliers. The standard deviation and the average are reported for each customer's group of transactions:

	Cust Number (No)	Trans Amount	STDEV	AVERAGE	Group Number
1	065003	4,954.64	1015.58	833.83	1
2	**262001**	3,567.34	772.44	438.81	2

	Cust Number (No)	Trans Amount	STDEV	AVERAGE	Group Number
3	**262001**	(2,044.82)	772.44	438.81	2
4	376005	(931.55)	411.18	484.57	3
5	501657	5,549.19	1332.80	441.14	4
6	811002	3,409.82	634.20	672.10	5
7	925007	3,393.87	736.48	906.16	6

How outliers are identified for customer 262001

Customer 262001 has 101 transactions in the **Ar** table, and two of them are reported as outliers because they exceed the outlier boundaries for that customer:

Outlier	Lower boundary	Upper boundary	Outlier
(2,044.82)	(1,878.51)	2,756.13	3,567.34

How outlier boundaries are calculated for customer 262001

The outlier boundaries are the average of all customer 262001 transactions, plus or minus the specified multiple of the standard deviation of the transactions:

Average of all customer 262001 transactions	438.81
Specified multiple of the standard deviation	3
Standard deviation of the transactions	772.44
	438.81 ± (3 * 772.44) = 438.81 ± 2,317.32 = **(1,878.51) (lower boundary)** = **2,756.13 (upper boundary)**

Using MEDIAN to identify transaction amounts that are out of the ordinary for each customer

You use MEDIAN, instead of AVERAGE, to perform the same outlier test that you performed in the example above.

```
OPEN Ar
OUTLIERS MEDIAN PKEY No ON Amount NUMSTDEV 3 PRESORT TO "Outliers_Cus-
tomer_AR_Median.fil" OPEN
```

The test returns 10 outliers instead of the 7 that are returned in the previous test. Depending on the nature of the data, MEDIAN and AVERAGE can return somewhat different results:

	Cust Number (No)	Trans Amount	STDEV	MEDIAN	Group Number
1	065003	4,954.64	1015.58	663.68	1
2	262001	(2,044.82)	772.44	450.67	2
3	262001	3,567.34	772.44	450.67	2
4	376005	(931.55)	411.18	517.16	3
5	501657	4,426.14	1332.80	146.80	4
6	501657	5,549.19	1332.80	146.80	4
7	811002	3,409.82	634.20	624.53	5
8	925007	2,972.78	736.48	717.88	6
9	925007	3,030.71	736.48	717.88	6
10	925007	3,393.87	736.48	717.88	6

How outlier boundaries are calculated for each customer

The outlier boundaries are the median value of each customer's transactions, plus or minus the specified multiple of the standard deviation of the transactions.

For example, for customer 262001: 450.67 ± (3 * 772.44)

Remarks

For more information about how this command works, see "Identifying outliers" on page 1098.

Add outlier boundary fields to the results table

Analytics automatically adds the **STDEV** and **AVERAGE** or **MEDIAN** calculated fields to the outliers results table. You may find it useful to also add two computed fields that show the outliers boundaries used to identify the outliers in the results table.

1. Open the outliers results table.
2. Paste this expression into the Analytics command line, edit it as required, and press Enter:

   ```
   DEFINE FIELD Lower_Boundary COMPUTED AVERAGE - (number_of_std_devs *
   STDEV)
   ```

   - For *number_of_std_devs*, substitute the actual standard deviation multiple you used.
   - If you used median as a center point rather than average, substitute MEDIAN for AVERAGE.
3. Paste this expression into the Analytics command line, edit it as required, and press Enter:

   ```
   DEFINE FIELD Upper_Boundary COMPUTED AVERAGE + (number_of_std_devs *
   STDEV)
   ```

   - For *number_of_std_devs*, substitute the actual standard deviation multiple you used.
   - If you used median as a center point rather than average, substitute MEDIAN for AVERAGE.
4. Right-click the view and select **Add Columns**.
5. From the **Available Fields** list, double-click **Lower_Boundary** and **Upper_Boundary** to add them to the **Selected Fields** list.
6. Click **OK**.
7. Optional. Reposition the added fields by dragging the column headers.

PASSWORD command

Creates a password definition, without a password value, that prompts users for a password while a script is running.

Syntax

```
PASSWORD num <prompt>
```

Parameters

Name	Description
num	A value from 1 to 10 that uniquely identifies the password definition.
prompt optional	A valid character expression to display in the dialog box used to prompt for the password. Enclose literal strings in quotation marks. If *prompt* is omitted, a default dialog box with no message is displayed.

Examples

Prompting for password information

You use the PASSWORD command to prompt the user for the three passwords required in a script. Once the user enters the required passwords, the script can complete the remaining processing without interruption:

```
PASSWORD 1 "Enter the password for the Receivables database"
PASSWORD 2 "Enter the password for the Payables database"
PASSWORD 3 "Enter the password for the Customer database"
```

Specifying a password when refreshing an Analytics table

You combine the PASSWORD command with the REFRESH command to update a password-protected data file:

```
PASSWORD 1 "Password:"
REFRESH Abc PASSWORD 1
```

Specifying a password to define a server table

You use the PASSWORD command with the DEFINE TABLE DB command to define a server table via AX Connector, which requires one password for the database profile, and another for the associated server profile:

```
DEFINE TABLE DB SOURCE Inventory_DBProfile PASSWORD 9 PASSWORD 3
```

Remarks

When to use PASSWORD

Use the PASSWORD command to prompt a user to enter password information before a script accesses, imports, or refreshes password-protected data.

You can create up to ten different password definitions in a script .

PASSWORD is useful if:

- you want to avoid typing an actual password in a script, which the SET PASSWORD command requires
- individual users need to enter distinct passwords

How passwords are stored

User-entered passwords are temporarily and securely stored in memory.

When a user types a password into the prompt dialog box, the characters are masked using asterisks (*). The password does not appear in either the script or the log.

Storing passwords for server-based analytics

The PASSWORD command is not supported in analytics run in Robots or on AX Server, or in legacy server scripts.

You can use the PASSWORD tag to prompt for a password when a user schedules an analytic in Robots or on AX Server.

You can use the SET PASSWORD command to specify passwords in legacy server scripts.

PAUSE command

Pauses a script, and displays information in a dialog box for users.

Syntax

```
PAUSE message <IF test>
```

Parameters

Name	Description
message	A message to display in the dialog box. The maximum length is 199 characters. *message* must be enclosed in quotation marks. If the message contains double quotation marks, enclose it in single quotation marks.
IF *test* optional	A conditional expression that must be true in order to process each record. The command is executed on only those records that satisfy the condition. **Note** The IF parameter is evaluated against only the records remaining in a table after any scope parameters have been applied (WHILE, FIRST, NEXT).

Examples

Displaying an error message

You require user input to meet specific requirements. When you detect that the input does not meet those requirements, you use the PAUSE command and display an error message in a dialog box:

```
PAUSE "The product class must be a 2-digit value."
```

Remarks

When to use PAUSE

Use PAUSE to display read-only messages on screen in the course of running a script. You can display error messages or information such as the result of an analytic operation.

How it works

While the message dialog box is displayed, execution of the script is halted and only resumes once the user clicks **OK** to close the message dialog box. For this reason, you cannot use PAUSE in scripts or analytics that must run unattended.

Limitations

PAUSE has the following limitations:

- cannot be included inside the GROUP command
- cannot be used in analytics run in Robots, or on AX Server

PREDICT command

Applies a predictive model to an unlabeled data set to predict classes or numeric values associated with individual records.

> **Note**
> The PREDICT command is not supported if you are running Analytics on a 32-bit computer. The computation required by the command is processor-intensive and better suited to 64-bit computers.

Syntax

```
PREDICT MODEL model_name TO table_name <IF test> <WHILE test> <FIRST
range|NEXT range>
```

Parameters

Name	Description
MODEL *model_name*	The name of the model file to use for predicting classes or values. You use a model file previously generated by the TRAIN command. You must specify the `*.model` file extension. For example: `MODEL "Loan_default_prediction.model"` **Note** The model file must have been trained on a data set with the same fields as the unlabeled data set – or substantially the same fields. You cannot use a model file trained in version 14.1 of Analytics. Version 14.1 model files are not compatible with subsequent versions of Analytics. Train a new predictive model to use with the PREDICT command.
TO *table_name*	The name of the Analytics table output by the prediction process. The table contains the key fields you specified during the training process, and either one or two fields generated by the prediction process: ◦ **Predicted** – the predicted classes or numeric values associated with each record

Name	Description
	in the unlabeled data set ◦ **Probability** - (classification only) the probability that a predicted class is accurate Specify *table_name* as a quoted string with a .FIL file extension. For example: `TO "Loan_applicants_default_predicted.FIL"` By default, the table data file (.FIL) is saved to the folder containing the Analytics project. Use either an absolute or relative file path to save the data file to a different, existing folder: ◦ `TO "C:\Loan_applicants_default_predicted.FIL"` ◦ `TO "ML Predict output\Loan_applicants_default_predicted.FIL"` **Note** Table names are limited to 64 alphanumeric characters, not including the .FIL extension. The name can include the underscore character (_), but no other special characters, or any spaces. The name cannot start with a number.
IF *test* optional	A conditional expression that must be true in order to process each record. The command is executed on only those records that satisfy the condition. **Note** The IF parameter is evaluated against only the records remaining in a table after any scope parameters have been applied (WHILE, FIRST, NEXT).
WHILE *test* optional	A conditional expression that must be true in order to process each record. The command is executed until the condition evaluates as false, or the end of the table is reached. **Note** If you use WHILE in conjunction with FIRST or NEXT, record processing stops as soon as one limit is reached.
FIRST *range* | NEXT *range* optional	The number of records to process: ◦ **FIRST** - start processing from the first record until the specified number of records is reached ◦ **NEXT** - start processing from the currently selected record until the specified number of records is reached Use *range* to specify the number of records to process. If you omit FIRST and NEXT, all records are processed by default.

Examples

Use a classification model to make predictions

You input a classification model to the PREDICT command to make predictions about which current loan applicants will default if given a loan.

You previously produced the classification model using the TRAIN command with a set of historical loan data, including loan default information.

```
OPEN "Loan_applicants_current"
PREDICT MODEL "Loan_default_prediction.model" TO "Loan_applicants_
default_predicted.FIL"
```

Use a regression model to make predictions

You input a regression model to the PREDICT command to make predictions about the future sale price of houses.

You previously produced the regression model using the TRAIN command with a set of recent house sales data, including the sale price.

```
OPEN "House_price_evaluation"
PREDICT MODEL "House_price_prediction.model" TO "House_prices_pre-
dicted.FIL"
```

Remarks

For more information about how this command works, see "Predicting classes and numeric values" on page 1292.

PRINT command

Prints a text file, an Analytics log file, or an Analytics project item that has been exported as an external file – a script (.aclscript), a table layout (.layout), or a workspace (.wsp). You can also print a graph that has been generated by a command.

Syntax

```
PRINT {file_name|GRAPH}
```

Parameters

Name	Description
file_name | GRAPH	The item to print: ◦ ***file_name*** – the relative or absolute path and file name of the file to print For example, `"C:\ACL Data\Sample Data Files\ACL_Demo.log"` or `"Sample Data Files\ACL_Demo.log"`. If the path or file name includes spaces you must enclose *file_name* in quotation marks. ◦ **GRAPH** – the graph previously output as the result of a command

Examples

Printing a log file

To print the log file for the `ACL_Demo.acl` project, specify the following command:

```
PRINT "C:\ACL Data\Sample Data Files\ACL_Demo.log"
```

Printing a graph

To print the graph produced from the BENFORD command, specify the following commands:

```
OPEN Metaphor_APTrans_2002
BENFORD ON Invoice_Amount LEADING 1 TO GRAPH
PRINT GRAPH
```

Remarks

Selecting a printer

The printer used is the default printer configured in Microsoft Windows. To change the printer you need to change the default printer in Windows.

Related commands

To print the contents of an Analytics table in a project, use the DO REPORT command.

PROFILE command

Generates summary statistics for one or more numeric fields, or numeric expressions, in an Analytics table.

Syntax

```
PROFILE {<FIELDS> numeric_field <...n>|<FIELDS> ALL <EXCLUDE numeric_field
<...n>>} <IF test> <WHILE test> <FIRST range|NEXT range>
```

Parameters

Name	Description
FIELDS *numeric_field* <...*n*> | FIELDS ALL	Specify individual fields to profile, or specify ALL to profile all numeric fields in the Analytics table.
EXCLUDE *numeric_field* optional	Only valid when profiling using FIELDS ALL. The field or fields to exclude from the command. EXCLUDE allows you to fine-tune FIELDS ALL, by excluding the specified fields. EXCLUDE must immediately follow FIELDS ALL. For example: `FIELDS ALL EXCLUDE field_1 field_2`
IF *test* optional	A conditional expression that must be true in order to process each record. The command is executed on only those records that satisfy the condition. **Note** The IF parameter is evaluated against only the records remaining in a table after any scope parameters have been applied (WHILE, FIRST, NEXT).
WHILE *test* optional	A conditional expression that must be true in order to process each record. The command is executed until the condition evaluates as false, or the end of the table is reached.

Name	Description
	Note If you use WHILE in conjunction with FIRST or NEXT, record processing stops as soon as one limit is reached.
FIRST *range* | NEXT *range* optional	The number of records to process: ◦ **FIRST** - start processing from the first record until the specified number of records is reached ◦ **NEXT** - start processing from the currently selected record until the specified number of records is reached Use *range* to specify the number of records to process. If you omit FIRST and NEXT, all records are processed by default.

Examples

Profiling a single field

You profile the **Salary** field:

```
OPEN Employee_Payroll
PROFILE FIELDS Salary
```

The command generates the following output:

Field Name	Total Value	Absolute Value	Minimum	Maximum
SALARY	1,152,525	1,152,525	15,340	52,750

Remarks

Statistics displayed in output

The following statistics are displayed for each numeric field or numeric expression specified for the command:

- total value
- absolute value

- minimum value
- maximum value

QUIT command

Ends the current session and closes Analytics.

Syntax

```
QUIT
```

Examples

Check if a file exists and close Analytics if it does not

You have created a script for others to run, but if a required file does not exist, you want to close Analytics.

The example below checks if the required `Inventory.csv` file exists, and closes Analytics if it does not:

```
IF FILESIZE("Inventory.csv") = -1 QUIT
```

Automatically close Analytics after a script completes

The script below summarizes the Inventory table, and produces output results, then automatically closes Analytics:

```
OPEN Inventory
SUMMARIZE ON Location ProdCls SUBTOTAL Value TO "Inventory_value_by_location_class.FIL" PRESORT CPERCENT
QUIT
```

Remarks

Changes are saved

When QUIT executes, any Analytics tables that are open are saved and closed before quitting.

If you modified the active view or a script and did not save the changes, Analytics prompts you to save the changes before quitting.

RANDOM command

Generates a set of random numbers.

Syntax

```
RANDOM NUMBER n <SEED seed_value> MINIMUM min_value MAXIMUM max_value <COLUMNS
n> <UNIQUE> <SORTED> <TO {SCREEN|filename}> <APPEND>
```

Parameters

Name	Description
NUMBER *n*	The size of the set of random numbers to be generated. A maximum of 32767 numbers can be generated.
SEED *seed_value* optional	The value used to initialize the random number generator. If you specify a seed value, it can be any number. Each unique seed value results in a different set of random numbers. If you respecify the same seed value, the same set of random numbers is generated. Regenerating the same set of random numbers can be required if you need to replicate analysis. ◦ **Seed value** - explicitly specify a seed value, and save the value, if you want the option of replicating a particular set of random numbers. ◦ **No seed value** - enter a seed value of '0', or leave the seed value blank, if you want Analytics to randomly select a seed value.
MINIMUM *min_value*	The smallest possible number in the set of random numbers. Any valid numeric value or expression is allowed.
MAXIMUM *max_value*	The greatest possible number in the set of random numbers. Any valid numeric value or expression is allowed.
COLUMNS *n* optional	The number of columns used to display the set of random numbers. If you omit COLUMNS, the default is 6 columns.
UNIQUE optional	Include only unique numbers in the set of random numbers. If you omit UNIQUE, duplicate values are allowed in the set of random numbers.

Name	Description
	Note Do not specify UNIQUE when the specified size of the set of random numbers exceeds 75 percent of the range between MINIMUM and MAXIMUM. Doing so can result in too many random number selections being discarded.
SORTED optional	Displays the set of random numbers in ascending order. If you omit SORTED, the numbers are displayed in the order in which they are randomly selected.
TO SCREEN | *filename* optional	The location to send the results of the command to: ◦ **SCREEN** - displays the results in the Analytics display area **Tip** You can click any linked result value in the display area to drill down to the associated record or records in the source table. ◦ ***filename*** - saves the results to a file Specify *filename* as a quoted string with the appropriate file extension. For example: `TO "Output.TXT"` By default, the file is saved to the folder containing the Analytics project. Use either an absolute or relative file path to save the file to a different, existing folder: • `TO "C:\Output.TXT"` • `TO "Results\Output.TXT"` If you omit TO, the set of random numbers is output to screen.
APPEND optional	Appends the command output to the end of an existing file instead of overwriting it. **Note** You must ensure that the structure of the command output and the existing file are identical: • the same fields • the same field order • matching fields are the same length • matching fields are the same data type Analytics appends output to an existing file regardless of its structure. If the structure of the output and the existing file do not match, jumbled, missing, or inaccurate data can result.

Examples

Generate a text file with 100 random numbers

You want to pull 100 hard copy files at random from a set of files with numbering that ranges from 10,000 to 20,000.

You can use the RANDOM command to generate a text file with 100 random numbers between 10,000 and 20,000. You then pull the hard copy files that match the random numbers. The numbers are arranged in 10 columns, are unique, and are sorted in ascending order:

```
RANDOM NUMBER 100 SEED 45387 MINIMUM 10000 MAXIMUM 20000 COLUMNS 10
UNIQUE SORTED TO "Random_Numbers.txt"
```

Remarks

For more information about how this command works, see "Generating random numbers" on page 219.

Random number algorithm

The RANDOM command uses the default Analytics random number algorithm. Unlike the SAMPLE command, the RANDOM command does not have the option of using the Mersenne-Twister random number algorithm.

RCOMMAND command

Passes an Analytics table to an external R script as a **data frame** and creates a new table in the Analytics project using output from the external R script.

Syntax

```
RCOMMAND FIELDS field <...n> RSCRIPT path_to_script TO table_name <IF test>
<WHILE test> <FIRST range|NEXT range> <KEEPTITLE> <SEPARATOR character>
<QUALIFIER character> <OPEN>
```

Parameters

Name	Description
FIELDS *field_name* <...*n*>	The fields from the source Analytics table, or the expressions, to include in the data frame that is sent to the R script. Depending on the edition of Analytics that you are using, you may encounter errors when sending data containing some special characters to R: ◦ **non-Unicode** - "\" ◦ **Unicode** - "ÿ" or "Ŝ" ◦ **Both** - box drawing characters such as blocks, black squares, and vertical broken bars **Note** Mixed language data is also not supported, for example a table containing both Japanese and Chinese characters.
RSCRIPT *path_to_script*	The full or relative path to the R script on the file system. Enclose *path_to_script* in quotation marks.
TO *table_name*	The location to send the results of the command to: ◦ ***table_name*** - saves the results to an Analytics table Specify *table_name* as a quoted string with a .FIL file extension. For example: `TO "Output.FIL"` By default, the table data file (.FIL) is saved to the folder containing the Analytics project. Use either an absolute or relative file path to save the data file to a different,

Name	Description
	existing folder: • `TO "C:\Output.FIL"` • `TO "Results\Output.FIL"` **Note** Table names are limited to 64 alphanumeric characters, not including the .FIL extension. The name can include the underscore character (_), but no other special characters, or any spaces. The name cannot start with a number. The output table is created from the data frame or matrix that the R script returns.
IF *test* optional	A condition that must be met to process the current record. The data frame passed to the R script contains only those records that meet the condition.
WHILE *test* optional	A conditional expression that must be true in order to process each record. The command is executed until the condition evaluates as false, or the end of the table is reached. **Note** If you use WHILE in conjunction with FIRST or NEXT, record processing stops as soon as one limit is reached.
FIRST *range* | NEXT *range* optional	The number of records to process: ◦ **FIRST** - start processing from the first record until the specified number of records is reached ◦ **NEXT** - start processing from the currently selected record until the specified number of records is reached Use *range* to specify the number of records to process. If you omit FIRST and NEXT, all records are processed by default. **Caution** There is a known issue in the current version with NEXT when running the RCOMMAND. Avoid using this option as the record reference may reset to the first record regardless of which record is selected.
KEEPTITLE optional	Treat the first row of data as field names instead of data. If omitted, generic field names are used. This option is required if you want to retrieve data using column names in the R script.
SEPARATOR *character* optional	The character to use as the separator between fields. You must specify the character as a quoted string. The default character is a comma.

Name	Description
	Note Avoid using any characters that appear in the input fields. If the SEPARATOR character appears in the input data, the results may be affected.
QUALIFIER *character* optional	The character to use as the text qualifier to wrap and identify field values. You must specify the character as a quoted string. The default character is a double quotation mark. **Note** Avoid using any characters that appear in the input fields. If the QUALIFIER character appears in the input data, the results may be affected.
OPEN optional	Opens the table created by the command after the command executes. Only valid if the command creates an output table.

Examples

Getting R up and running (Hello world)

You create a hello world script to test your connection between Analytics and R:

Analytics command

```
RCOMMAND FIELDS "Hello", ", world!" TO "r_result" RSCRIPT "C:\scripts\r_
scripts\analysis.r"
```

R script (analysis.r)

```
srcTable<-acl.readData()

# create table to send back to ACL
output<-data.frame(
  c(srcTable[1,1]),
  c(srcTable[1,2])
)
```

```
# add column names and send table back to ACL
colnames(output) <- c("Greeting","Subject")
acl.output<-output
```

Accessing field data using row and column coordinates

You send a number of invoice fields to an R script for analysis outside Analytics:

Analytics command

```
RCOMMAND FIELDS Department_Code Invoice_Amount Invoice_Date Invoice_Num-
ber Vendor_Number TO "r_result" RSCRIPT "C:\scripts\r_scripts\ana-
lysis.r"
```

R script (analysis.r)

```
# Retrieves invoice number from second row of data frame in R script
srcTable<-acl.readData()
srcTable[2,4]
```

Accessing field data using column names

You send a number of invoice fields to an R script for analysis outside Analytics. You use the KEEPTITLE option so that columns can be referenced by name in R:

Analytics command

```
RCOMMAND FIELDS Department_Code Invoice_Amount Invoice_Number TO "r_res-
ult" RSCRIPT "C:\scripts\r_scripts\analysis.r" KEEPTITLE
```

R script (analysis.r)

```
# Retrieves invoice number from second row of data frame in R script
srcTable<-acl.readData()
srcTable["2","Invoice_Number"]
```

Sending invoice records that exceed 1000.00 value to R script

You send a number of invoice fields to an R script for analysis outside Analytics. You use IF to limit the records sent to R. Only those records with an invoice amount exceeding 1000.00 are sent:

Analytics command

```
RCOMMAND FIELDS Department_Code Invoice_Amount Invoice_Number TO "r_res-
ult" IF Invoice_Amount > 1000.00 RSCRIPT "C:\scripts\r_scripts\ana-
lysis.r" KEEPTITLE
```

R script (analysis.r)

```
# Retrieves invoice number from second row of data frame in R script
srcTable<-acl.readData()
srcTable["2","Invoice_Number"]
```

Sending invoice records and returns multiplied invoice amounts

You send a number of invoice fields to an R script for analysis outside Analytics. The R script performs a single action against every cell in the named column:

Analytics command

```
RCOMMAND FIELDS Department_Code Invoice_Amount Invoice_Number TO "r_res-
ult" RSCRIPT "C:\scripts\r_scripts\analysis.r" KEEPTITLE
```

R script (analysis.r)

```
# Returns slice of ACL table with value doubled
srcTable<-acl.readData()
acl.output<-srcTable["Invoice_Amount"] * 2
```

Remarks

For more information about how this command works, see "Running R scripts" on page 1322.

Referencing Analytics data in the R script

The Analytics table is passed to the script as an R **data frame**. Data frames are tabular data objects that may contain columns of different modes, or types, of data.

To work with the data frame created by Analytics in an R script, invoke the `acl.readData()` function and store the returned data frame in a variable:

```
# stores the Analytics table in a data frame called myTable that can be ref-
erenced throughout the script
myTable<-acl.readData()
```

To retrieve data from a cell in the data frame, you can use one of the following approaches:

- Using row and column coordinates:

  ```
  # Retrieves the value in the first row and second column of the data
  frame
  myTable[1,2]
  ```

  > **Note**
  > Coordinates are based on the order of fields specified in the command, not the table layout or view that is currently open.

- Using row and column names:

```
# Retrieves the value in the first row and "myColumnTitle" column of the
data frame
myTable["1","myColumnTitle"]
```

  You must specify the **KEEPTITLE** option of the command to use column names.

  Rows are named "1", "2", "3", and increment accordingly. You may also use a combination of names and coordinates.

Passing data back to Analytics

To return a data frame or matrix back to Analytics and create a new table, use the following syntax:

```
# Passes myNewTable data frame back to Analytics to create a new table
acl.output<-myNewTable
```

> **Note**
> You must return a data frame or a matrix to Analytics when the R script terminates. Ensure the columns in the data frame or matrix contain only atomic values and not lists, matrices, arrays, or non-atomic objects. If the values cannot be translated into Analytics data types, the command fails.

Data type mappings

Analytics data types are translated into R data types using a translation process between the Analytics project and the R script:

Analytics data type	R data type(s)
Logical	Logical
Numeric	Numeric
Character	Character
Datetime	Date, POSIXct, POSIXlt

Performance and file size limits

The time it takes to run your R script and process the data that is returned increases for input data exceeding 1 GB. R does not support input files of 2 GB or higher.

The number of records sent to R also affects performance. For two tables with the same file size but a differing record count, processing the table with fewer records is faster.

Handling multi-byte character data

If you are sending data to R in a multi-byte character set, such as Chinese, you must set the system locale appropriately in your R script. To successfully send a table of multi-byte data to R, the first line of the R script must contain the following function:

```
# Example that sets locale to Chinese
Sys.setlocale("LC_ALL","Chinese")
```

For more information about Sys.setlocale( ), see the R documentation.

R log file

Analytics logs R language messages to an `aclrlang.log` file in the project folder. Use this log file for debugging R errors.

> **Tip**
> The log file is available in the Results folder of Analytics Exchange analytic jobs.

Running R scripts on AX Server

If you are writing an analysis app to run on AX Server and you want to work with external R scripts:

1. Upload the file as a related file with the analysis app.
2. Use the `FILE` analytic tag to identify the file(s).
3. Reference the file(s) using the relative path `./filename.r`.

> **Note**
> Using a related file ensures that the TomEE application server account has sufficient permissions to access the file when running R with Analytics Exchange.

REFRESH command

Updates the data in an Analytics table from its associated data source.

Syntax

```
REFRESH <table_name> <PASSWORD num>
```

Parameters

Name	Description
table_name optional	The name of the Analytics table to refresh. If you do not specify a *table_name*, the open table is refreshed.
PASSWORD *num* optional	The password definition to use. You do not use PASSWORD *num* to prompt for, or specify, an actual password. The password definition refers to a password previously supplied or set using the PASSWORD command, the SET PASSWORD command, or the PASSWORD analytic tag. *num* is the number of the password definition. For example, if two passwords have been previously supplied or set in a script, or when scheduling an analytic, `PASSWORD 2` specifies that password #2 is used. For more information about supplying or setting passwords, see: ◦ "PASSWORD command" on page 1893 ◦ "SET command" on page 1960 ◦ "PASSWORD tag" on page 2530 **Note** The password is used to access the original source data system. You cannot use REFRESH with a password for file-based data sources, with the exception of PDFs.

Examples

Refreshing a table with no password required

If a password is not required for the data source, just specify the REFRESH command and the name of the Analytics table to refresh.

```
REFRESH Invoices
```

Refreshing a table with a password in an interactive script

If you are creating an interactive script, you can prompt the user for the password:

```
PASSWORD 1 "Enter your password:"
REFRESH Invoices PASSWORD 1
```

If you are refreshing a table originally imported from a password-protected data source using the ACCESSDATA command, the password prompt is automatic and does not need to be separately specified:

```
REFRESH Invoices
```

Refreshing a table with a password in a non-interactive script

You can set the password in a script if you do not want to prompt the user for the value:

```
SET PASSWORD 1 TO "password"
REFRESH Invoices PASSWORD 1
```

The disadvantage of this method is that the password appears as clear text in the script.

Refreshing a table with a password in an AX Server analytic

If you are creating an AX Server analytic, you can prompt the user for the password when the analytic is scheduled, or run ad-hoc:

```
COMMENT
//ANALYTIC Refresh Table
//PASSWORD 1 "Enter your password:"
END
REFRESH Invoices PASSWORD 1
```

Remarks

For more information about how this command works, see "Updating data in Analytics tables" on page 709.

How it works

The REFRESH command updates the contents of a table by re-running the IMPORT command, or the ACCESSDATA command, initially used to define and import the table.

REFRESH updates table content only

The REFRESH command updates only the content of existing fields in an Analytics table. It cannot update an Analytics table layout.

You cannot use REFRESH if the structure of the source data has changed – for example, if fields have been added or removed. You must re-import the data.

Data sources that support refreshing

You can use the REFRESH command to update the content of an Analytics table created using any of the following commands:

- IMPORT ACCESS
- IMPORT DELIMITED

- IMPORT EXCEL
- IMPORT ODBC (legacy ODBC command)
- IMPORT PDF
- IMPORT PRINT
- IMPORT SAP
- IMPORT XBRL
- IMPORT XML
- ACCESSDATA (ODBC data sources)

REFRESH and ACCESSDATA

The following guidelines apply when refreshing a table imported from an ODBC data source using the ACCESSDATA command.

- **Open table** – If the table is open when you refresh it, you temporarily need disk space equal to twice the size of the table. If you have limited disk space, close the table first before refreshing it.
- **Analytics 12** – Tables that were imported using the ACCESSDATA command in version 12 of Analytics are not refreshable, even if you are using a more recent version of Analytics.

  If you want to be able to refresh these tables, re-import them using Analytics 12.5 or later.

REFRESH and passwords

You can use the REFRESH command with password-protected data sources that exist in a database, or in a cloud data service.

You cannot use the REFRESH command with password-protected file-based data sources, such as Excel files. The one exception is password-protected PDFs.

REFRESH and the Analysis App window

Do not use the REFRESH command in scripts that you intend to run in the Analysis App window.

Depending on how a table is imported, refreshing the data in the table is either not supported, or produces unpredictable results, if attempted in the Analysis App window.

If you want to refresh data as part of a script run in the Analysis App window, use either an IMPORT command, or the ACCESSDATA command, and overwrite the table.

RENAME command

Renames an Analytics project item or a file.

Syntax

```
RENAME item_type name <AS|TO> new_name <OK>
```

Parameters

Name	Description
item_type name	The type and name of the project item or file that you want to rename. **Note** In most cases, you cannot rename an item or file if it is active, open, or in use. Specify one of the following valid types: ◦ **FIELD** - physical data field, computed field, or variable • The table containing the field must be open. However, the active view cannot include the field. • You cannot rename a field that is referenced by a computed field. ◦ **FORMAT** - Analytics table ◦ **INDEX** - index ◦ **REPORT** - report or view ◦ **WORKSPACE** - workspace ◦ **SCRIPT (or BATCH)** - script ◦ **DATA** - Analytics data file (.fil) ◦ **FILE** - data file in the file system ◦ **LOG** - Analytics log file (.log) ◦ **TEXT** - text file
AS | TO *new_name*	The new name for the project item or file. **Note** Length limitations apply to most Analytics project item names. For more information, see "Character and size limits in Analytics" on page 1357.
OK optional	Deletes or overwrites items without asking you to confirm the action.

Examples

Renaming a field

You need to rename the **ProdNo** field to **ProdNum**. You use OK to perform the action without additional confirmation:

```
OPEN Inventory
RENAME FIELD ProdNo AS ProdNum OK
```

REPORT command

Formats and generates a report based on the open Analytics table.

Syntax

```
REPORT <ON break_field <PAGE> <NODUPS> <WIDTH characters> <AS display_name>>
<...n> FIELD other_fields <WIDTH characters> <AS display_name> <...n>
<SUPPRESS> <NOZEROS> <LINE n other_fields> <PRESORT <sort_field>> <...n>
<SUMMARIZED> <SKIP n> <EOF> <TO {SCREEN|PRINT|filename <HTML>}> <IF test>
<WHILE test> <FIRST range|NEXT range> <HEADER header_text> <FOOTER footer_
text> <APPEND>
```

Parameters

Name	Description
ON *break_field* PAGE NODUPS WIDTH *characters* AS *display_name* <...n> optional	The character field or fields used to break the report into sections. A new report section and subtotal is created each time the value in *break_field* changes. Breaking reports into sections can make them easier to scan. ◦ ***break_field*** - the field to use as a break field To run a report based on a view (DO REPORT), the break field must be the leftmost character field in the view. ◦ **PAGE** - inserts a page break when the break field value changes ◦ **NODUPS** - suppresses duplicate display values in the break field For example, if the customer name is listed for each invoice record, you can make the report more readable by listing only the first instance of each customer name. ◦ **WIDTH *characters*** - the output length of the field in characters ◦ **AS *display_name*** - the display name (alternate column title) for the field in the report Specify *display_name* as a quoted string. Use a semi-colon (;) between words if you want a line break in the column title. If you want the display name to be the same as the field name, or an existing display name in the source table, do not use AS. **Note** You must specify ON to use *break_field* , PAGE, NODUPS, or PRESORT.

Name	Description
FIELD *other_fields* WIDTH *characters* AS *display_name* <...*n*>	The fields to be included in the report. ◦ **WIDTH *characters*** - the output length of the field in characters ◦ **AS *display_name*** - the display name (alternate column title) for the field in the report Specify *display_name* as a quoted string. Use a semi-colon (;) between words if you want a line break in the column title. If you want the display name to be the same as the field name, or an existing display name in the source table, do not use AS. The SUBTOTAL and ACCUMULATE keywords are synonyms for FIELD, and have been deprecated. All numeric fields are automatically subtotaled. **Note** Break fields are automatically included in the report and do not need to be specified as *other_fields*.
SUPPRESS optional	Excludes blank detail lines from the report.
NOZEROS optional	Substitutes blank values for zero values in the field. For example, if a report includes a large number of zero values in a field, the report is easier to read if it only displays non-zero values.
LINE *n other_fields* optional	Specifies the number of output lines in the column and the fields to appear on the line number *n*. If no value is specified, the column defaults to a single line. The value of *n* must be between 2 and 60 inclusive. Column headings on the report are determined solely by the fields on the first line. *other_fields* specifies appropriate fields or expressions for the report.
PRESORT *sort_field* <...*n*> optional	◦ Sorts *break_field* , if one or more break fields are specified. ◦ Sorts *sort_field*, if one or more sort fields are specified. PRESORT does not sort the fields listed as *other_fields* unless they are also listed as *sort_field*.
SUMMARIZED optional	Produces a report with subtotals and totals only, and no detail lines. Subtotals are generated for the unique break field values. If SUMMARIZED is not specified, Analytics produces a report that includes detail lines, as well as subtotals for each of the specified key break fields.
SKIP *n* optional	Inserts blank lines between detail lines in the report. *n* must be an integer specifying the number of lines to insert. For example, `SKIP 1` produces a double-spaced report.
EOF optional	Execute the command one more time after the end of the file has been reached. This ensures that the final record in the table is processed when inside a GROUP command. Only use EOF if all fields are computed fields referring to earlier records.

Name	Description
TO SCREEN | PRINT| *filename* <HTML> optional	The location to send the results of the command to: ◦ **SCREEN** - displays the results in the Analytics display area > **Tip** > You can click any linked result value in the display area to drill down to the associated record or records in the source table. ◦ ***filename*** - saves the results to a file Specify *filename* as a quoted string with the appropriate file extension. For example: `TO "Output.TXT"` By default, the file is saved to the folder containing the Analytics project. Use either an absolute or relative file path to save the file to a different, existing folder: • `TO "C:\Output.TXT"` • `TO "Results\Output.TXT"` ◦ **PRINT** - sends the results to the default printer By default, reports output to a file are saved as ASCII text files. Specify `HTML` if you want to output the report as an HTML file (.htm). If you omit TO, the report is output to screen.
IF *test* optional	A conditional expression that must be true in order to process each record. The command is executed on only those records that satisfy the condition. > **Note** > The IF parameter is evaluated against only the records remaining in a table after any scope parameters have been applied (WHILE, FIRST, NEXT).
WHILE *test* optional	A conditional expression that must be true in order to process each record. The command is executed until the condition evaluates as false, or the end of the table is reached. > **Note** > If you use WHILE in conjunction with FIRST or NEXT, record processing stops as soon as one limit is reached.
FIRST *range* | NEXT *range* optional	The number of records to process: ◦ **FIRST** - start processing from the first record until the specified number of records is reached ◦ **NEXT** - start processing from the currently selected record until the specified number of records is reached Use *range* to specify the number of records to process. If you omit FIRST and NEXT, all records are processed by default.
HEADER *header_text* optional	The text to insert at the top of each page of a report. *header_text* must be specified as a quoted string. The value overrides the Analytics

Name	Description
	HEADER system variable.
FOOTER *footer_text* optional	The text to insert at the bottom of each page of a report. *footer_text* must be specified as a quoted string. The value overrides the Analytics FOOTER system variable.
APPEND optional	Appends the command output to the end of an existing file instead of overwriting it. **Note** You must ensure that the structure of the command output and the existing file are identical: • the same fields • the same field order • matching fields are the same length • matching fields are the same data type Analytics appends output to an existing file regardless of its structure. If the structure of the output and the existing file do not match, jumbled, missing, or inaccurate data can result.

Examples

Generating an HTML report

You generate a report from the **Ar** table and output the report to a formatted HTML file:

```
OPEN Ar
REPORT ON No FIELD Due Type Amount TO "C:\Reports\AR.htm" HTML
```

RETRIEVE command

Retrieves the result of a Direct Link query submitted for background processing.

> **Note**
> The RETRIEVE command is only supported if Direct Link is installed and configured on your local computer and on your organization's SAP system.

Syntax

```
RETRIEVE table_name PASSWORD num
```

Parameters

Name	Description
table_name	The name of the table originally created in Analytics by the Direct Link query. The table must already exist before you use RETRIEVE.
PASSWORD *num*	The password definition to use. You do not use PASSWORD *num* to prompt for, or specify, an actual password. The password definition refers to a password previously supplied or set using the PASSWORD command, the SET PASSWORD command, or the PASSWORD analytic tag. *num* is the number of the password definition. For example, if two passwords have been previously supplied or set in a script, or when scheduling an analytic, `PASSWORD 2` specifies that password #2 is used. For more information about supplying or setting passwords, see: ◦ "PASSWORD command" on page 1893 ◦ "SET command" on page 1960 ◦ "PASSWORD tag" on page 2530 **Note** The password is used to access the SAP system.

Examples

Retrieving the Background mode query result

You set the password and then retrieve the Background mode query result for an Analytics table named **DD02T_Data**:

```
SET PASSWORD 1 TO "pwd"
RETRIEVE DD02T_Data PASSWORD 1
```

SAMPLE command

Draws a sample of records using either the record sampling or monetary unit sampling method.

Record samplingMonetary unit sampling

Syntax

> **Note**
> The syntax does not include filtering (IF statements) or scope parameters because applying these options compromises the validity of a sample.

Fixed interval selection method

```
SAMPLE <ON> RECORD INTERVAL interval_value <FIXED initial_value>
{RECORD|FIELDS field_name <...n>|FIELDS ALL <EXCLUDE field_name <...n>>} TO
table_name <LOCAL> <OPEN> <APPEND>
```

Cell selection method

```
SAMPLE <ON> RECORD CELL INTERVAL interval_value <RANDOM seed_value>
{RECORD|FIELDS field_name <...n>|FIELDS ALL <EXCLUDE field_name <...n>>} TO
table_name <LOCAL> <OPEN> <APPEND> <MERSENNE_TWISTER>
```

Random selection method

```
SAMPLE <ON> RECORD NUMBER sample_size <RANDOM seed_value> <ORDER>
{RECORD|FIELDS field_name <...n>|FIELDS ALL <EXCLUDE field_name <...n>>} TO
table_name <LOCAL> <OPEN> <APPEND> <MERSENNE_TWISTER>
```

Parameters

> **Note**
> Do not include thousands separators when you specify values.

Name	Description
ON RECORD	Use record sampling.
INTERVAL *interval_value* FIXED *initial_value* | CELL INTERVAL *interval_value* | NUMBER *sample_size*	**INTERVAL *interval_value* FIXED *initial_value*** Use the **fixed interval** selection method. An initial record is selected and all subsequent selections are a fixed interval or distance apart - for example, every 20th record after the initial selection. ◦ **INTERVAL *interval_value*** - specify the interval value that was generated by calculating the sample size ◦ **FIXED *initial_value*** - specify the initial record number selected If you specify an *initial_value* of zero ('0'), or omit FIXED, Analytics randomly selects the initial record. **CELL INTERVAL *interval_value*** Use the **cell** selection method. The data set is divided into multiple equal-sized cells or groups, and one record is randomly selected from each cell. The *interval_value* dictates the size of each cell. Specify the interval value that was generated by calculating the sample size. **NUMBER *sample_size*** Use the **random** selection method. All records are randomly selected from the entire data set. Specify the sample size that was generated by calculating the sample size.
RANDOM *seed_value* optional	**Note** Cell and random selection methods only. The seed value to use to initialize the random number generator in Analytics. If you specify a value of zero ('0'), or omit RANDOM, Analytics randomly selects the seed value.
ORDER optional	**Note** Random selection method only. You can only use ORDER when specify FIELDS. Adds the ORDER field to the output results. This field displays the order in which each record is randomly selected.

Name	Description
RECORD | FIELDS *field_name <...n>* | FIELDS ALL	◦ **RECORD** - the entire record is included in the output table Fields are included in the order that they appear in the source table layout. ◦ **FIELDS *field_name*** - individual fields, rather than the entire record, are included in the output table Specify the field(s) or expressions to include. If you specify multiple fields, they must be separated by spaces. Fields are included in the order that you list them. ◦ **FIELDS ALL** - all fields are included in the output table Fields are included in the order that they appear in the source table layout.
EXCLUDE *field_name* optional	Only valid when sampling using FIELDS ALL. The field or fields to exclude from the command. EXCLUDE allows you to fine-tune FIELDS ALL, by excluding the specified fields. EXCLUDE must immediately follow FIELDS ALL. For example: `FIELDS ALL EXCLUDE field_1 field_2`
TO *table_name*	The location to send the results of the command to: ◦ ***table_name*** - saves the results to an Analytics table Specify *table_name* as a quoted string with a .FIL file extension. For example: `TO "Output.FIL"` By default, the table data file (.FIL) is saved to the folder containing the Analytics project. Use either an absolute or relative file path to save the data file to a different, existing folder: • `TO "C:\Output.FIL"` • `TO "Results\Output.FIL"` **Note** Table names are limited to 64 alphanumeric characters, not including the .FIL extension. The name can include the underscore character (_), but no other special characters, or any spaces. The name cannot start with a number.
LOCAL optional	Saves the output file in the same location as the Analytics project. **Note** Applicable only when running the command against a server table with an output file that is an Analytics table. The LOCAL parameter must immediately follow the TO parameter.
OPEN optional	Opens the table created by the command after the command executes. Only valid if the command creates an output table.

Name	Description
APPEND optional	Appends the command output to the end of an existing file instead of overwriting it. **Note** You must ensure that the structure of the command output and the existing file are identical: • the same fields • the same field order • matching fields are the same length • matching fields are the same data type Analytics appends output to an existing file regardless of its structure. If the structure of the output and the existing file do not match, jumbled, missing, or inaccurate data can result.
MERSENNE_TWISTER optional	**Note** Cell and random selection methods only. The random number generator in Analytics uses the Mersenne-Twister algorithm. If you omit MERSENNE_TWISTER, the default Analytics algorithm is used. **Note** You should only use the default Analytics algorithm if you require backward compatibility with Analytics scripts or sampling results created prior to Analytics version 12.

Examples

Draw a record sample

You are going to use record sampling to estimate the rate of deviation from a prescribed control in an account containing invoices.

After calculating a statistically valid sample size, you are ready to draw the sample. You are going to use the random selection method.

The example below:

- Samples the open Analytics table
- Uses the random selection method with a seed value of 123456
- Specifies a sample size of 95 records
- Includes only specified fields in the output table
- Specifies that the random number generator in Analytics uses the Mersenne-Twister algorithm

```
SAMPLE ON RECORD RANDOM 123456 NUMBER 95 FIELDS RefNum CustNum Amount
Date Type TO "Ar_record_sample" OPEN MERSENNE_TWISTER
```

Remarks

For more information about how this command works, see "Performing record sampling" on page 983.

Syntax

> **Note**
> The syntax does not include filtering (IF statements) or scope parameters because applying these options compromises the validity of a sample.

Fixed interval selection method

```
SAMPLE <ON> mus_numeric_field INTERVAL interval_value <FIXED initial_value>
<CUTOFF top_stratum_cutoff_value> <SUBSAMPLE> <NOREPLACEMENT> {RECORD|FIELDS
field_name <...n>|FIELDS ALL <EXCLUDE field_name <...n>>} TO table_name
<LOCAL> <OPEN> <APPEND>
```

Cell selection method

```
SAMPLE <ON> mus_numeric_field CELL INTERVAL interval_value <CUTOFF top_
stratum_cutoff_value> <RANDOM seed_value> <SUBSAMPLE> <NOREPLACEMENT>
{RECORD|FIELDS field_name <...n>|FIELDS ALL <EXCLUDE field_name <...n>>} TO
table_name <LOCAL> <OPEN> <APPEND> <MERSENNE_TWISTER>
```

Random selection method

```
SAMPLE <ON> mus_numeric_field NUMBER sample_size POPULATION absolute_value
<RANDOM seed_value> <SUBSAMPLE> <NOREPLACEMENT> <ORDER> {RECORD|FIELDS field_
name <...n>|FIELDS ALL <EXCLUDE field_name <...n>>} TO table_name <LOCAL>
<OPEN> <APPEND> <MERSENNE_TWISTER>
```

Parameters

> **Note**
> Do not include thousands separators when you specify values.

Name	Description
ON *mus_numeric_field*	Use monetary unit sampling (MUS). *mus_numeric_field* is the numeric field or expression to use as the basis for the sampling.
INTERVAL *interval_value* FIXED *initial_value* | CELL INTERVAL *interval_value* | NUMBER *sample_size* POPULATION *absolute_value*	**INTERVAL *interval_value* FIXED *initial_value*** Use the **fixed interval** selection method. An initial monetary unit is selected and all subsequent selections are a fixed interval or distance apart – for example, every 5000th monetary unit after the initial selection. ◦ **INTERVAL *interval_value*** – specify the interval value that was generated by calculating the sample size ◦ **FIXED *initial_value*** – specify the initial monetary unit selected If you specify an *initial_value* of zero ('0'), or omit FIXED, Analytics randomly selects the initial monetary unit. **CELL INTERVAL *interval_value*** Use the **cell** selection method. The data set is divided into multiple equal-sized cells or groups, and one monetary unit is randomly selected from each cell. The *interval_value* dictates the size of each cell. Specify the interval value that was generated by calculating the sample size. **NUMBER *sample_size* POPULATION *absolute_value*** Use the **random** selection method. All monetary units are randomly selected from the entire data set. ◦ **NUMBER *sample_size*** – specify the sample size that was generated by calculating the sample size ◦ **POPULATION *absolute_value*** – specify the total absolute value of *mus_numeric_field*, which is the population from which the sample will be selected
CUTOFF *top_stratum_cutoff_value* optional	> **Note** > Fixed interval and cell selection methods only. A top stratum cutoff value. Amounts in the *mus_numeric_field* greater than or equal to the cutoff value are automatically selected and included in the sample. If you omit CUTOFF, a default cutoff value equal to the *interval_value* is used.

Name	Description
RANDOM *seed_value* optional	**Note** Cell and random selection methods only. The seed value to use to initialize the random number generator in Analytics. If you specify a value of zero ('0'), or omit RANDOM, Analytics randomly selects the seed value.
SUBSAMPLE optional	**Note** You can only use SUBSAMPLE when specify FIELDS. Adds the SUBSAMPLE field to the output results. If each amount in a sample field represents a total of several separate transactions, and you want to perform audit procedures on only one transaction from each sampled total amount, you can use the values in the SUBSAMPLE field to randomly select the individual transactions. For more information, see "Performing monetary unit sampling" on page 1013.
NOREPLACEMENT optional	The same record is not selected more than once. As a result, the sample may contain fewer records than calculated by the SIZE command. If NOREPLACEMENT is omitted, or if you specify REPLACEMENT, records can be selected more than once.
ORDER optional	**Note** Random selection method only. You can only use ORDER when specify FIELDS. Adds the ORDER field to the output results. This field displays the order in which each record is randomly selected.
RECORD | FIELDS *field_name* <...*n*> | FIELDS ALL	◦ **RECORD** - the entire record is included in the output table Fields are included in the order that they appear in the source table layout. ◦ **FIELDS *field_name*** - individual fields, rather than the entire record, are included in the output table Specify the field(s) or expressions to include. If you specify multiple fields, they must be separated by spaces. Fields are included in the order that you list them. ◦ **FIELDS ALL** - all fields are included in the output table Fields are included in the order that they appear in the source table layout.
EXCLUDE *field_name* optional	Only valid when sampling using FIELDS ALL. The field or fields to exclude from the command. EXCLUDE allows you to fine-tune FIELDS ALL, by excluding the specified fields. EXCLUDE must immediately follow FIELDS ALL. For example:

Name	Description
	`FIELDS ALL EXCLUDE field_1 field_2`
TO *table_name*	The location to send the results of the command to: ◦ ***table_name*** - saves the results to an Analytics table Specify *table_name* as a quoted string with a .FIL file extension. For example: `TO "Output.FIL"` By default, the table data file (.FIL) is saved to the folder containing the Analytics project. Use either an absolute or relative file path to save the data file to a different, existing folder: • `TO "C:\Output.FIL"` • `TO "Results\Output.FIL"` **Note** Table names are limited to 64 alphanumeric characters, not including the .FIL extension. The name can include the underscore character (_), but no other special characters, or any spaces. The name cannot start with a number.
LOCAL optional	Saves the output file in the same location as the Analytics project. **Note** Applicable only when running the command against a server table with an output file that is an Analytics table. The LOCAL parameter must immediately follow the TO parameter.
OPEN optional	Opens the table created by the command after the command executes. Only valid if the command creates an output table.
APPEND optional	Appends the command output to the end of an existing file instead of overwriting it. **Note** You must ensure that the structure of the command output and the existing file are identical: • the same fields • the same field order • matching fields are the same length • matching fields are the same data type Analytics appends output to an existing file regardless of its structure. If the structure of the output and the existing file do not match, jumbled, missing, or inaccurate data can result.
MERSENNE_TWISTER optional	**Note** Cell and random selection methods only. The random number generator in Analytics uses the Mersenne-Twister algorithm.

Name	Description
	If you omit MERSENNE_TWISTER, the default Analytics algorithm is used. **Note** You should only use the default Analytics algorithm if you require backward compatibility with Analytics scripts or sampling results created prior to Analytics version 12.

Examples

Draw a monetary unit sample

You are going to use monetary unit sampling to estimate the total amount of monetary misstatement in an account containing invoices.

After calculating a statistically valid sample size, you are ready to draw the sample. You are going to use the fixed interval selection method.

The example below:

- Samples the open Analytics table based on a transaction amount field
- Uses the fixed interval selection method with an interval value of $6,283.33
- Specifies that the first record selected contains the 100,000th monetary unit (the number of cents in $1,000)
- Uses a top stratum cutoff of $5,000
- Includes the entire record in the output table

```
SAMPLE ON Amount INTERVAL 6283.33 FIXED 1000.00 CUTOFF 5000.00 RECORD TO
"Ar_monetary_unit_sample" OPEN
```

Remarks

For more information about how this command works, see "Performing monetary unit sampling" on page 1013.

SAVE command

Copies an Analytics table and saves it with a different name, or saves an Analytics project.

Syntax

To create a copy of an Analytics table and save it with a different name:

```
SAVE new_table FORMAT ACL_table
```

To save changes to the current project:

```
SAVE
```

Parameters

Name	Description
new_table	The name of the new Analytics table to create and save. **Note** Table names are limited to 64 alphanumeric characters. The name can include the underscore character (_), but no other special characters, or any spaces. The name cannot start with a number.
FORMAT *ACL_table*	The name of the existing Analytics table. Use the name of the table layout, not the name of an associated data file.

Examples

Creating a new table based on an existing one

You create a new table called **Payables_March** based on the existing table **Payables_master**. **Payables_March** can then be linked to the March payables data file:

```
SAVE Payables_March FORMAT Payables_master
```

Remarks

How it works

SAVE FORMAT produces a result similar to copying and pasting an Analytics table in the **Overview** tab in the **Navigator**. A new Analytics table is created and associated to the same data file or data source as the original table.

If required, you can link the newly created table to a different data source.

Using SAVE to avoid prompts

At certain points, Analytics prompts you to save changes to the current project. To avoid interruptions in the execution of scripts, you can use the SAVE command to save changes before Analytics prompts you.

SAVE LAYOUT command

Saves an Analytics table layout to an external table layout file (.layout), or saves table layout metadata to an Analytics table.

> **Note**
> Prior to version 11 of Analytics, external table layout files used an .fmt file extension. You can still save a table layout file with an .fmt extension by manually specifying the extension.

Syntax

```
SAVE LAYOUT {FILE|TABLE} TO {file_name|table_name}
```

Parameters

Name	Description
FILE | TABLE	◦ **FILE** - save an Analytics table layout to an external table layout file (.layout) ◦ **TABLE** - save table layout metadata to an Analytics table (.fil)
TO *file_name* | *table_name*	The name of the output file, and the output location: ◦ ***filename*** - the name of the .layout file Specify the *filename* as a quoted string. For example: `TO "Ap_Trans.layout"`. The .layout file extension is used by default, so specifying it is optional. By default, the file is saved to the folder containing the Analytics project. Use either an absolute or relative file path to save the file to a different, existing folder: • `TO "C:\Ap_Trans.layout"` • `TO "Table Layouts\Ap_Trans.layout"` **Note** Limit the table layout name to 64 alphanumeric characters, not including the .layout extension, to ensure that the name is not truncated when the table layout is imported back into Analytics. The name can include the underscore character (_), but no other special characters, or any spaces. The name cannot start with a number. ◦ ***table_name*** -the name of the Analytics table and .fil file

Name	Description
	Specify the *table_name* as a quoted string. For example: `TO "Ap_Trans_layout_metadata.fil"`. The .fil file extension is used by default, so specifying it is optional. By default, the table data file (.fil) is saved to the folder containing the Analytics project. Use either an absolute or relative file path to save the data file to a different, existing folder: • `TO "C:\Ap_Trans_layout_metadata.fil"` • `TO "Layout Metadata\Ap_Trans_layout_metadata.fil"` **Note** Table names are limited to 64 alphanumeric characters, not including the .FIL extension. The name can include the underscore character (_), but no other special characters, or any spaces. The name cannot start with a number.

Examples

Saving a table layout to an external table layout file (.layout)

The following examples save the table layout used by the open table to an external table layout file called **Ap_Trans.layout**:

Here, the table layout file is saved in the Analytics project folder:

```
SAVE LAYOUT FILE TO Ap_Trans.layout
```

Here, the table layout file is saved in the specified folder:

```
SAVE LAYOUT FILE TO "C:\ACL_DATA\AP Audit 2013\Ap_Trans.layout"
```

Saving a copy of table layout metadata to a new Analytics table

The following examples save a copy of the metadata in the table layout used by the open table to a new Analytics table called **Ap_Trans_layout_metadata**.

Here, the new Analytics table is saved in the Analytics project folder:

```
SAVE LAYOUT TABLE TO Ap_Trans_layout_metadata
```

Here, the new Analytics table is saved in the specified folder:

```
SAVE LAYOUT TABLE TO "C:\ACL_DATA\AP Audit 2013\Ap_Trans_layout_
metadata"
```

Remarks

SAVE LAYOUT file vs table

The SAVE LAYOUT command is used for two different purposes:

- **FILE** – saves the table layout of the open Analytics table to an external table layout file with a `.layout` extension
- **TABLE** – extracts the metadata from the table layout of the open Analytics table and saves it to a new Analytics table

SAVE LAYOUT FILE

How it works

SAVE LAYOUT FILE saves the table layout of the open Analytics table to an external table layout file with a .layout extension.

A table layout contains metadata that provides a structured interpretation of the raw data in an associated source data file. A table layout does not contain any source data itself.

When to use SAVE LAYOUT FILE

Saving a table layout as a .layout file makes the table layout and its metadata portable and reusable.

The .layout file can be imported into any Analytics project and associated with a matching source data file. The data elements in the source data file must match the field definitions specified by the table layout metadata.

For example, you could save the table layout of a transactions file from March, and associate it with a source data file containing transactions from April, assuming the structure of the data in the March and April source data files is identical. Used in this manner, .layout files can save you the labor of creating a new table layout from scratch.

For more information about the structure of Analytics tables, see the "Structuring data with table layouts" on page 696.

SAVE LAYOUT TABLE

How it works

SAVE LAYOUT TABLE extracts the metadata from the table layout of the open Analytics table and saves it to a new Analytics table.

The new table is not the table layout itself, but rather a regular Analytics table that contains a summary of the table layout metadata for the original table. Having access to this summary in an Analytics script can allow you to make decisions in the script based on the information.

For each field in the original table, the following pieces of table layout metadata are extracted into the new table.

> **Note**
> The field names in the new table are always generated in English regardless of which localized version of Analytics you are using.

Field name in new table	Table layout metadata
field_name	The name of the field
data_type	The data type of the field
category	The data category of the field
start_position	The start position of the field
field_length	The length of the field
decimals	The number of decimal places in the field (numeric fields only)
format	The format of the field (datetime and numeric fields only)
alternate_title	The alternate column title of the field
column_width	The width of the column in the view

Additional details

Computed fields	Computed fields are included in the extracted metadata, but the expression used by the computed field, and any conditions, are not recorded. Start position, field length, and decimal places are also not recorded for computed fields.
Related fields	Related fields are not included because they are not part of the table layout.
Field-level filters Field notes	Field-level filters and field notes are not included.
Alternate column title Column width	The values recorded for alternate column title and column width are the ones specified in the table layout, not the view-level values that can be specified for columns.

SAVE LOG command

Saves the entire command log, or the log entries for the current Analytics session, to an external file.

Syntax

```
SAVE LOG <SESSION> AS filename {<ASCII>|HTML} <OK>
```

Parameters

Name	Description
SESSION optional	Only log entries for the current Analytics session are saved.
AS *filename*	The name of the output file. Specify the *filename* as a quoted string. For example: `AS "Command Log"`. You can specify a file extension (.txt, or .htm or .html), but it is not required. By default, the file is saved to the folder containing the Analytics project. Use either an absolute or relative file path to save the file to a different, existing folder: ◦ `AS "C:\Command Log.TXT"` ◦ `AS "Results\Command Log.TXT"`
ASCII | HTML	The format of the output file: ◦ **ASCII (or no keyword)** - a plain text ASCII file. ◦ **HTML** - an HTML file.
OK optional	If a file with the same name as *filename* already exists, it is overwritten without confirmation.

Examples

Save the command log from payables analysis

You have performed data analysis on the March payables file and you want to save the associated command log as part of your working papers.

The following example saves the entries from the current Analytics session to an HTML file. If a file with the same name already exists it is overwritten without confirmation:

```
SAVE LOG SESSION AS "C:\Payables_March_Log.htm" HTML OK
```

SAVE TABLELIST command

Saves a list of all tables in an Analytics project to an Analytics table or a CSV file.

Syntax

```
SAVE TABLELIST {FILE|TABLE} TO {table_name|file_name}
```

Parameters

Name	Description
FILE | TABLE	◦ **FILE** - saves the table list to a CSV file (.csv) ◦ **TABLE** - saves the table list to an Analytics table
TO *table_name* | *file_name*	The location to save the table list: ◦ ***table_name*** - the name of the output Analytics table and the associated .fil file when using TABLE The .fil file extension is used by default and does not need to be specified. The table is saved in the same folder as the Analytics project, and cannot be saved in any other folder. **Note** Analytics table names are limited to 64 alphanumeric characters. The name can include the underscore character (_), but no other special characters, or any spaces. The name cannot start with a number. ◦ ***file_name*** - the name of the .csv file when using FILE The .csv file extension is used by default and does not need to be specified. You can specify an absolute or relative path to save the CSV file in an existing folder other than the folder containing the Analytics project. If you specify a relative path, it is relative to the Analytics working directory. You must specify values as quoted strings if they contain any spaces.

Examples

Creating a new table

You create a new table in the Analytics project called **Table_list_complete**:

```
SAVE TABLELIST TABLE TO Table_list_complete
```

Creating a CSV file

You create a new CSV file in the `C:\ACL Data` folder called `Table_list_complete.csv`:

```
SAVE TABLELIST FILE TO "C:\ACL Data\Table_list_complete"
```

Remarks

Output columns

The output Analytics table or CSV file contains three columns:

- **table_name** - the name of the Analytics table layout
- **type** - an indication whether the Analytics table is a local table or a server table
- **Data_file_Path** - the full path to the source data file

SAVE WORKSPACE command

Creates and saves a workspace.

Syntax

```
SAVE WORKSPACE workspace_name {field_name <...n>}
```

Parameters

Name	Description
workspace_name	The name of the workspace to create and add to the current Analytics project.
field_name <...n>	The name of the field to add to the workspace. You can include multiple field names separated by spaces.

Example

Activating a workspace

You create a workspace called **Inventory_margin** with two computed fields from the **Metaphor_Inventory_2002** table. Then you activate the workspace so that the fields are available in the **Inventory** table:

```
OPEN Metaphor_Inventory_2002
SAVE WORKSPACE Inventory_margin Gross_unit_margin Percent_unit_margin
OPEN Inventory
ACTIVATE WORKSPACE Inventory_margin OK
```

Remarks

Field names used to create computed fields must match

The names of any fields used in expressions that create a computed field that is saved in a workspace must match the names of the fields in the table that uses the workspace.

For example, if a workspace contains the computed field `Value=Sale_price*Quantity`, the active table must also contain fields called **Sale_price** and **Quantity**.

SEEK command

Searches an indexed character field for the first value that matches the specified character expression or character string.

Syntax

```
SEEK search_expression
```

Parameters

Name	Description
search_expression	The character expression to search for. You can use any valid character expression, character variable, or quoted string. *search_expression* is case-sensitive, and can include leading spaces, which are treated like characters.

Examples

Locate the first value in a field that matches a character variable

The Card_Number field has been defined as a character field and is indexed in ascending order.

The example below locates the first value in the field that exactly matches, or starts with, the value contained in the *v_card_num* variable.

```
INDEX ON Card_Number TO "CardNum" OPEN
SET INDEX TO "CardNum"
SEEK v_card_num
```

Locate the first value in a field that matches a character string

The Card_Number field has been defined as a character field and is indexed in ascending order.

The example below locates the first value in the field that exactly matches, or starts with, the character literal "AB-123":

```
INDEX ON Card_Number TO "CardNum" OPEN
SET INDEX TO "CardNum"
SEEK "AB-123"
```

Remarks

For more information about how this command works, see "Selecting the first matching record" on page 1167.

How it works

Use the SEEK command to move directly to the first record in a table containing the specified *search_expression* in the indexed character field.

- **If the *search_expression* is found** – the first matching record in the table is selected.
- **If the *search expression* is not found** – the message "No index matched key" is displayed, and the table is positioned at the first record with a greater value than the search expression.

  If there are no values in the indexed field greater than the search expression, the table is positioned at the first record.

Index required

To use SEEK to search a character field, you must first index the field in ascending order. If multiple character fields are indexed in ascending order, only the first field specified in the index is searched.

SEEK cannot be used to search non-character index fields, or character fields indexed in descending order.

Partial matching supported

Partial matching is supported. The search expression can be contained by a longer value in the indexed field. However, the search expression must appear at the start of the field to constitute a match.

The SEEK command is not affected by the **Exact Character Comparisons** option (SET EXACT ON/OFF).

SEQUENCE command

Determines if one or more fields in an Analytics table are in sequential order, and identifies out-of-sequence items.

Syntax

```
SEQUENCE <ON> {<FIELDS> key_field <D> <...n>|<FIELDS> ALL <EXCLUDE field_name
<...n>>} <UNFORMATTED> <ERRORLIMIT n> <IF test> <WHILE test> <FIRST range|NEXT
range> <TO {SCREEN|filename|PRINT}> <APPEND> <HEADER header_text> <FOOTER
footer_text> <PRESORT> <ISOLOCALE locale_code>
```

Parameters

Name	Description
ON FIELDS *key_field* D <...*n*> | FIELDS ALL	One or more character, numeric, or datetime fields to test for sequential order. ◦ **FIELDS *key_field*** - test the specified field or fields Multiple fields must be separated by spaces, and can be different data types. If you test by more than one field, fields are tested in the order that you list them. Include D to test key field values in descending order. The default test order is ascending. ◦ **FIELDS ALL** - test all fields in the table If you test by all fields, fields are tested in the order that they appear in the table layout. Testing key field values in ascending order is the only option for FIELDS ALL. **Note** When you test by more than one field, you are testing for a nested sequential order in the source table. Valid use of SEQUENCE requires that you specify test fields in the same order as the existing nested sequential order in the source table. Multiple test fields are tested as a nested group. They are not tested independently of one another.
EXCLUDE *field_name* optional	Only valid when testing for sequential order using FIELDS ALL. The field or fields to exclude from the command. EXCLUDE allows you to fine-tune FIELDS ALL, by excluding the specified fields.

Name	Description
	EXCLUDE must immediately follow FIELDS ALL. For example: `FIELDS ALL EXCLUDE field_1 field_2`
UNFORMATTED optional	Suppresses page headings and page breaks when the results are output to a file.
ERRORLIMIT *n* optional	The number of errors allowed before the command is terminated. The default value is 10.
IF *test* optional	A conditional expression that must be true in order to process each record. The command is executed on only those records that satisfy the condition. **Note** The IF parameter is evaluated against only the records remaining in a table after any scope parameters have been applied (WHILE, FIRST, NEXT).
WHILE *test* optional	A conditional expression that must be true in order to process each record. The command is executed until the condition evaluates as false, or the end of the table is reached. **Note** If you use WHILE in conjunction with FIRST or NEXT, record processing stops as soon as one limit is reached.
FIRST *range* | NEXT *range* optional	The number of records to process: ◦ **FIRST** - start processing from the first record until the specified number of records is reached ◦ **NEXT** - start processing from the currently selected record until the specified number of records is reached Use *range* to specify the number of records to process. If you omit FIRST and NEXT, all records are processed by default.
TO SCREEN | *filename* | PRINT optional	The location to send the results of the command to: ◦ **SCREEN** - displays the results in the Analytics display area **Tip** You can click any linked result value in the display area to drill down to the associated record or records in the source table. ◦ ***filename*** - saves the results to a file Specify *filename* as a quoted string with the appropriate file extension. For example: `TO "Output.TXT"` By default, the file is saved to the folder containing the Analytics project.

Name	Description
	Use either an absolute or relative file path to save the file to a different, existing folder: • `TO "C:\Output.TXT"` • `TO "Results\Output.TXT"` ◦ **PRINT** - sends the results to the default printer
APPEND optional	Appends the command output to the end of an existing file instead of overwriting it. **Note** You must ensure that the structure of the command output and the existing file are identical: • the same fields • the same field order • matching fields are the same length • matching fields are the same data type Analytics appends output to an existing file regardless of its structure. If the structure of the output and the existing file do not match, jumbled, missing, or inaccurate data can result.
HEADER *header_text* optional	The text to insert at the top of each page of a report. *header_text* must be specified as a quoted string. The value overrides the Analytics HEADER system variable.
FOOTER *footer_text* optional	The text to insert at the bottom of each page of a report. *footer_text* must be specified as a quoted string. The value overrides the Analytics FOOTER system variable.
PRESORT optional	Sorts the table on the key field before executing the command. **Note** You cannot use PRESORT inside the GROUP command.
ISOLOCALE *locale_code* optional	**Note** Applicable in the Unicode edition of Analytics only. The system locale in the format *language_country*. For example, to use Canadian French, enter `fr_ca`. Use the following codes: ◦ **language** - ISO 639 standard language code ◦ **country** - ISO 3166 standard country code If you do not specify a country code, the default country for the language is used. If you do not use ISOLOCALE, the default system locale is used.

Analytics output variables

Name	Contains
WRITE*n*	The total number of sequence errors identified by the command.

Examples

Testing for out-of-sequence employee IDs

You write any sequence errors identified in the **EmployeeID** field to a text file:

```
SEQUENCE ON EmployeeID ERRORLIMIT 10 TO "SequenceErrors.txt"
```

Remarks

For more information about how this command works, see "Testing sequential order" on page 1188.

Using SEQUENCE inside a GROUP

If you use SEQUENCE inside a GROUP command, the command executes to avoid interfering with the processing of the group, but no further data sequence errors are reported.

SET command

Sets a configurable Analytics option.

> **Note**
>
> The SET command sets an Analytics option for the duration of the Analytics session only. This behavior applies whether you use the SET command in the Analytics command line or in an Analytics script.
>
> To set Analytics options so that they persist between Analytics sessions, you must use the **Options** dialog box. For more information, see "Configuring Analytics options" on page 118.

Syntax

Syntax	Examples and remarks
SET BEEP *value*	`SET BEEP 2` Specifies the number of beeps to sound when command processing is completed. The *value* parameter must be between 1 and 255.
SET CENTURY *value*	`SET CENTURY 40` Specifies the start-of-century year for two-digit years. The *value* parameter must be from 0 to 99. Setting the start-of-century value to 40 means that two-digit years 40 to 99 are interpreted as 1940 to 1999, and two-digit years 00 to 39 are interpreted as 2000 to 2039.
SET CLEAN {ON | OFF}	`SET CLEAN ON` When this option is turned on, Analytics replaces invalid character data with blanks and invalid numeric data with zeros.
SET DATE <TO> {0 | 1 | 2 | *string*}	`SET DATE "YYYY/MM/DD"` Specifies how Analytics displays dates, and the date portion of datetimes, in views, reports, and exported files.

Syntax	Examples and remarks
	◦ `SET DATE 0` sets the date to MM/DD/YYYY format ◦ `SET DATE 1` sets the date to MM/DD/YY format ◦ `SET DATE 2` sets the date to DD/MM/YY format ◦ `SET DATE "<string>"` sets the date to the custom format you specify When using the SET DATE command to specify custom date formats, you must use 'D' for Day, 'M' for Month, and 'Y' for Year, even if you have specified different date format characters in the **Options** dialog box. For example: `SET DATE "DD MMM YYYY"`
SET DELETE_FILE {ON | OFF}	`SET DELETE_FILE ON` Default setting: OFF Specify ON to automatically delete the associated data file when you delete a table layout. Specify OFF to prevent the associated data file from being deleted when you delete a table layout. You must include the underscore (_) in DELETE_FILE. Specifying SET DELETE_FILE, without any parameter, in the command line displays whether DELETE_FILE is currently on or off. **Caution** Use caution when turning this option on. It may be an original data file that is deleted along with the table. Data files are deleted outright. They are not sent to the Windows Recycle Bin.
SET DESIGNATION *value*	`SET DESIGNATION "Produced by ABC Corporation"` The *value* parameter is a quoted string that specifies the label to display at the top of each printed page.

SET ECHO {ON | NONE}	`SET ECHO NONE`
	`COM Commands and results in scripts excluded from log.`
	`SET ECHO ON`
	Specify NONE to stop writing commands and results in scripts to the Analytics command log. Specify ON to resume logging.
	The SET ECHO command applies only to the logging of commands and results in scripts. Commands performed through the user interface or issued

<table>
<tr><th>Syntax</th><th>Examples and remarks</th></tr>
<tr><td></td><td>from the command line, and any results they produce, are always logged, regardless of how ECHO is set.

You can issue the SET ECHO NONE/ON command in a script or from the command line, but regardless of where you issue the command, it affects only the logging of commands and results in scripts.

Specifying SET ECHO, without any parameter, in the command line displays whether the logging of commands and results in scripts is currently on or off.</td></tr>
<tr><td>SET EXACT {ON | OFF}</td><td><pre>SET EXACT ON</pre>
Default setting: OFF

Controls how Analytics compares character fields, expressions, or literal values.

Note
Blank spaces are treated like characters.

◦ SET EXACT is OFF - Analytics uses the shorter string when comparing two strings of unequal length. The comparison starts with the leftmost characters and moves to the right.

For example, "AB" is equal to "AB", and it is also considered equal to "ABC".

◦ SET EXACT is ON - comparison strings must be exactly identical to be a match. When comparing two strings of unequal length, Analytics pads the shorter string with trailing blank spaces to match the length of the longer string.

For example, "AB" is equal to "AB", but it is not considered equal to "ABC".

For more examples illustrating SET EXACT, see "Exact Character Comparisons" in "Table options" on page 122.

You can use the ALLTRIM( ) function to remove leading and trailing blank spaces and ensure that only text characters and internal spaces are compared.

For example: <code>ALLTRIM(" AB") = ALLTRIM("AB")</code> is True when the values are wrapped with ALLTRIM( ), but False otherwise.

Some Analytics commands and functions are affected by SET EXACT and some are not:

<table><tr><th>Affected</th><th>Not affected</th></tr><tr><td>◦ LOCATE command
◦ MATCH( ) function
◦ BETWEEN( ) function</td><td>◦ JOIN command
◦ DEFINE RELATION command
◦ FIND( ) function
◦ FINDMULTI( ) function</td></tr></table></td></tr>
<tr><td>SET FILTER <TO> {test | filter_name}</td><td><pre>SET FILTER TO ProdNo = "070104347"</pre></td></tr>
</table>

Syntax	Examples and remarks
	`SET FILTER TO ProdNoFilter` Creates a global filter (view filter) on the open table, and specifies either a logical test, or the name of an existing saved filter. Specifying SET FILTER, without any parameter, removes any filter from the open table.
SET FOLDER *folder path*	Specifies the Analytics project folder in the **Overview** tab for command output. The default output folder is the folder containing the active table. This a DOS-style path using the format **/foldername/subfoldername**, in which the initial slash (/) indicates the root level in the **Overview** tab. You must specify a full file path. ◦ `SET FOLDER /Tables/Results` sets the output folder to the Results subfolder. If the Results subfolder does not exist, it is created. ◦ `SET FOLDER /` sets the output folder to the root level in the **Overview** tab ◦ `SET FOLDER` sets the output folder to the default (the folder containing the active table) The output folder remains as whatever you set it – until you reset it, or close the project. Upon opening the project, the output folder reverts to the default of the active table folder.
SET FORMAT {ON | OFF}	`SET FORMAT ON` Default setting: OFF If you use the ON parameter, Analytics automatically displays the current table layout and computed field definitions when you open a new table. The results appear in the command log.
SET FUZZYGROUPSIZE <TO> *num*	`SET FUZZYGROUPSIZE TO 10` Specifies the maximum number of items that can appear in a fuzzy duplicate group in the output results. The *num* parameter cannot be less than 2 or greater than 100. The default size is 20. The specified size remains in effect for the duration of the Analytics session.
SET GRAPH *type*	`SET GRAPH LINE` Specifies the graph type to use for all subsequently generated graphs. The commands run must be compatible with the specified graph type. For example, the BENFORD command cannot produce a PIE2D or PIE3D chart. If an incompatible graph type is specified the default graph type is used (BAR3D). The *type* parameter must be one of the following:

Syntax	Examples and remarks
	◦ PIE2D ◦ PIE3D ◦ BAR2D ◦ BAR3D - This is the default graph type. ◦ STACKED2D ◦ STACKED3D ◦ LAYERED ◦ LINE ◦ BENFORD - Combines 2D bar graph and 2D line graph.
SET HISTORY <TO> *value*	`SET HISTORY TO 50` Specifies the maximum number of table history entries to retain. The *value* parameter must be between 1 and 100.
SET INDEX <TO> *value*	`SET INDEX TO "CustomerCode.INX"` Specifies the index to apply to the active table.
SET LEARN <TO> *script*	`SET LEARN TO InventoryRec` Specifies the name of the script file that the **Script Recorder** uses to record commands.
SET LOCKAUTOSAVEFILE {ON | OFF}	`SET LOCKAUTOSAVEFILE ON` Default setting: OFF Specifies the mechanism for saving data to the Analytics log file (`*.LOG`). ◦ **LOCKAUTOSAVEFILE is ON** - log data is saved directly to disk, without using a write buffer. Saving log data directly to disk, without using a write buffer, can help prevent corruption of the log file when complex scripts run and potentially conflict with other processes on your computer such as anti-virus monitoring or automated backups. **Note** Specifying ON may cause Analytics to run slower. Use this option only if you have issues with log file corruption. ◦ **LOCKAUTOSAVEFILE is OFF** - log data is saved to a write buffer before

Syntax	Examples and remarks
	being saved to disk. The write buffer is an temporary data storage location that provides faster access than the hard disk drive and therefore overall faster execution of Analytics scripts.

SET LOG <TO> {*file* | OFF}	`SET LOG TO "analysis.log"`
	`SET LOG OFF`
	The first command switches logging to the specified log. If the specified log does not exist, it is created.
	The second command restores logging to the original Analytics command log.
	Note
	The maximum length of an Analytics project path and log name is 259 characters, which includes the file path, the log name, and the file extension (.log).

SET LOOP <TO> *num*	`SET LOOP TO 20` Specifies the maximum number of loops that can be executed by the LOOP command before the command is terminated. The *num* range is 0 to 32767, where 0 turns off loop testing.
SET MARGIN *side* <TO> *value*	`SET MARGIN TOP TO 100` Specify LEFT, RIGHT, TOP, or BOTTOM for the *side* parameter. If you want to change the margin on all sides, you need to specify each margin with a separate SET MARGIN command. Specifying a *value* of 100 creates a margin of 1 inch.
SET MATH <TO> {FIRST | LAST | MIN | MAX}	`SET MATH TO MIN` Default setting: MAX Specifies how decimal precision works when two operands are evaluated in a numeric expression. ◦ **FIRST** - use the number of decimal places of the first operand in a pair of operands ◦ **LAST** - use the number of decimal places of the last operand in a pair of operands ◦ **MIN** - use the minimum number of decimal places in a pair of operands

Syntax	Examples and remarks
	◦ **MAX** - use the maximum number of decimal places in a pair of operands In multi-operand expressions, the SET MATH setting works on a pairwise basis, applying the specified setting to each pair of operands, rounding as necessary, as they are evaluated in the standard mathematical order (BOMDAS). If the SET MATH setting reduces the number of decimal places in a result, the result is rounded, not truncated. For more information, see "Controlling rounding and decimal precision in numeric expressions" on page 803. **Note** You cannot use SET MATH while an Analytics table is open.
SET MONTHS <TO> *string*	Specifies the default three-character abbreviations for month names. The *string* parameter is the list of month abbreviations separated by commas.
SET NOTIFYFAILSTOP {ON | OFF}	`SET NOTIFYFAILSTOP ON` Default setting: OFF ◦ **NOTIFYFAILSTOP is OFF** - Analytics allows a script to continue even if a NOTIFY command in the script fails. ◦ **NOTIFYFAILSTOP is ON** - Analytics stops processing a script, and writes a message to the log, if a NOTIFY command in the script fails. The script stops after the initial failure, or after the specified number of NOTIFYRETRYATTEMPTS, if none of the attempts are successful.
SET NOTIFYRETRYATTEMPTS <TO> *num*	`SET NOTIFYRETRYATTEMPTS TO 10` Specifies the number of times the NOTIFY command will attempt to send an email if the initial attempt is unsuccessful. Enter a number from 0 to 255. If you enter 0, no additional attempts are made after an initial failure. The default is 5. One possible reason for the NOTIFY command failing to send an email is that the email server is unavailable.
SET NOTIFYRETRYINTERVAL <TO> *seconds*	`SET NOTIFYRETRYINTERVAL TO 30` Specifies the amount of time in seconds between NOTIFYRETRYATTEMPTS. Enter a number from 1 to 255. The default is 10 seconds.
SET ORDER <TO> *values*	Specifies the sort sequence for character fields. The *values* parameter lists all of the characters for the selected sort order. For more information, see "The Sort Order option and sort sequences " on page 1123.

Syntax	Examples and remarks
SET OVERFLOW {ON | OFF}	`SET OVERFLOW OFF` Default setting: ON If OFF is specified Analytics does not stop processing when an overflow error occurs.
SET PASSWORD *num* <TO> *string*	`SET PASSWORD 1 TO "password123"` Used to create a password definition, and specify a password value, for unattended script execution. The *num* parameter uniquely identifies the password definition and must be a value from 1 to 10. Specify the password value as a quoted string.
SET PERIODS <TO> *value* <,...*n*>	`SET PERIODS TO "0,30,90,180,10000"` Specifies the default aging periods used by the AGE command.
SET PICTURE *format*	`SET PICTURE "(9,999,999.99)"` Specifies the default formatting for numeric values.
SET READAHEAD <TO> *size*	Specifies the size of the data block read. You should only change this setting if you are advised to do so by Support.
SET RETRY <TO> *num* SET RETRYIMPORT <TO> *num*	`SET RETRY TO 50` Specifies the number of times Analytics attempts to import or export data if the initial attempt is unsuccessful. Enter a number from 0 to 255. If you enter 0, no additional attempts are made after an initial failure. The default is 0. There is no waiting period between retry attempts. Each successive attempt is made immediately after a preceding failure. The ability to specify retry attempts is useful when connecting to databases or cloud data services, which can be temporarily unavailable. Applies to the following commands: ◦ ACCESSDATA ◦ IMPORT GRCPROJECT ◦ IMPORT GRCRESULTS ◦ IMPORT SAP ◦ RETRIEVE ◦ REFRESH (for tables initially created using ACCESSDATA or IMPORT SAP only)

Syntax	Examples and remarks
	◦ EXPORT . . . ACLGRC (export to HighBond Results) **Note** SET RETRYIMPORT is retained for backward compatibility. SET RETRYIMPORT and SET RETRY perform identical actions.
SET SAFETY {ON | OFF}	`SET SAFETY OFF` Specify ON to display a confirmation dialog box when overwriting any of the following: ◦ fields in table layouts ◦ Analytics tables ◦ files, including Analytics data files (.fil) Specify OFF to prevent the dialog box from being displayed. Specifying SET SAFETY, without any parameter, in the command line displays whether SAFETY is currently on or off.
SET SEPARATORS <TO> *values*	`SET SEPARATORS TO ".,,"` Specifies the default decimal, thousands, and list separators used by Analytics. The SET SEPARATORS values must be three valid separator characters in the following order: ◦ decimal (period, comma, or space) ◦ thousands (period, comma, or space) ◦ list (semi-colon, comma, or space) Among the three separators, the decimal separator must be unique. You must specify all three separators when you use the command. The list separator is used primarily to separate function parameters.

SET SESSION <*session_name*>	`SET SESSION`
	
	`SET SESSION "Analysis"`
	
	Creates a new session in the Analytics command log. The session is identified by the current timestamp.
	The optional *session_name* allows you to add up to 30 characters of additional identifying information. Quotation marks are permitted but not required.

Syntax	Examples and remarks
SET SORTMEMORY *num*	`SET SORTMEMORY 800` Specifies the maximum amount of memory allocated for sorting and indexing processes. The *num* parameter must be a value from 0 to 2000 megabytes (MB), to be entered in 20MB increments. If the sort memory is set to 0, Analytics uses the memory currently available.
SET SUPPRESSTIME {ON | OFF}	`SET SUPPRESSTIME ON` Default setting: OFF Only for use when defining an Analytics table that uses an ODBC data source (IMPORT ODBC command), or direct database access (DEFINE TABLE DB command). If you use the ON parameter, when defining the table Analytics suppresses the time portion of datetime values. For example, 20141231 235959 is read, displayed in views, and subsequently processed as 20141231. Including this command in a pre-datetime Analytics script (pre v.10.0) that assumes the time portion of datetime data will be truncated allows the script to run in the datetime-enabled version of Analytics. Analytics suppresses the time portion by using only the date portion of the datetime format. The time data is still present in the .fil file or the database table. If required, you can redefine the field or define a new field to include the time portion of the data. If SET SUPPRESSTIME = OFF, Analytics tables defined using ODBC or direct database access include full datetime values. You can issue the SET SUPPRESSTIME ON/OFF command in a script or from the command line. Specifying SET SUPPRESSTIME, without any parameter, in the command line displays whether the suppression of the time portion of datetime data is currently on or off.
SET SUPPRESSXML {ON | OFF}	`SET SUPPRESSXML ON` Default setting: OFF Specifies that command output is in plain text rather than formatted text.
SET TEST {ON | OFF}	`SET TEST ON` Specifies whether the results of IF, WHILE, FOR, and NEXT tests associated with GROUP commands should be recorded in the log.

Syntax	Examples and remarks

SET TIME <TO> *string*	`SET TIME "hh:mm:ss PM"`
	
	Specifies how Analytics displays the time portion of datetimes, and standalone time values, in views, reports, and exported files.
	
	When using the SET TIME command to specify time formats, you must use 'h' for Hour, 'm' for Minute, and 's' for Second, even if you have specified different time format characters in the **Options** dialog box. For example:
	
	`SET TIME TO "hh:mm"`

SET UTCZONE {ON | OFF}	`SET UTCZONE OFF` Default setting: ON ◦ **UTCZONE is ON** - Analytics changes the display of local times with a UTC offset to the UTC equivalent of the local time. (UTC is Coordinated Universal Time, the time at zero degrees longitude.) ◦ **UTCZONE is OFF** - Analytics displays local times with a UTC offset without converting them to UTC. For example: ◦ 01 Jan 2015 04:59:59 (`SET UTCZONE ON`) ◦ 31 Dec 2014 23:59:59-05:00 (`SET UTCZONE OFF`) Conversion of local time to UTC is for display purposes only, and does not affect the source data. You can change back and forth between the two different display modes whenever you want to.
SET VERIFY {ON | OFF | BLANK}	`SET VERIFY ON` When ON is specified, Analytics automatically checks whether the contents of a data field correspond to the field's data type in the table layout whenever a table is opened. When BLANK is specified, Analytics replaces invalid character data with blanks and invalid numeric data with zeros, in addition to the verification described for the ON parameter.
SET WIDTH <TO> *characters*	`SET WIDTH TO 20` Specifies the default display width in characters for numeric computed fields or ad hoc numeric expressions when Analytics cannot determine the maximum width.

SIZE command

Calculates a statistically valid sample size, and sample interval, for record sampling or monetary unit sampling.

Record samplingMonetary unit sampling

Syntax

```
SIZE RECORD CONFIDENCE confidence_level POPULATION population_size PRECISION
tolerable_rate <ERRORLIMIT expected_rate> <TO {SCREEN|filename}>
```

Parameters

> **Note**
> Do not include thousands separators, or percentage signs, when you specify values.

Name	Description
RECORD	Calculate sample size for a record sample. ATTRIBUTE is a deprecated parameter that does the same thing as RECORD.
CONFIDENCE *confidence_level*	The desired confidence level that the resulting sample is representative of the entire population. For example, specifying 95 means that you want to be confident that 95% of the time the sample will in fact be representative. Confidence is the complement of "sampling risk". A 95% confidence level is the same as a 5% sampling risk.
POPULATION *population_size*	The number of records in the table you are sampling.
PRECISION *tolerable_rate*	The tolerable deviation rate, which is the maximum rate of deviation from a prescribed control that can occur and you still consider the control effective. For example, specifying 5 means that the deviation rate must be greater than 5% for you to consider the control not effective.
ERRORLIMIT *expected_rate* optional	The expected population deviation rate. This is the rate of deviation from a prescribed control that you expect to find. For example, specifying 1 means that you expect the deviation rate to be 1%.

Name	Description
	If you omit this parameter, an expected population deviation rate of 0% is used.
TO SCREEN | *filename*	The location to send the results of the command to: ◦ **SCREEN** - displays the results in the Analytics display area **Tip** You can click any linked result value in the display area to drill down to the associated record or records in the source table. ◦ ***filename*** - saves the results to a file Specify *filename* as a quoted string with the appropriate file extension. For example: `TO "Output.TXT"` By default, the file is saved to the folder containing the Analytics project. Use either an absolute or relative file path to save the file to a different, existing folder: • `TO "C:\Output.TXT"` • `TO "Results\Output.TXT"`

Analytics output variables

Name	Contains
SAMPINT*n*	The required sample interval calculated by the command.
SAMPSIZE*n*	The required sample size calculated by the command.

Examples

Calculate the required size and interval for a record sample

You have decided to use record sampling to estimate the rate of deviation from a prescribed control in an account containing invoices.

Before drawing the sample, you must first calculate the statistically valid sample size and sample interval.

You want to be confident that 95% of the time the sample drawn by Analytics will be representative of the population as a whole.

Using your specified confidence level, the example below calculates a sample size of 95, and a sample interval value of 8.12, to use when drawing a record sample:

```
SIZE RECORD CONFIDENCE 95 POPULATION 772 PRECISION 5 ERRORLIMIT 1 TO
SCREEN
```

Remarks

For more information about how this command works, see "Calculating sample size for a record sample" on page 976.

Syntax

```
SIZE MONETARY CONFIDENCE confidence_level POPULATION population_size
MATERIALITY tolerable_misstatement <ERRORLIMIT expected_misstatement> <TO
{SCREEN|filename}>
```

Parameters

> **Note**
> Do not include thousands separators, or percentage signs, when you specify values.

Name	Description
MONETARY	Calculate sample size for a monetary unit sample.
CONFIDENCE *confidence_level*	The desired confidence level that the resulting sample is representative of the entire population. For example, specifying 95 means that you want to be confident that 95% of the time the sample will in fact be representative. Confidence is the complement of "sampling risk". A 95% confidence level is the same as a 5% sampling risk.
POPULATION *population_size*	The total absolute value of the numeric sample field.
MATERIALITY *tolerable_*	The tolerable misstatement, which is the maximum total amount of misstatement

Name	Description
misstatement	that can occur in the sample field without being considered a material misstatement. For example, specifying 29000 means that the total amount of misstatement must be greater than $29,000 to be considered a material misstatement.
ERRORLIMIT *expected_misstatement* optional	The expected misstatement. This is the total amount of misstatement that you expect the sample field to contain. For example, specifying 5800 means that you expect the total amount of misstatement to be $5,800. If you omit this parameter, an expected misstatement of $0.00 is used.
TO SCREEN | *filename*	The location to send the results of the command to: ◦ **SCREEN** - displays the results in the Analytics display area **Tip** You can click any linked result value in the display area to drill down to the associated record or records in the source table. ◦ ***filename*** - saves the results to a file Specify *filename* as a quoted string with the appropriate file extension. For example: `TO "Output.TXT"` By default, the file is saved to the folder containing the Analytics project. Use either an absolute or relative file path to save the file to a different, existing folder: • `TO "C:\Output.TXT"` • `TO "Results\Output.TXT"`

Analytics output variables

Name	Contains
SAMPINT*n*	The required sample interval calculated by the command.
SAMPSIZE*n*	The required sample size calculated by the command.

Examples

Calculate the required size and interval for a monetary unit sample

You have decided to use monetary unit sampling to estimate the total amount of monetary misstatement in an account containing invoices.

Before drawing the sample, you must first calculate the statistically valid sample size and sample interval.

You want to be confident that 95% of the time the sample drawn by Analytics will be representative of the population as a whole.

Using your specified confidence level, the example below calculates a sample size of 93, and a sample interval value of 6,283.33, to use when drawing a monetary unit sample:

```
SIZE MONETARY CONFIDENCE 95 POPULATION 585674.41 MATERIALITY 29000
ERRORLIMIT 5800 TO SCREEN
```

Remarks

For more information about how this command works, see "Calculating sample size for a monetary unit sample" on page 1005.

SORT command

Sorts records in an Analytics table into an ascending or descending sequential order, based on a specified key field or fields. The results are output to a new, physically reordered Analytics table.

Syntax

```
SORT {<ON> key_field <D> <...n>|<ON> ALL <EXCLUDE field_name <...n>>} <FIELDS
field_name <AS display_name> <...n>|FIELDS ALL <EXCLUDE field_name <...n>>> TO
tablename <LOCAL> <IF test> <WHILE test> <FIRST range|NEXT range> <APPEND>
<OPEN> <ISOLOCALE locale_code>
```

Parameters

Name	Description
ON *key_field* D <...*n*> | ON ALL	The key field or fields, or the expression, to use for sorting. You can sort by any type of field, including computed fields and ad hoc expressions, regardless of data type. ◦ **ON *key_field*** - use the specified field or fields If you sort by more than one field, you create a nested sort in the output table. The order of nesting follows the order in which you specify the fields. Include D to sort a key field in descending order. The default sort order is ascending. ◦ **ON ALL** - use all fields in the table If you sort by all the fields in a table you create a nested sort in the output table. The order of nesting follows the order in which the fields appear in the table layout. An ascending sort order is the only option for ON ALL.
EXCLUDE *field_name* optional	Only valid when sorting using ON ALL. The field or fields to exclude from the command. EXCLUDE allows you to fine-tune ON ALL, by excluding the specified fields. EXCLUDE must immediately follow ON ALL. For example: `ON ALL EXCLUDE field_1 field_2`

Name	Description
FIELDS *field_name* <...*n*> | FIELDS ALL optional	**Note** Key fields are automatically included in the output table, and do not need to be specified using FIELDS. The fields to include in the output: ◦ **FIELDS *field_name*** - use the specified fields Fields are used in the order that you list them. Converts computed fields to physical fields of the appropriate data type in the destination table - ASCII or Unicode (depending on the edition of Analytics), ACL (the native numeric data type), Datetime, or Logical. Populates the physical fields with the actual computed values. ◦ **FIELDS ALL** - use all fields in the table Fields are used in the order that they appear in the table layout. Converts computed fields to physical fields of the appropriate data type in the destination table - ASCII or Unicode (depending on the edition of Analytics), ACL (the native numeric data type), Datetime, or Logical. Populates the physical fields with the actual computed values. ◦ **omit FIELDS** - the entire record is included in the sorted output table: all fields, and any undefined portions of the record Fields are used in the order that they appear in the table layout. Computed fields are preserved. **Tip** If you need only a portion of the data contained in a record, do not include all fields, or the entire record, in the sorted output table. Select only the fields you need, which in most cases speeds up the sorting process.
AS *display_name* optional	Only valid when using FIELDS. The display name (alternate column title) for the field in the view in the new Analytics table. If you want the display name to be the same as the field name, or an existing display name in the source table, do not use AS. Specify *display_name* as a quoted string. Use a semi-colon (;) between words if you want a line break in the column title. **Note** AS works only when outputting to a new table. If you are appending to an existing table, the alternate column titles in the existing table take precedence.
EXCLUDE *field_name* optional	Only valid when using FIELDS ALL. The field or fields to exclude from the command. EXCLUDE allows you to fine-tune FIELDS ALL, by excluding the specified fields. EXCLUDE must immediately follow FIELDS ALL. For example:

Name	Description
	`FIELDS ALL EXCLUDE field_1 field_2`
TO *table_name*	The location to send the results of the command to: ◦ ***table_name*** - saves the results to an Analytics table Specify *table_name* as a quoted string with a .FIL file extension. For example: `TO "Output.FIL"` By default, the table data file (.FIL) is saved to the folder containing the Analytics project. Use either an absolute or relative file path to save the data file to a different, existing folder: • `TO "C:\Output.FIL"` • `TO "Results\Output.FIL"` **Note** Table names are limited to 64 alphanumeric characters, not including the .FIL extension. The name can include the underscore character (_), but no other special characters, or any spaces. The name cannot start with a number.
LOCAL optional	Saves the output file in the same location as the Analytics project. **Note** Applicable only when running the command against a server table with an output file that is an Analytics table. The LOCAL parameter must immediately follow the TO parameter.
IF *test* optional	A conditional expression that must be true in order to process each record. The command is executed on only those records that satisfy the condition. **Note** The IF parameter is evaluated against only the records remaining in a table after any scope parameters have been applied (WHILE, FIRST, NEXT).
WHILE *test* optional	A conditional expression that must be true in order to process each record. The command is executed until the condition evaluates as false, or the end of the table is reached. **Note** If you use WHILE in conjunction with FIRST or NEXT, record processing stops as soon as one limit is reached.
FIRST *range* | NEXT *range* optional	The number of records to process: ◦ **FIRST** - start processing from the first record until the specified number of records is reached ◦ **NEXT** - start processing from the currently selected record until the specified

Name	Description
	number of records is reached Use *range* to specify the number of records to process. If you omit FIRST and NEXT, all records are processed by default.
APPEND optional	Appends the command output to the end of an existing file instead of overwriting it. **Note** You must ensure that the structure of the command output and the existing file are identical: • the same fields • the same field order • matching fields are the same length • matching fields are the same data type Analytics appends output to an existing file regardless of its structure. If the structure of the output and the existing file do not match, jumbled, missing, or inaccurate data can result.
OPEN optional	Open the table and apply the index to the table.
ISOLOCALE *locale_code* optional	**Note** Applicable in the Unicode edition of Analytics only. The system locale in the format *language_country*. For example, to use Canadian French, enter `fr_ca`. Use the following codes: ◦ **language** - ISO 639 standard language code ◦ **country** - ISO 3166 standard country code If you do not specify a country code, the default country for the language is used. If you do not use ISOLOCALE, the default system locale is used.

Examples

Sort on a single field, output entire records

You want to sort the records in the sample **Inventory** table by product number. The sorted records are extracted to a new Analytics table called **Inventory_Product_Number**.

Entire records are included in the output table:

```
SORT ON ProdNo TO "Inventory_Product_Number"
```

To switch from the default ascending sort order to a descending sort order, you add D after the key field name:

```
SORT ON ProdNo D TO "Inventory_Product_Number"
```

Sort on a single field, output a subset of fields

You want to sort the records in the sample **Inventory** table by product number. Only the key field and the specified non-key fields are extracted to a new Analytics table called **Inventory_Quantity_on_Hand**.

The third non-key field, **QtyOH**, is given the display name **Qty on Hand** in the output table:

```
SORT ON ProdNo FIELDS ProdDesc ProdStat QtyOH AS "Qty on Hand" TO
"Inventory_Quantity_on_Hand"
```

Sort on a single field, output all fields

You want to sort the records in the sample **Inventory** table by product number. All fields are extracted to a new Analytics table called **Inventory_Product_Number**.

The difference between using **FIELDS ALL** and outputting the entire record is that **FIELDS ALL** converts any computed fields in the source table to physical fields in the output table, and populates the fields with the actual computed values:

```
SORT ON ProdNo FIELDS ALL TO "Inventory_Product_Number"
```

Sort on multiple fields (nested sort)

You want to sort the records in the sample **Inventory** table by location, then by product class, and then by product number. The sorted records are extracted to a new Analytics table called **Inventory_Location_Class_Number**.

```
SORT ON Location ProdCls ProdNo TO "Inventory_Location_Class_Number"
```

Sort using related fields

You want to sort the records in the sample **Ap_Trans** table by the following fields:

- vendor state (related **Vendor** table)
- vendor city (related **Vendor** table)
- vendor number (**Ap_Trans** table)

All three key fields and the specified non-key fields, including the related field **Vendor.Vendor_Name**, are extracted to a new Analytics table called **Ap_Trans_State_City**:

```
SORT ON Vendor.Vendor_State Vendor.Vendor_City Vendor_No FIELDS
Vendor.Vendor_Name Invoice_No Invoice_Date Invoice_Amount Prodno Quant-
ity Unit_Cost TO "Ap_Trans_State_City"
```

Remarks

For more information about how this command works, see "Sorting records" on page 1127.

The sort sequence used by the SORT command

The SORT command uses whatever sort sequence is specified in the **Sort Order** option (**Tools > Options > Table**). The default sort sequences are shown below.

For detailed information, see "The Sort Order option and sort sequences " on page 1123.

Analytics Edition	Sort Order default	Associated sort sequence
non-Unicode	System Default (ASCII)	Numbers, then uppercase, then lowercase: `0, 1, 2... A, B, C... a, b, c...` For example, “Z” sorts before “a”.
Unicode	Mix Languages (UCA) (Unicode collation	Numbers, then lowercase and uppercase intermixed:

Analytics Edition	Sort Order default	Associated sort sequence
	algorithm)	`0, 1, 2... a, A, b, B, c, C...` For example, “a” sorts before “Z”.

Case sensitivity

SORT is case sensitive. Depending on which edition of Analytics you are using (non-Unicode or Unicode), casing in strings may affect sorting.

You can use the UPPER( ) function in conjunction with SORT if you do not want case to affect sorting:

```
SORT ON UPPER(key_field) TO "Sorted_Table"
```

Sorting on related fields

You can sort on related fields, and include related fields as non-key fields in a sorted output table. To reference a related field in the SORT command specify *child table name.field name*.

Fixed-length vs variable-length data files

The SORT command works on both fixed-length and variable-length data files.

STATISTICS command

Calculates statistics for one or more numeric or datetime fields in an Analytics table.

Syntax

```
STATISTICS {<ON> field_name <...n>|<ON> ALL <EXCLUDE field_name <...n>>} <STD>
<MODMEDQ> <NUMBER n> <TO {SCREEN|filename|PRINT}> <IF test> <WHILE test>
<FIRST range|NEXT range> <APPEND>
```

Parameters

Name	Description
ON *field_name* <...*n*> | ON ALL	Specify one or more numeric or datetime fields to generate statistics for, or specify ON ALL to generate statistics for all numeric and datetime fields in the Analytics table.
EXCLUDE *field_name* optional	Only valid when generating statistics using ON ALL. The field or fields to exclude from the command. EXCLUDE allows you to fine-tune ON ALL, by excluding the specified fields. EXCLUDE must immediately follow ON ALL. For example: `ON ALL EXCLUDE field_1 field_2`
STD optional	Calculates the standard deviation of the fields specified, in addition to the other statistics.
MODMEDQ optional	Calculates the mode, median, first quartile, and third quartile values of the fields specified, in addition to the other statistics.
NUMBER *n* optional	The number of high and low values to retain during processing. The default value is 5.
TO SCREEN | *filename* | PRINT optional	The location to send the results of the command to: ◦ **SCREEN** - displays the results in the Analytics display area

Name	Description
	Tip You can click any linked result value in the display area to drill down to the associated record or records in the source table. ◦ ***filename*** - saves the results to a file Specify *filename* as a quoted string with the appropriate file extension. For example: `TO "Output.TXT"` By default, the file is saved to the folder containing the Analytics project. Use either an absolute or relative file path to save the file to a different, existing folder: • `TO "C:\Output.TXT"` • `TO "Results\Output.TXT"` ◦ **PRINT** - sends the results to the default printer
IF *test* optional	A conditional expression that must be true in order to process each record. The command is executed on only those records that satisfy the condition. **Note** The IF parameter is evaluated against only the records remaining in a table after any scope parameters have been applied (WHILE, FIRST, NEXT).
WHILE *test* optional	A conditional expression that must be true in order to process each record. The command is executed until the condition evaluates as false, or the end of the table is reached. **Note** If you use WHILE in conjunction with FIRST or NEXT, record processing stops as soon as one limit is reached.
FIRST *range* | NEXT *range* optional	The number of records to process: ◦ **FIRST** - start processing from the first record until the specified number of records is reached ◦ **NEXT** - start processing from the currently selected record until the specified number of records is reached Use *range* to specify the number of records to process. If you omit FIRST and NEXT, all records are processed by default.
APPEND optional	Appends the command output to the end of an existing file instead of overwriting it.

Name	Description
	Note You must ensure that the structure of the command output and the existing file are identical: • the same fields • the same field order • matching fields are the same length • matching fields are the same data type Analytics appends output to an existing file regardless of its structure. If the structure of the output and the existing file do not match, jumbled, missing, or inaccurate data can result.

Analytics output variables

> **Note**
> If you generate statistics for more than one field in a table, the system-generated output variables contain values for the first listed field only.

Name	Contains
ABS*n*	The absolute value calculated by the command.
AVERAGE*n*	The mean value calculated by the command.
COUNT*n*	The record count calculated by the command. ◦ If the variable name is COUNT1, it is storing the record count for the most recent command executed. ◦ If the variable name is COUNT*n*, where *n* is greater than 1, the variable is storing the record count for a command executed within a GROUP command. The value of *n* is assigned based on the line number of the command in the GROUP. For example, if the command is one line below the GROUP command it is assigned the value COUNT2. If the command is four lines below the GROUP command, it is assigned the value COUNT5.
HIGH*n*	The 5th highest value identified by the command. The 5th highest is the default setting. The setting can be changed using the NUMBER parameter. For example, `NUMBER 3` specifies that the 3rd highest value is stored. **Note** When Analytics identifies the highest value, duplicate values are not factored out. For example, if values in descending order are 100, 100, 99, 98, the 3rd highest value is 99, not 98.
LOW*n*	The 5th lowest value identified by the command.

Name	Contains
	The 5th lowest is the default setting. The setting can be changed using the NUMBER parameter. For example, `NUMBER 3` specifies that the 3rd lowest value is stored. **Note** When Analytics identifies the lowest value, duplicate values are not factored out. For example, if values in ascending order are 1, 1, 2, 3, the 3rd lowest value is 2, not 3.
MAX*n*	The maximum value identified by the command.
MEDIAN*n*	The median value identified by the command.
MIN*n*	The minimum value identified by the command.
MODE*n*	The most frequently occurring value identified by the command.
Q25*n*	The first quartile value (lower quartile value) calculated by the command.
Q75*n*	The third quartile value (upper quartile value) calculated by the command.
RANGE*n*	The difference between the maximum and minimum values calculated by the command.
STDDEV*n*	The standard deviation value calculated by the command.
TOTAL*n*	The total value calculated by the command. The value of *n* is 1 unless the TOTAL command is inside a GROUP command, in which case the value of *n* corresponds to the line number of the TOTAL command in the GROUP command. For more information, see "GROUP command" on page 1741.

Examples

Generating conditional statistics

You generate statistics for the **Quantity** field in records where the product class ID is 01:

```
STATISTICS ON Quantity IF ProdCls = "01"
```

STRATIFY command

Groups records into numeric intervals based on values in a numeric field. Counts the number of records in each interval, and also subtotals specified numeric fields for each interval.

Syntax

```
STRATIFY <ON> numeric_field MINIMUM value MAXIMUM value {<INTERVALS
number>|FREE interval_value <...n> last_interval} <SUPPRESS>
<SUBTOTAL numeric_field <...n>|SUBTOTAL ALL <EXCLUDE numeric_field <...n>>>
<KEY break_field> <TO {SCREEN|table_name|filename|GRAPH|PRINT}> <LOCAL> <IF
test> <FIRST range|NEXT range> <WHILE test> <APPEND> <OPEN> <HEADER header_
text> <FOOTER footer_text> <STATISTICS>
```

Parameters

Name	Description
ON *numeric_field*	The numeric field or expression to be stratified.
MINIMUM *value*	Applies to numeric fields only. The minimum value of the first numeric interval. MINIMUM is optional if you are using FREE, otherwise it is required.
MAXIMUM *value*	Applies to numeric fields only. The maximum value of the last numeric interval. MAXIMUM is optional if you are using FREE, otherwise it is required.
INTERVALS *number* optional	Applies to numeric fields only. The number of equal-sized intervals Analytics produces over the range specified by the MINIMUM and MAXIMUM values. If you do not specify a number of intervals, the default number is used. The default is specified by the **Intervals** number on the **Command** tab in the **Options** dialog box.
FREE *interval_value <...n> last_interval* optional	Applies to numeric fields only. Creates custom-sized intervals by specifying the start point of each interval and the end point of the last interval. If you specify MINIMUM and MAXIMUM values, those values are the start point of the first interval and the end point of the last interval, and each *interval_value*

Name	Description
	creates an additional interval within the range. The interval values you specify must be greater than the MINIMUM value, and equal to or less than the MAXIMUM value. Interval values must be in numeric sequence and cannot contain duplicate values: `FREE -1000, 0, 1000, 2000, 3000` If you specify both FREE and INTERVALS, then INTERVALS is ignored.
SUPPRESS optional	Values above the MAXIMUM value and below the MINIMUM value are excluded from the command output.
SUBTOTAL *numeric_field <...n>* | SUBTOTAL ALL optional	One or more numeric fields or expressions to subtotal for each group. Multiple fields must be separated by spaces. Specify ALL to subtotal all the numeric fields in the table. If you do not select a subtotal field, the field you are stratifying on is automatically subtotaled. You must explicitly specify the stratify field if you want to subtotal it along with one or more other fields, or if you want to include statistics for the subtotaled stratify field.
EXCLUDE *numeric_field* optional	Only valid when using SUBTOTAL ALL. The field or fields to exclude from the command. EXCLUDE allows you to fine-tune SUBTOTAL ALL, by excluding the specified fields. EXCLUDE must immediately follow SUBTOTAL ALL. For example: `SUBTOTAL ALL EXCLUDE field_1 field_2`
KEY *break_field* optional	The field or expression that groups subtotal calculations. A subtotal is calculated each time the value of *break_field* changes. *break_field* must be a character field or expression. You can specify only one field, but you can use an expression that contains more than one field.
TO SCREEN *table_name* | *filename* | GRAPH | PRINT	The location to send the results of the command to: ◦ **SCREEN** - displays the results in the Analytics display area **Tip** You can click any linked result value in the display area to drill down to the associated record or records in the source table. ◦ ***table_name*** - saves the results to an Analytics table Specify *table_name* as a quoted string with a .FIL file extension. For example: `TO "Output.FIL"` By default, the table data file (.FIL) is saved to the folder containing the Analytics project.

Name	Description
	Use either an absolute or relative file path to save the data file to a different, existing folder: • `TO "C:\Output.FIL"` • `TO "Results\Output.FIL"` **Note** Table names are limited to 64 alphanumeric characters, not including the .FIL extension. The name can include the underscore character (_), but no other special characters, or any spaces. The name cannot start with a number. ◦ ***filename*** - saves the results to a file Specify *filename* as a quoted string with the appropriate file extension. For example: `TO "Output.TXT"` By default, the file is saved to the folder containing the Analytics project. Use either an absolute or relative file path to save the file to a different, existing folder: • `TO "C:\Output.TXT"` • `TO "Results\Output.TXT"` ◦ **GRAPH** - displays the results in a graph in the Analytics display area ◦ **PRINT** - sends the results to the default printer
LOCAL optional	Saves the output file in the same location as the Analytics project. **Note** Applicable only when running the command against a server table with an output file that is an Analytics table. The LOCAL parameter must immediately follow the TO parameter.
IF *test* optional	A conditional expression that must be true in order to process each record. The command is executed on only those records that satisfy the condition. **Note** The IF parameter is evaluated against only the records remaining in a table after any scope parameters have been applied (WHILE, FIRST, NEXT).
FIRST *range* | NEXT *range* optional	The number of records to process: ◦ **FIRST** - start processing from the first record until the specified number of records is reached ◦ **NEXT** - start processing from the currently selected record until the specified number of records is reached Use *range* to specify the number of records to process. If you omit FIRST and NEXT, all records are processed by default.
WHILE *test* optional	A conditional expression that must be true in order to process each record. The command is executed until the condition evaluates as false, or the end of the table

Name	Description
	is reached. **Note** If you use WHILE in conjunction with FIRST or NEXT, record processing stops as soon as one limit is reached.
APPEND optional	Appends the command output to the end of an existing file instead of overwriting it. **Note** You must ensure that the structure of the command output and the existing file are identical: • the same fields • the same field order • matching fields are the same length • matching fields are the same data type Analytics appends output to an existing file regardless of its structure. If the structure of the output and the existing file do not match, jumbled, missing, or inaccurate data can result.
OPEN optional	Opens the table created by the command after the command executes. Only valid if the command creates an output table.
HEADER *header_text* optional	The text to insert at the top of each page of a report. *header_text* must be specified as a quoted string. The value overrides the Analytics HEADER system variable.
FOOTER *footer_text* optional	The text to insert at the bottom of each page of a report. *footer_text* must be specified as a quoted string. The value overrides the Analytics FOOTER system variable.
STATISTICS optional	**Note** Cannot be used unless SUBTOTAL is also specified. Calculates average, minimum, and maximum values for all SUBTOTAL fields.

Examples

Stratifying on invoice amount

You need to stratify an accounts receivable table on the **Invoice_Amount** field. The invoice amount is also automatically subtotaled.

The output is grouped into $1000 intervals:

- from $0 to $999.99
- from $1,000 to $1,999.99
- so on

The total invoice amount is included for each interval.

```
OPEN Ar
STRATIFY ON Invoice_Amount MINIMUM 0 MAXIMUM 10000 INTERVALS 10 TO
"Stratified_invoices.FIL"
```

Remarks

For more information about how this command works, see "Stratifying data" on page 1245.

How it works

STRATIFY groups records into equal-sized, or custom-sized, numeric intervals based on values in a numeric field.

The output contains a single record for each interval, with a count of the number of records in the source table that fall into each interval.

Automatically populate the MINIMUM and MAXIMUM values

You can run the STATISTICS or PROFILE commands on the stratify field before running the STRATIFY command to automatically populate the MINIMUM and MAXIMUM parameter values with the lowest and highest values in the field.

Names of auto-generated subtotal and statistics fields

If you use STATISTICS to perform statistical calculations on one or more SUBTOTAL fields, and output the results to an Analytics table, the fields auto-generated by the parameters have the following names:

Description of auto-generated field	Field name in output table	Alternate column title (display name) in output table
Subtotal field	*subtotaled field name in source table*	**Total** + *subtotaled alternate column title in source table*

Description of auto-generated field	Field name in output table	Alternate column title (display name) in output table
Average field	**a_***subtotaled field name in source table*	**Average** + *subtotaled alternate column title in source table*
Minimum field	**m_***subtotaled field name in source table*	**Minimum** + *subtotaled alternate column title in source table*
Maximum field	**x_***subtotaled field name in source table*	**Maximum** + *subtotaled alternate column title in source table*

SUMMARIZE command

Groups records based on identical values in one or more character, numeric, or datetime fields. Counts the number of records in each group, and also subtotals specified numeric fields for each group.

Syntax

```
SUMMARIZE {ON key_field <...n>|ON ALL <EXCLUDE field_name <...n>>}
<SUBTOTAL numeric_field <...n>|SUBTOTAL ALL <EXCLUDE numeric_field <...n>>>
<OTHER field <...n>|OTHER ALL <EXCLUDE field_name <...n>>> <TO {SCREEN|table_
name|PRINT}> <LOCAL> <IF test> <WHILE test> <FIRST range|NEXT range> <PRESORT>
<APPEND> <OPEN> <HEADER header_text> <FOOTER footer_text> <STATISTICS>
<MODMEDQ> <STDEV> <CPERCENT> <ISOLOCALE locale_code>
```

Parameters

Name	Description
ON *key_field <...n>* | ON ALL	One or more character, numeric, or datetime fields to summarize. ◦ **ON *key_field*** - use the specified field or fields Multiple fields must be separated by spaces, and can be different data types. If you summarize by more than one field, fields are summarized in the order that you list them. If you specify PRESORT, the nested sort of the output table follows the same order. ◦ **ON ALL** - use all fields in the table If you summarize by all fields, fields are summarized in the order that they appear in the table layout. If you specify PRESORT, the nested sort of the output table follows the same order.
EXCLUDE *field_name* optional	Only valid when summarizing using ON ALL. The field or fields to exclude from the command. EXCLUDE allows you to fine-tune ON ALL, by excluding the specified fields. EXCLUDE must immediately follow ON ALL. For example: `ON ALL EXCLUDE field_1 field_2`

Name	Description
SUBTOTAL *numeric_field <...n>* | SUBTOTAL ALL optional	One or more numeric fields or expressions to subtotal for each group. Multiple fields must be separated by spaces. Specify ALL to subtotal all the numeric fields in the table.
EXCLUDE *numeric_field* optional	Only valid when using SUBTOTAL ALL. The field or fields to exclude from the command. EXCLUDE allows you to fine-tune SUBTOTAL ALL, by excluding the specified fields. EXCLUDE must immediately follow SUBTOTAL ALL. For example: `SUBTOTAL ALL EXCLUDE field_1 field_2`
OTHER *field <...n>* | OTHER ALL optional	One or more additional fields to include in the output. ◦ **OTHER *field <...n>*** - include the specified field or fields ◦ **OTHER ALL** - include all fields in the table that are not specified as key fields or subtotal fields Use OTHER only with fields that contain the same value for all records in each summarized group. If you specify a field that contains values that are different for a summarized group, only the value for the first record in the group is displayed, which is not meaningful. For example: ◦ **summarize a table on customer number** - an appropriate "other field" is **Customer Name**. Typically, the customer name is identical for all records with the same customer number. ◦ **summarize a vendor table by state** - an inappropriate "other field" is **City**. Only the first city listed for each state appears in the output. In this instance, the better approach is to summarize using both state and city as key fields, in that order.
EXCLUDE *field_name* optional	Only valid when using OTHER ALL. The field or fields to exclude from the command. EXCLUDE allows you to fine-tune OTHER ALL, by excluding the specified fields. EXCLUDE must immediately follow OTHER ALL. For example: `OTHER ALL EXCLUDE field_1 field_2`
TO SCREEN *table_name* | PRINT	The location to send the results of the command to: ◦ **SCREEN** - displays the results in the Analytics display area **Tip** You can click any linked result value in the display area to drill down to the associated record or records in the source table. ◦ ***table_name*** - saves the results to an Analytics table Specify *table_name* as a quoted string with a .FIL file extension. For example: `TO`

Name	Description
	`"Output.FIL"` By default, the table data file (.FIL) is saved to the folder containing the Analytics project. Use either an absolute or relative file path to save the data file to a different, existing folder: • `TO "C:\Output.FIL"` • `TO "Results\Output.FIL"` **Note** Table names are limited to 64 alphanumeric characters, not including the .FIL extension. The name can include the underscore character (_), but no other special characters, or any spaces. The name cannot start with a number. ◦ **PRINT** - sends the results to the default printer
LOCAL optional	Saves the output file in the same location as the Analytics project. **Note** Applicable only when running the command against a server table with an output file that is an Analytics table. The LOCAL parameter must immediately follow the TO parameter.
IF *test* optional	A conditional expression that must be true in order to process each record. The command is executed on only those records that satisfy the condition. **Note** The IF parameter is evaluated against only the records remaining in a table after any scope parameters have been applied (WHILE, FIRST, NEXT).
WHILE *test* optional	A conditional expression that must be true in order to process each record. The command is executed until the condition evaluates as false, or the end of the table is reached. **Note** If you use WHILE in conjunction with FIRST or NEXT, record processing stops as soon as one limit is reached.
FIRST *range* | NEXT *range* optional	The number of records to process: ◦ **FIRST** - start processing from the first record until the specified number of records is reached ◦ **NEXT** - start processing from the currently selected record until the specified number of records is reached Use *range* to specify the number of records to process. If you omit FIRST and NEXT, all records are processed by default.

Name	Description
PRESORT optional	Sorts the table on the key field before executing the command. **Note** You cannot use PRESORT inside the GROUP command. ## If you use PRESORT If you use **PRESORT**, the output is sorted and contains a single, unique group for each set of identical values, or identical combination of values, in the key field or fields. **Tip** If the input table is already sorted, you can save processing time by not specifying **PRESORT**. ## If you do not use PRESORT If you do not use **PRESORT**, the output results use the sort order of the input table. If the key field or fields contain non-sequential identical values, the output results contain more than one group for each set of identical values, or identical combination of values. **Note** Depending on the context, more than one group for each set of identical values, or identical combination of values, can defeat the purpose of summarizing.
APPEND optional	Appends the command output to the end of an existing file instead of overwriting it. **Note** You must ensure that the structure of the command output and the existing file are identical: • the same fields • the same field order • matching fields are the same length • matching fields are the same data type Analytics appends output to an existing file regardless of its structure. If the structure of the output and the existing file do not match, jumbled, missing, or inaccurate data can result.
OPEN optional	Opens the table created by the command after the command executes. Only valid if the command creates an output table.
HEADER *header_text* optional	The text to insert at the top of each page of a report. *header_text* must be specified as a quoted string. The value overrides the Analytics HEADER system variable.

Name	Description
FOOTER *footer_text* optional	The text to insert at the bottom of each page of a report. *footer_text* must be specified as a quoted string. The value overrides the Analytics FOOTER system variable.
STATISTICS optional	**Note** Cannot be used unless SUBTOTAL is also specified. Calculates average, minimum, and maximum values for all SUBTOTAL fields.
MODMEDQ optional	**Note** Cannot be used unless SUBTOTAL is also specified. Calculates mode, median, first quartile, and third quartile values for all SUBTOTAL fields.
STDEV optional	**Note** Cannot be used unless SUBTOTAL is also specified. Calculates standard deviation and percentage of total for all SUBTOTAL fields.
CPERCENT optional	Calculates percentage of record count for each group.
ISOLOCALE optional	**Note** Applicable in the Unicode edition of Analytics only. The system locale in the format *language_country*. For example, to use Canadian French, enter `fr_ca`. Use the following codes: ◦ **language** - ISO 639 standard language code ◦ **country** - ISO 3166 standard country code If you do not specify a country code, the default country for the language is used. If you do not use ISOLOCALE, the default system locale is used.

Examples

Total transaction amount per customer

You summarize an accounts receivable table on the **Customer_Number** field, and subtotal the **Trans_Amount** field. The output is grouped by customer and includes the total transaction

amount for each customer:

```
OPEN Ar
SUMMARIZE ON Customer_Number SUBTOTAL Trans_Amount TO "Customer_
total.FIL" PRESORT
```

Total transaction amount per customer per transaction date

You summarize an accounts receivable table on the **Customer_Number** and **Trans_Date** fields. You subtotal the **Trans_Amount** field.

The output is grouped by customer, and within customer by date, and includes the total transaction amount for each customer for each date the customer had transactions.

```
OPEN Ar
SUMMARIZE ON Customer_Number Trans_Date SUBTOTAL Trans_Amount TO "Cus-
tomer_total_by_date.FIL" PRESORT
```

Total, average, minimum, and maximum transaction amounts per customer per transaction date

You add STATISTICS to the previous example.

In addition to the subtotaled transaction amount for each customer for each date the customer had transactions, you also calculate the average, minimum, and maximum transaction amounts for each customer for each date:

```
OPEN Ar
SUMMARIZE ON Customer_Number Trans_Date SUBTOTAL Trans_Amount TO "Cus-
tomer_stats_by_date.FIL" PRESORT STATISTICS
```

Identical transaction amounts, same date

You summarize a credit card transactions table on the **Trans_Date** and **Trans_Amount** fields.

The output is grouped by date, and within date by amount. You can use the associated count to identify transactions with identical amounts and identical dates:

```
OPEN CC_Trans
SUMMARIZE ON Trans_Date Trans_Amount TO "Transactions_by_date_amoun-
t.FIL" OPEN PRESORT
SET FILTER TO COUNT > 1
```

Remarks

For more information about how this command works, see "Summarizing data" on page 1267.

How it works

SUMMARIZE groups records that have the same value in a field, or the same combination of values across multiple fields. The output results contain a single record for each group, with a count of the number of records in the source table that belong to the group.

Subtotal and statistics: calculations and field names in the output results

You can use one or more optional parameters to perform statistical calculations on any SUBTOTAL field you specify. The statistical calculations are broken down by group in the output:

Optional Parameter	Alternate column title (display name) in output table	Field name in output table	Calculation performed on subtotaled field
SUBTOTAL	**Total** + *subtotaled alternate column title*	*subtotaled field name*	Subtotaled values for each group
STATISTICS	**Average** + *subtotaled alternate column title*	**a**_*subtotaled field name*	The average value for each group
	Minimum + *subtotaled alternate column title*	**m**_*subtotaled field name*	The minimum value for each group
	Maximum + *subtotaled alternate column title*	**x**_*subtotaled field name*	The maximum value for each group
MODMEDQ	**Median** + *subtotaled alternate column title*	**c**_*subtotaled field name*	The median value for each group

Optional Parameter	Alternate column title (display name) in output table	Field name in output table	Calculation performed on subtotaled field
			◦ Odd-numbered sets of values: the middle value ◦ Even-numbered sets of values: the average of the two values at the middle
	Mode + *subtotaled alternate column title*	**o**_*subtotaled field name*	The most frequently occurring value for each group ◦ Displays "N/A" if no value occurs more than once ◦ In the event of a tie, displays the lowest value
	Q25 + *subtotaled alternate column title*	**q**_*subtotaled field name*	The first quartile value for each group (lower quartile value) ◦ The result is an interpolated value based on an Analytics algorithm ◦ Produces the same result as the QUARTILE and QUARTILE.INC functions in Microsoft Excel
	Q75 + *subtotaled alternate column title*	**p**_*subtotaled field name*	The third quartile value for each group (upper quartile value) ◦ The result is an interpolated value based on an Analytics algorithm ◦ Produces the same result as the QUARTILE and QUARTILE.INC functions in Microsoft Excel
STDEV	**STDDEV** + *subtotaled alternate column title*	**d**_*subtotaled field name*	The standard deviation for each group

Optional Parameter	Alternate column title (display name) in output table	Field name in output table	Calculation performed on subtotaled field
	% Field + *subtotaled alternate column title*	**f_***subtotaled field name*	Each group's subtotal expressed as a percentage of the field total
CPERCENT	**Percent of Count**	**COUNT_PERCENTAGE**	The percentage of source table records belonging to each group **Note** Does not require a subtotal field

TOP command

Moves to the first record in an Analytics table.

Syntax

```
TOP
```

Parameters

This command does not have any parameters.

Remarks

When to use TOP

Use TOP to move to the first record in a table if a previous command, such as FIND, selected another record in the table.

TOTAL command

Calculates the total value of one or more fields in an Analytics table.

Syntax

```
TOTAL {<FIELDS> numeric_field <...n>|<FIELDS> ALL <EXCLUDE numeric_field
<...n>>} <IF test> <WHILE test> <FIRST range|NEXT range>
```

Parameters

Name	Description
FIELDS *numeric_field* <...*n*> | FIELDS ALL	The numeric field or fields to total. Specify ALL to total each of the numeric fields in the table.
EXCLUDE *numeric_field* optional	Only valid when totaling using FIELDS ALL. The field or fields to exclude from the command. EXCLUDE allows you to fine-tune FIELDS ALL, by excluding the specified fields. EXCLUDE must immediately follow FIELDS ALL. For example: `FIELDS ALL EXCLUDE field_1 field_2`
IF *test* optional	A conditional expression that must be true in order to process each record. The command is executed on only those records that satisfy the condition. **Note** The IF parameter is evaluated against only the records remaining in a table after any scope parameters have been applied (WHILE, FIRST, NEXT).
WHILE *test* optional	A conditional expression that must be true in order to process each record. The command is executed until the condition evaluates as false, or the end of the table is reached. **Note** If you use WHILE in conjunction with FIRST or NEXT, record processing stops as soon as one limit is reached.

Name	Description
FIRST *range* | NEXT *range* optional	The number of records to process: ◦ **FIRST** - start processing from the first record until the specified number of records is reached ◦ **NEXT** - start processing from the currently selected record until the specified number of records is reached Use *range* to specify the number of records to process. If you omit FIRST and NEXT, all records are processed by default.

Analytics output variables

> **Note**
> If you total more than one field in a table, the system-generated output variable contains the total for the first listed field only.

Name	Contains
TOTAL*n*	The total value calculated by the command. The value of *n* is 1 unless the TOTAL command is inside a GROUP command, in which case the value of *n* corresponds to the line number of the TOTAL command in the GROUP command. For more information, see "GROUP command" on page 1741.

Examples

Totaling the first 25 records

You calculate the total amount of the **MKTVAL** field for the first 25 records in the table:

```
TOTAL FIELDS MKTVAL FIRST 25
```

Remarks

When to use TOTAL

Use TOTAL to verify the completeness and accuracy of the source data and to produce control totals. The command calculates the arithmetic sum of the specified fields or expressions.

TRAIN command

Uses automated machine learning to create an optimum predictive model using a training data set.

> **Note**
> The TRAIN command is not supported if you are running Analytics on a 32-bit computer. The computation required by the command is processor-intensive and better suited to 64-bit computers.

Syntax

```
TRAIN {CLASSIFIER|REGRESSOR} <ON> key_field <...n> TARGET labeled_field SCORER
{ACCURACY|AUC|F1|LOGLOSS|PRECISION|RECALL|MAE|MSE|R2} SEARCHTIME minutes
MAXEVALTIME minutes MODEL model_name TO table_name <IF test> <WHILE test>
<FIRST range|NEXT range> FOLDS number_of_folds <SEED seed_value> <LINEAR>
<NOFP>
```

> **Note**
> The maximum supported size of the data set used with the TRAIN command is 1 GB.

Parameters

Name	Description
CLASSIFIER | REGRESSOR	The prediction type to use when training a predictive model: ◦ **CLASSIFIER** - use classification algorithms to train a model Use classification if you want to predict which class or category records belong to. ◦ **REGRESSOR** - use regression algorithms to train a model Use regression if you want to predict numeric values associated with records.
ON *key_field* <*...n*>	One or more training input fields. Fields can be character, numeric, or logical. Multiple fields must be separated by spaces.

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>

> **Note**
> Character fields must be "categorical". That is, they must identify categories or classes, and contain a maximum number of unique values.
>
> The maximum is specified by the **Maximum Categories** option (**Tools > Options > Command**).

</td></tr>
<tr><td>TARGET labeled_field</td><td>

The field that the model is being trained to predict based on the training input fields.

The different prediction types (classification or regression) work with different field data types:

Valid with CLASSIFIER	a character or logical target field
Valid with REGRESSOR	a numeric target field

</td></tr>
<tr><td>SCORER ACCURACY | AUC | F1 | LOGLOSS | PRECISION | RECALL | MAE | MSE | R2</td><td>

The metric to use when scoring (tuning and ranking) the generated models.

The generated model with the best value for this metric is kept, and the rest of the models are discarded.

A different subset of metrics is valid depending on the prediction type you are using (classification or regression):

Valid with CLASSIFIER	ACCURACY | AUC | F1 | LOGLOSS | PRECISION | RECALL
Valid with REGRESSOR	MAE | MSE | R2

> **Note**
> The classification metric AUC is only valid when *labeled_field* contains binary data - that is, two classes, such as Yes/No, or True/False.

</td></tr>
<tr><td>SEARCHTIME minutes</td><td>

The total time in minutes to spend training and optimizing a predictive model.

Training and optimizing involves searching across different pipeline configurations (different model, preprocessor, and hyperparameter combinations).

> **Note**
> Total runtime of the TRAIN command is SEARCHTIME plus up to twice MAXEVALTIME.

> **Tip**
> Specify a SEARCHTIME that is at least 10x the MAXEVALTIME
>
> This time allotment strikes a reasonable balance between processing time and allowing a variety of model types to be evaluated.

</td></tr>
<tr><td>MAXEVALTIME minutes</td><td>Maximum runtime in minutes per model evaluation.</td></tr>
</table>

Name	Description
	Tip Allot 45 minutes for every 100 MB of training data. This time allotment strikes a reasonable balance between processing time and allowing a variety of model types to be evaluated.
MODEL *model_name*	The name of the model file output by the training process. The model file contains the model best fitted to the training data set. You will input the model to the PREDICT command to generate predictions about a new, unseen data set. Specify *model_name* as a quoted string. For example: `TO "Loan_default_prediction"` You can specify the `*.model` file extension, or let Analytics automatically specify it. By default, the model file is saved to the folder containing the Analytics project. Use either an absolute or relative file path to save the model file to a different, existing folder: ◦ `TO "C:\Loan_default_prediction"` ◦ `TO "ML Train output\Loan_default_prediction.model"`
TO *table_name*	The name of the model evaluation table output by the training process. The model evaluation table contains two distinct types of information: ◦ **Scorer/Metric** - for the classification or regression metrics, quantitative estimates of the predictive performance of the model file output by the training process Different metrics provide different types of estimates. **Scorer** identifies the metric you specified with SCORER. **Metric** identifies the metrics you did not specify. ◦ **Importance/Coefficient** - in descending order, values indicating how much each feature (predictor) contributes to the predictions made by the model Specify *table_name* as a quoted string with a .FIL file extension. For example: `TO "Model_evaluation.FIL"` By default, the table data file (.FIL) is saved to the folder containing the Analytics project. Use either an absolute or relative file path to save the data file to a different, existing folder: ◦ `TO "C:\Model_evaluation.FIL"` ◦ `TO "ML Train output\Model_evaluation.FIL"` **Note** Table names are limited to 64 alphanumeric characters, not including the .FIL extension. The name can include the underscore character (_), but no other special characters, or any spaces. The name cannot start with a number.
IF *test* optional	A conditional expression that must be true in order to process each record. The command is executed on only those records that satisfy the condition.

Name	Description
	Note The IF parameter is evaluated against only the records remaining in a table after any scope parameters have been applied (WHILE, FIRST, NEXT).
WHILE *test* optional	A conditional expression that must be true in order to process each record. The command is executed until the condition evaluates as false, or the end of the table is reached. **Note** If you use WHILE in conjunction with FIRST or NEXT, record processing stops as soon as one limit is reached.
FIRST *range* | NEXT *range* optional	The number of records to process: ◦ **FIRST** - start processing from the first record until the specified number of records is reached ◦ **NEXT** - start processing from the currently selected record until the specified number of records is reached Use *range* to specify the number of records to process. If you omit FIRST and NEXT, all records are processed by default.
FOLDS *number_of_folds*	The number of cross-validation folds to use when evaluating and optimizing the model. Folds are subdivisions of the training data set, and are used in a cross-validation process. Typically, using from 5 to 10 folds yields good results when training a model. The minimum number of folds allowed is 2, and the maximum number is 10. **Tip** Increasing the number of folds can produce a better estimate of the predictive performance of a model, but it also increases overall runtime.
SEED *seed_value* optional	The seed value to use to initialize the random number generator in Analytics. If you omit SEED, Analytics randomly selects the seed value. Explicitly specify a seed value, and record it, if you want to replicate the training process with the same data set in the future.
LINEAR optional	Train and score only linear models. If LINEAR is omitted, all model types relevant to classification or regression are evaluated. **Note** With larger data sets, the training process typically completes more quickly if you include only linear models. Including only linear models guarantees coefficients in the output.

Name	Description
NOFP optional	Exclude feature selection and data preprocessing from the training process. Feature selection is the automated selection of the fields in the training data set that are the most useful in optimizing the predictive model. Automated selection can improve predictive performance, and reduce the amount of data involved in model optimization. Data preprocessing performs transformations such as scaling and standardizing on the training data set to make it better suited for the training algorithms. **Caution** You should only exclude feature selection and data preprocessing if you have a reason for doing so.

Examples

Train a classification model

You want to train a classification model that you can use in a subsequent process to predict which loan applicants will default.

You train the model using a set of historical loan data with a known outcome for each loan, including whether the client defaulted.

In the subsequent prediction process, you will use the model produced by the TRAIN command to process current loan applicant data.

```
OPEN "Loan_applicants_historical"
TRAIN CLASSIFIER ON Age Job_Category Salary Account_Balance Loan_Amount
Loan_Period Refinanced Credit_Score TARGET Default SCORER LOGLOSS
SEARCHTIME 960 MAXEVALTIME 90 MODEL "Loan_default_prediction.model" TO
"Model_evaluation.FIL" FOLDS 5
```

Train a regression model

You want to train a regression model that you can use in a subsequent process to predict the future sale price of houses.

You train the model using a set of recent house sales data, including the sale price.

In the subsequent prediction process, you will use the model produced by the TRAIN command to generate house price evaluations.

```
OPEN "House_sales"
TRAIN REGRESSOR ON Lot_Size Bedrooms Bathrooms Stories Driveway Recroom
Full_Basement Gas_HW Air_conditioning Garage_Places Preferred_Area
TARGET Price SCORER MSE SEARCHTIME 960 MAXEVALTIME 90 MODEL "House_
price_prediction.model" TO "Model_evaluation.FIL" FOLDS 5
```

Remarks

For more information about how this command works, see "Predicting classes and numeric values" on page 1292.

VERIFY command

Checks for data validity errors in one or more fields in an Analytics table by verifying that the data is consistent with the field definitions in the table layout.

Syntax

```
VERIFY {<FIELDS> field_name <...n>|<FIELDS> ALL <EXCLUDE field_name <...n>>}
<IF test> <WHILE test> <FIRST range|NEXT range> <ERRORLIMIT n> <TO
{SCREEN|filename|PRINT}> <APPEND>
```

Parameters

Name	Description
FIELDS *field_name* <...*n*> | FIELDS ALL	The fields or expressions to verify. Specify ALL to verify all fields in the table. **Note** By definition, computed fields, ad hoc expressions, and binary fields are always valid.
EXCLUDE *field_name* optional	Only valid when verifying using FIELDS ALL. The field or fields to exclude from the command. EXCLUDE allows you to fine-tune FIELDS ALL, by excluding the specified fields. EXCLUDE must immediately follow FIELDS ALL. For example: `FIELDS ALL EXCLUDE field_1 field_2`
IF *test* optional	A conditional expression that must be true in order to process each record. The command is executed on only those records that satisfy the condition. **Note** The IF parameter is evaluated against only the records remaining in a table after any scope parameters have been applied (WHILE, FIRST, NEXT).
WHILE *test* optional	A conditional expression that must be true in order to process each record. The command is executed until the condition evaluates as false, or the end of the table

Name	Description
	is reached. **Note** If you use WHILE in conjunction with FIRST or NEXT, record processing stops as soon as one limit is reached.
FIRST *range* | NEXT *range* optional	The number of records to process: ◦ **FIRST** - start processing from the first record until the specified number of records is reached ◦ **NEXT** - start processing from the currently selected record until the specified number of records is reached Use *range* to specify the number of records to process. If you omit FIRST and NEXT, all records are processed by default.
ERRORLIMIT *n* optional	The number of errors allowed before the command is terminated. The default value is 10.
TO SCREEN | *filename* | PRINT optional	The location to send the results of the command to: ◦ **SCREEN** - displays the results in the Analytics display area **Tip** You can click any linked result value in the display area to drill down to the associated record or records in the source table. ◦ ***filename*** - saves the results to a file Specify *filename* as a quoted string with the appropriate file extension. For example: `TO "Output.TXT"` By default, the file is saved to the folder containing the Analytics project. Use either an absolute or relative file path to save the file to a different, existing folder: • `TO "C:\Output.TXT"` • `TO "Results\Output.TXT"` ◦ **PRINT** - sends the results to the default printer
APPEND optional	Appends the command output to the end of an existing file instead of overwriting it. **Note** You must ensure that the structure of the command output and the existing file are identical: • the same fields • the same field order • matching fields are the same length • matching fields are the same data type Analytics appends output to an existing file regardless of its structure. If the structure of the output and the existing file do not match, jumbled, missing, or inaccurate data can result.

Analytics output variables

Name	Contains
WRITE*n*	The total number of data validity errors in all fields verified by the command.

Examples

Verifying data and specifying an error limit

You verify all of the columns in a table and set the error limit to 10. The command stops processing if 10 data validity errors are detected:

```
VERIFY ALL ERRORLIMIT 10 TO "ImportErrors.txt"
```

Remarks

How it works

VERIFY compares the values in one or more fields to the data type specified for each of the fields in the table layout and reports any errors. The command ensures the following:

- **character fields** – contain only valid characters, and that no unprintable characters are present
- **numeric fields** – contain only valid numeric data. In addition to numbers, numeric fields can contain one preceding plus sign or minus sign and one decimal point
- **datetime fields** – contain valid dates, datetimes, or times

For each error that is identified, the record number and field name are output, along with the invalid value in hexadecimal format.

Functions overview

An ACLScript function is a computerized routine in Analytics, narrow in scope, that performs a specific task or calculation and returns a value.

For example, the ALLTRIM( ) function removes any leading or trailing spaces from the text values in a field.

A full list of functions available in Analytics, organized by category, appears on subsequent pages:

Conventions and usage

Function syntax conventions

Convention	Description

parentheses `( )`	◦ Function input values must be enclosed by parentheses:
	`ALLTRIM(Vendor_Name)`
	◦ The opening parenthesis must immediately follow the function name with no intervening space:
	`ALLTRIM(Vendor_Name)`
	not:
	`ALLTRIM (Vendor_Name)`

Convention	Description
	◦ Parentheses must be used even if no input values are being specified: `RECNO( )`

separators	◦ Function input values must be separated by a separator character:
	`SUBSTRING(Product_ID,5,12)`
	◦ Valid separator characters are a blank space, a comma, or a semi-colon.
	The comma or semi-colon separator must be specified in the **List Separator** option on the **Numeric** tab in the **Options** dialog box.
	Tip
	For improved readability you can use both a blank space and one of the other separator characters:
	`SUBSTRING(Product_ID, 5, 12)`

qualifiers	◦ Single or double quotation marks are required around literal character values:
	`EXCLUDE(Product_ID, "#-")`
	◦ Backquotes are required around literal datetime values:
	`AGE(Due_date, `20141231`)`
	◦ No qualifiers are used with numeric values:
	`ABS(-7.2)`
	◦ No qualifiers are used with logical values (`T`/`F`):
	`LEVDIST(Vendor_Name, Vendor_Name_2, F)`
	◦ No qualifiers are used with field names:
	`ALLTRIM(Vendor_Name)`

literal datetime format	◦ Literal date values must be entered in **YYYYMMDD** or **YYMMDD** format: • `` `20141231` ``

Convention	Description
	• `141231` ◦ Literal time values must be entered in **hhmmss** or **hhmm** format, and preceded by a space, `T`, or `t`: • `t235959` • `20141231 2359`

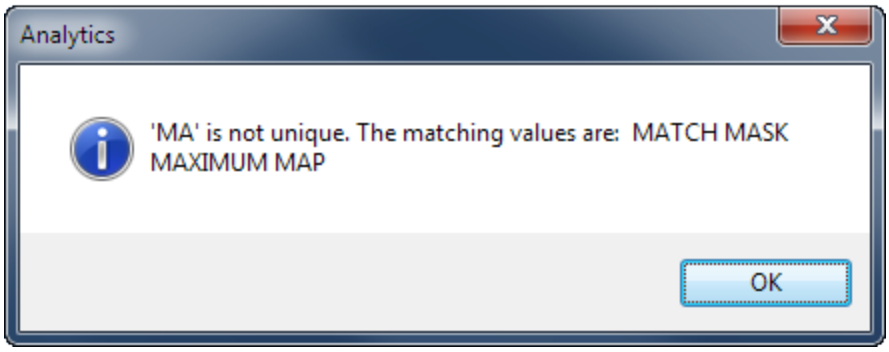
Abbreviating function names

> **Caution**
>
> ACL recommends that you do not abbreviate function names in computed fields, expressions, or scripts, and that you use the full version of each name.
>
> Abbreviation makes computed fields, expressions, or scripts harder to read and to understand. Without complete function names, searching functions in the online help becomes more difficult.
>
> Abbreviation is especially problematic if your Analytics projects or scripts will be modified or inherited by someone else who may not be familiar with the abbreviations.

When specifying functions in computed fields, expressions, or scripts, you can abbreviate their names. You must include enough leading characters from a function name to uniquely identify the function among all Analytics functions.

For example:

- `MAX` uniquely identifies the `MAXIMUM` function and therefore is a valid abbreviation.
- `MA` does not uniquely identify the `MAXIMUM` function and generates an error message.



You can make an abbreviation as short as you want, provided that it still uniquely identifies the function.

For example, all the following abbreviations are valid for the `ALLTRIM` function:

- `ALLTR`
- `ALLT`
- `ALL`
- `AL`

> **Note**
> As abbreviations get shorter they become harder for other users to recognize.

Function documentation conventions

Convention	Used for:
UPPERCASE	The name of the ACLScript function. **Note** Throughout Analytics documentation, function names are presented in uppercase, which is simply a formatting convention. Analytics does not require that functions are entered in uppercase.
italic	User-supplied function parameters.
| (vertical bar)	Separates syntax items enclosed in brackets or braces. You can use only one of the items.
`< >` (angled brackets)	Optional syntax items. Do not type the brackets.
`{ }` (braces)	Required syntax items. Do not type the braces.
`<,...n>`	Indicates the preceding item can be repeated *n* number of times. The occurrences are separated by commas.

Data type terms used in function documentation

The following terms are used to identify the data type of function parameter arguments, and return values:

Term	Means you can use:
Character	Any field name, expression, or variable that belongs to the Analytics Character (C) category, or a string literal
Numeric	Any field name, expression, or variable that belongs to the Analytics Numeric (N) data category, or a numeric value
Datetime	Any field name, expression, or variable that belongs to the Analytics Datetime (D) category, or a datetime literal
Logical	Any field name, expression, or variable that belongs to the Analytics Logical (L) category, or a logical value

Term	Means you can use:
Field	The name of a field from any Analytics data category

Search, replace

Search functions let you perform different types of searches. You can search data for specific words or sequences of characters, values within a range, blank values, and values matching a pattern.

Replace functions give you different options for searching and replacing data.

> **Tip**
> For numerous examples of using functions to perform powerful and effective searching and filtering of data in tables, see "Search and filter using Analytics functions" on page 1173.

Function descriptions

Function	Description
AT( )	Returns a number specifying the starting location of a particular occurrence of a substring within a character value.
BETWEEN( )	Returns a logical value indicating whether the specified value falls within a range.
CLEAN( )	Replaces the first invalid character in a string, and all subsequent characters, with blanks.
FIND( )	Returns a logical value indicating whether the specified string is present in a particular field, or anywhere in an entire record.
FINDMULTI( )	Returns a logical value indicating whether any string in a set of one or more specified strings is present in a particular field, or anywhere in an entire record.
ISBLANK( )	Returns a logical value indicating whether the input value is blank.
MAP( )	Returns a logical value indicating if a character string matches a specified format string containing wildcard characters, literal characters, or both.
MATCH( )	Returns a logical value indicating whether the specified value matches any of the values it is compared against.
OCCURS( )	Returns a count of the number of times a substring occurs in a specified character value.
REGEXFIND( )	Returns a logical value indicating whether the pattern specified by a regular expression

Function	Description
	occurs in a string.
REGEXREPLACE( )	Replaces all instances of strings matching a regular expression with a new string.
REPLACE( )	Replaces all instances of a specified character string with a new character string.
TEST( )	Returns a logical value indicating whether a specified string occurs at a specific byte position in a record.

Comparison

Comparison functions provide different ways to find text values that are identical, or nearly identical, to a specified value.

> **Tip**
> If you want to find identical text values only, you can also use this simple method:
> `field_name = "text value"`
> For example: `last_name = "Smith"`
> The *text value* is case-sensitive.

Function descriptions

Function	Description
DICECOEFFICIENT( )	Returns the Dice's coefficient of two specified strings, which is a measurement of how similar the two strings are.
ISFUZZYDUP( )	Returns a logical value indicating whether a string is a fuzzy duplicate of a comparison string.
LEVDIST( )	Returns the Levenshtein distance between two specified strings, which is a measurement of how much the two strings differ.
SOUNDEX( )	Returns the soundex code for the specified string, which can be used for phonetic comparisons with other strings.
SOUNDSLIKE( )	Returns a logical value indicating whether a string phonetically matches a comparison string.

Conversion

Conversion functions allow you to convert between data types. One important use of these functions is to prepare fields for Analytics commands that require input of a specific data type.

Function descriptions

Function	Description
BINTOSTR( )	Returns Unicode character data converted from ZONED or EBCDIC character data. Abbreviation for "Binary to String".
CTOD( )	Converts a character or numeric date value to a date. Can also extract the date from a character or numeric datetime value and return it as a date. Abbreviation for "Character to Date".
CTODT( )	Converts a character or numeric datetime value to a datetime. Abbreviation for "Character to Datetime".
CTOT( )	Converts a character or numeric time value to a time. Can also extract the time from a character or numeric datetime value and return it as a time. Abbreviation for "Character to Time".
DATE( )	Extracts the date from a specified date or datetime and returns it as a character string. Can also return the current operating system date.
DATETIME( )	Converts a datetime to a character string. Can also return the current operating system datetime.
DTOU( )	Converts an Analytics date value to a Unicode string in the specified language and locale format. Abbreviation for "Date to Unicode".
EBCDIC( )	Returns a string that has been converted to EBCDIC character encoding.
HASH( )	Returns a salted cryptographic hash value based on the input value.
LEADINGZEROS( )	Adds leading zeros to a character string or a number.
PACKED( )	Returns numeric data converted to the Packed data type.
STOD( )	Converts a serial date - that is, a date expressed as an integer - to a date value. Abbreviation for "Serial to Date".
STODT( )	Converts a serial datetime - that is, a datetime expressed as an integer, and a fractional portion of 24 hours - to a datetime value. Abbreviation for "Serial to Datetime".
STOT( )	Converts a serial time - that is, a time expressed as a fractional portion of 24 hours, with 24 hours equaling 1 - to a time value. Abbreviation for "Serial to Time".

Function	Description
STRING( )	Converts a numeric value to a character string.
TIME( )	Extracts the time from a specified time or datetime and returns it as a character string. Can also return the current operating system time.
UNSIGNED( )	Returns numeric data converted to the Unsigned data type.
UTOD( )	Converts a Unicode string containing a formatted date to an Analytics date value. Abbreviation for "Unicode to Date".
VALUE( )	Converts a character string to a numeric value.
ZONED( )	Converts numeric data to character data and adds leading zeros to the output.

Text

Text functions let you perform a variety of different tasks with character data.

For example, you can remove leading or trailing spaces, exclude or include specific characters, isolate just a portion of a character string, or standardize upper or lowercase.

Function descriptions

Function	Description
ALLTRIM( )	Returns a string with leading and trailing spaces removed from the input string.
BINTOSTR( )	Returns Unicode character data converted from ZONED or EBCDIC character data. Abbreviation for "Binary to String".
BLANKS( )	Returns a string containing a specified number of blank spaces.
DATETIME( )	Converts a datetime to a character string. Can also return the current operating system datetime.
DTOU( )	Converts an Analytics date value to a Unicode string in the specified language and locale format. Abbreviation for "Date to Unicode".
EBCDIC( )	Returns a string that has been converted to EBCDIC character encoding.
EXCLUDE( )	Returns a string that excludes the specified characters.
INCLUDE( )	Returns a string that includes only the specified characters.
INSERT( )	Returns the original string with specified text inserted at a specific byte location.

Function	Description
LAST( )	Returns a specified number of characters from the end of a string.
LEADINGZEROS( )	Adds leading zeros to a character string or a number.
LENGTH( )	Returns the number of characters in a string.
LOWER( )	Returns a string with alphabetic characters converted to lowercase.
LTRIM( )	Returns a string with leading spaces removed from the input string.
OMIT( )	Returns a string with one or more specified substrings removed.
PROPER( )	Returns a string with the first character of each word set to uppercase and the remaining characters set to lowercase.
REMOVE( )	Returns a string that includes only the specified characters.
REPEAT( )	Returns a string that repeats a substring a specified number of times.
REVERSE( )	Returns a string with the characters in reverse order.
RJUSTIFY( )	Returns a right-justified string the same length as a specified string, with any trailing spaces moved to the left of the string.
SORTWORDS( )	Returns a string with individual words sorted in sequential order.
SPLIT( )	Returns a specified segment from a string.
STRING( )	Converts a numeric value to a character string.
SUBSTR( )	Returns a specified substring from a string.
TRANSFORM( )	Reverses the display order of bi-directional text within a specified string.
TRIM( )	Returns a string with trailing spaces removed from the input string.
UPPER( )	Returns a string with alphabetic characters converted to uppercase.
ZONED( )	Converts numeric data to character data and adds leading zeros to the output.

Math

Math functions perform a variety of different mathematical calculations.

Function descriptions

Function	Description
ABS( )	Returns the absolute value of a numeric expression. The absolute value of a number is the number without its sign.
COS( )	Returns the cosine of an angle expressed in radians, with a precision of 15 decimal places.
DEC( )	Returns a value, or the result of a numeric expression, with the specified number of decimal places.
EXP( )	Returns the exponential value (base 10) of a numeric expression with a specified number of decimal places.
FREQUENCY( )	Returns the expected Benford frequency for sequential leading positive numeric digits to a precision of eight decimal places.
INT( )	Returns the integer value of a numeric expression or field value.
LEADING( )	Returns a string containing a specified number of leading digits.
LOG( )	Returns the logarithm (base 10) of a numeric expression or field value with a specified number of decimal places.
MAXIMUM( )	Returns the maximum value in a set of numeric values, or the most recent value in a set of datetime values.
MINIMUM( )	Returns the minimum value in a set of numeric values, or the oldest value in a set of datetime values.
MOD( )	Returns the remainder from dividing two numbers.
NORMDIST( )	Returns the probability that a random variable from a normally distributed data set is less than or equal to a specified value, or exactly equal to a specified value.
NORMSINV( )	Returns the z-score associated with a specified probability in a standard normal distribution. The z-score is the number of standard deviations a value is from the mean of a standard normal distribution.
PI( )	Returns the value of pi to 15 decimal places.
RAND( )	Returns a random number that falls within a specified boundary.
ROOT( )	Returns the square root of a numeric expression.
ROUND( )	Returns a rounded whole number for a numeric value.
SIN( )	Returns the sine of an angle expressed in radians, with a precision of 15 decimal places.

Function	Description
TAN( )	Returns the tangent of an angle expressed in radians, with a precision of 15 decimal places.
VALUE( )	Converts a character string to a numeric value.
ZONED( )	Converts numeric data to character data and adds leading zeros to the output.
ZSTAT( )	Returns the standard Z-statistic.

Date and time

Date and time functions let you perform a variety of different tasks with date, datetime, or time data.

For example, you can calculate the number of days between two dates, extract portions of a date, such as just the month, or find out the day of the week that corresponds to each date.

Additional information about date and time functions

Date and time functions can sometimes be challenging to use correctly. The ACLScript language reference describes the specific details of how each function works. For information about some general considerations when using date and time functions, see the following topics:

- "Using datetimes in expressions" on page 814
- "Serial datetimes" on page 827
- "How UTC offsets affect datetime expressions" on page 830

Function descriptions

Function	Description
AGE( )	Returns the number of elapsed days (the age) between a specified date and a specified cutoff date, or the current operating system date, or the number of elapsed days between any two dates.
CDOW( )	Returns the name of the day of the week for a specified date or datetime. Abbreviation for "Character Day of Week".
CMOY( )	Returns the name of the month of the year for a specified date or datetime. Abbreviation for "Character Month of Year".
CTOD( )	Converts a character or numeric date value to a date. Can also extract the date from a character or numeric datetime value and return it as a date. Abbreviation for "Character to Date".

Function	Description
CTODT( )	Converts a character or numeric datetime value to a datetime. Abbreviation for "Character to Datetime".
CTOT( )	Converts a character or numeric time value to a time. Can also extract the time from a character or numeric datetime value and return it as a time. Abbreviation for "Character to Time".
DATE( )	Extracts the date from a specified date or datetime and returns it as a character string. Can also return the current operating system date.
DATETIME( )	Converts a datetime to a character string. Can also return the current operating system datetime.
DAY( )	Extracts the day of the month from a specified date or datetime and returns it as a numeric value (1 to 31).
DOW( )	Returns a numeric value (1 to 7) representing the day of the week for a specified date or datetime. Abbreviation for "Day of Week".
EOMONTH( )	Returns the date of the last day of the month that is the specified number of months before or after a specified date.
GOMONTH( )	Returns the date that is the specified number of months before or after a specified date.
HOUR( )	Extracts the hour from a specified time or datetime and returns it as a numeric value using the 24-hour clock.
MAXIMUM( )	Returns the maximum value in a set of numeric values, or the most recent value in a set of datetime values.
MINIMUM( )	Returns the minimum value in a set of numeric values, or the oldest value in a set of datetime values.
MINUTE( )	Extracts the minutes from a specified time or datetime and returns it as a numeric value.
MONTH( )	Extracts the month from a specified date or datetime and returns it as a numeric value (1 to 12).
NOW( )	Returns the current operating system time as a Datetime data type.
SECOND( )	Extracts the seconds from a specified time or datetime and returns it as a numeric value.
STOD( )	Converts a serial date - that is, a date expressed as an integer - to a date value. Abbreviation for "Serial to Date".
STODT( )	Converts a serial datetime - that is, a datetime expressed as an integer, and a fractional portion of 24 hours - to a datetime value. Abbreviation for "Serial to Datetime".
STOT( )	Converts a serial time - that is, a time expressed as a fractional portion of 24 hours, with 24 hours equaling 1 - to a time value. Abbreviation for "Serial to Time".

Function	Description
TIME( )	Extracts the time from a specified time or datetime and returns it as a character string. Can also return the current operating system time.
TODAY( )	Returns the current operating system date as a Datetime data type.
UTOD( )	Converts a Unicode string containing a formatted date to an Analytics date value. Abbreviation for "Unicode to Date".
WORKDAY( )	Returns the number of workdays between two dates.
YEAR( )	Extracts the year from a specified date or datetime and returns it as a numeric value using the YYYY format.

Financial

Financial functions perform a variety of different calculations associated with annuities, loans, investments, principal, interest, and payments.

> **Note**
>
> Beginning with Analytics 12.0, a change made by Microsoft to its Visual C++ Redistributable Package, an Analytics prerequisite, causes the results of some Analytics financial functions to differ slightly from the results in previous versions of Analytics.
>
> The Visual C++ change was made by Microsoft to improve computational precision. As a result, rounding in Analytics functions such as PMT( ) and FVSCHEDULE( ) now behaves differently.

Function descriptions

Function	Description
CUMIPMT( )	Returns the cumulative interest paid on a loan during a range of periods.
CUMPRINC( )	Returns the cumulative principal paid on a loan during a range of periods.
EFFECTIVE( )	Returns the effective annual interest rate on a loan.
FVANNUITY( )	Returns the future value of a series of payments calculated using a constant interest rate. Future value is the sum of the payments plus the accumulated compound interest.
FVLUMPSUM( )	Returns the future value of a current lump sum calculated using a constant interest rate.
FVSCHEDULE( )	Returns the future value of a current lump sum calculated using a series of interest rates.

Function	Description
IPMT( )	Returns the interest paid on a loan for a single period.
NOMINAL( )	Returns the nominal annual interest rate on a loan.
NPER( )	Returns the number of periods required to pay off a loan.
PMT( )	Returns the amount of the periodic payment (principal + interest) required to pay off a loan.
PPMT( )	Returns the principal paid on a loan for a single period.
PVANNUITY( )	Returns the present value of a series of future payments calculated using a constant interest rate. Present value is the current, lump-sum value.
PVLUMPSUM( )	Returns the present value required to generate a specific future lump sum calculated using a constant interest rate. Present value is the current, lump-sum value.
RATE( )	Returns the interest rate per period.

Field and record

Field and record functions perform a variety of different tasks with the basic components that make up Analytics tables.

For example, you can test whether a field exists, find out the data type of a field, and capture record numbers. Field and record functions can perform useful helper tasks during data analysis in an Analytics script.

Function descriptions

Function	Description
FTYPE( )	Returns a character identifying the data category of a field or variable, or the type of an Analytics project item.
HASH( )	Returns a salted cryptographic hash value based on the input value.
ISDEFINED( )	Returns **T** (true) if the specified field or variable is defined, and **F** (false) otherwise.
OFFSET( )	Returns the value of a field with the starting position offset by a specified number of bytes.
RECLEN( )	Returns the length of the current record.
RECNO( )	Returns the current record number.

Function	Description
RECOFFSET( )	Returns a field value from a record that is a specified number of records from the current record.
VERIFY( )	Returns a logical value indicating whether the data in a physical data field is valid.

Table, file, and project

The table, file, and project functions perform helper tasks that can be useful during data analysis in an Analytics script.

For example, you can use the FTYPE( ) function to identify the data category of a field, which you may need to know in order to correctly apply other functions or commands to the field.

Function descriptions

Function	Description
FILESIZE( )	Returns the size of the specified file in bytes or -1 if the file does not exist.
FTYPE( )	Returns a character identifying the data category of a field or variable, or the type of an Analytics project item.
GETOPTIONS( )	Returns the current setting for the specified Analytics option (**Options** dialog box setting).
PROPERTIES( )	Returns properties information for the specified Analytics project item.

Variable testing

The variable testing functions tell you the data type of a variable, and whether a variable exists.

Function descriptions

Function	Description
FTYPE( )	Returns a character identifying the data category of a field or variable, or the type of an Analytics project item.
ISDEFINED( )	Returns **T** (true) if the specified field or variable is defined, and **F** (false) otherwise.

Python

ACLScript's Python functions incorporate the result of a calculation performed using the Python programming language into an Analytics script.

To use ACLScript's Python functions, you must install and configure a compatible version of Python on the computer where the Analytics script will run. For more information, see "Configuring Python for use with Analytics" on page 2601.

Function descriptions

Function	Description
PYDATE( )	Returns a date value calculated by a function in a external Python script. Data processing in Python is external to Analytics.
PYDATETIME( )	Returns a datetime value calculated by a function in an external Python script. Data processing in Python is external to Analytics.
PYLOGICAL( )	Returns a logical value calculated by a function in an external Python script. Data processing in Python is external to Analytics.
PYNUMERIC( )	Returns a numeric value calculated by a function in an external Python script. Data processing in Python is external to Analytics.
PYSTRING( )	Returns a character value calculated by a function in an external Python script. Data processing in Python is external to Analytics.
PYTIME( )	Returns a time value calculated by a function in an external Python script. Data processing in Python is external to Analytics.

R

ACLScript's R functions incorporate the result of a calculation performed using the R programming language into an Analytics script.

To use ACLScript's R functions, you must install a compatible version of R on the computer where the Analytics script will run. For more information, see "ACL for Windows system requirements" on page 2611.

Function descriptions

Function	Description
RDATE( )	Returns a date value calculated by an R function or script. Data processing in R is external to Analytics.
RDATETIME( )	Returns a datetime value calculated by an R function or script. Data processing in R is external to Analytics.
RLOGICAL( )	Returns a logical value calculated by an R function or script. Data processing in R is external to Analytics.
RNUMERIC( )	Returns a numeric value calculated by an R function or script. Data processing in R is external to Analytics.
RSTRING( )	Returns a string value calculated by an R function or script. Data processing in R is external to Analytics.
RTIME( )	Returns a time value calculated by an R function or script. Data processing in R is external to Analytics.

Bit and character encoding

Bit and character encoding functions provide a set of tools for discovering and manipulating data at the level of bits, bytes, and character encoding.

Function descriptions

Function	Description
ASCII( )	Returns the ASCII code for a specified character.
BIT( )	Returns the binary representation for the specified byte position in the current record as an eight character string.
BYTE( )	Returns the character stored in the specified byte position in the current record.
CHR( )	Returns the character associated with the specified ASCII code.
DBYTE( )	Returns the Unicode character located at the specified byte position in a record.
DHEX( )	Converts a Unicode string to a hexadecimal string.
DIGIT( )	Returns the upper or lower digit of a specified Packed data type byte.

Function	Description
HEX( )	Converts an ASCII string to a hexadecimal string.
HTOU( )	Converts a hexadecimal string to a Unicode string. Abbreviation for "Hexadecimal to Unicode".
MASK( )	Performs a bitwise AND operation on the first bytes of two character strings.
SHIFT( )	Returns a single character string with the bits in the first character of the input value shifted to the left or right.

ABS( ) function

Returns the absolute value of a numeric expression. The absolute value of a number is the number without its sign.

Syntax

```
ABS(number)
```

Parameters

Name	Type	Description
number	numeric	The value to find the absolute value for.

Output

Numeric.

Examples

Basic examples

Returns 7.2:

```
ABS(7.2)
```

Returns 7.2:

```
ABS(-7.2)
```

AGE( ) function

Returns the number of elapsed days (the age) between a specified date and a specified cutoff date, or the current operating system date, or the number of elapsed days between any two dates.

Syntax

```
AGE(date/datetime/string <,cutoff_date>)
```

Parameters

Name	Type	Description
date/datetime/string	character datetime	The field, expression, or literal value to age.
cutoff_date optional	character datetime	The field, expression, or literal value to which *date/datetime/string* is compared. If omitted, the current operating system date is used as the cutoff date.

> **Note**
> *date/datetime/string* and *cutoff_date* can both accept a datetime value. You cannot use AGE( ) with time values alone.
> For more information, see "Using AGE( ) with datetime data" on page 2038.

Output

Numeric.

Examples

Basic examples

No cutoff date

Returns the number of days between 31 Dec 2014 and the current date:

- If a positive value is returned, it is equal to the number of days in the past December 31, 2014 occurred
- If a negative value is returned, it is equal to the number of days in the future December 31, 2014 occurs
- If 0 is returned, December 31, 2014 is the current date

```
AGE(`20141231`)
```

Returns the number of days between each date in the **Due_date** field and the current date:

```
AGE(Due_date)
```

Mixing data types

Returns 518, the number of days between the two specified dates:

```
AGE(`20130731`,`20141231`)
```

```
AGE("20130731","20141231")
```

```
AGE(`20130731`,"20141231")
```

```
AGE(`20130731 235959`,`20141231`)
```

Using cutoff dates and fields

Returns the number of days between each date in the **Due_date** field and the cutoff date of December 31, 2014:

- Dates prior to the cutoff date return a positive value equal to the number of days before the cutoff day they occur
- Dates after the cutoff date return a negative value equal to the number of days after the cutoff day they occur

```
AGE(Due_date, `20141231`)
```

Returns the number of days between December 31, 2014 and each date in the **Due_date** field. Results are the same as the example immediately above, but the sign of the returned values (positive or negative) is reversed:

```
AGE(`20141231`, Due_date)
```

Comparing dates in fields

Returns the number of days between each date in the **Payment_date** field and a corresponding date in the **Due_date** field:

- Payment dates prior to due dates return a positive value, indicating timely payment
- Payment dates after due dates return a negative value, indicating late payment

```
AGE(Payment_date, Due_date)
```

Returns the number of days between each date in the **Payment_date** field and a corresponding date in the **Due_date** field plus a grace period of 15 days.

- Payment dates prior to due dates, or up to 15 days after due dates, return a positive value
- Payment dates more than 15 days after due dates return a negative value, indicating late payment outside the grace period

```
AGE(Payment_date, Due_date+15)
```

Advanced examples

Extracting overdue payments

Extract the name, amount, and invoice date for each record where the age of the invoice is greater than 180 days, based on a cutoff date of December 31, 2014:

```
EXTRACT FIELDS Name Amount Invoice_Date TO "Overdue" IF AGE(Invoice_
Date,`20141231`) > 180
```

Remarks

How it works

The AGE( ) function calculates the number of days between two dates.

When to use AGE( )

Use AGE( ) to compare two dates to determine overdue accounts, to perform aged analyses of balances, or to perform any task that requires the number of elapsed days between two dates.

Negative return values

A negative value is returned if the date specified for *date/datetime/string* is more recent than the date specified as the *cutoff_date*, or the operating system date if no *cutoff_date* is specified.

Returns -518:

```
AGE(`20141231`, `20130731`)
```

If you want the elapsed number of days between two dates to always be a positive number, regardless of which date is more recent, nest the AGE( ) function inside the ABS( ) function.

Returns 518:

```
ABS(AGE(`20141231`, `20130731`))
```

Using AGE( ) with datetime data

The AGE( ) function can accept datetime data in one or both parameters. However, you need to be careful if the time portion of the data includes a UTC offset (timezone indicator).

Datetime data without a UTC offset

The time portion of a datetime value does not affect the date calculation performed by AGE( ) if the time data does not include a UTC offset.

Datetime data with a UTC offset

The time portion of a datetime value can affect the date calculation performed by AGE( ) if the time data in one or both parameters includes a UTC offset. Analytics automatically reconciles the UTC offset before performing the calculation, which can cause the result to change by 1 day if reconciliation moves the time forward or backward through the boundary of midnight.

For more information, see "How UTC offsets affect datetime expressions" on page 830.

Using a field for the cutoff date

Unlike the AGE command, which requires a literal date value for the cutoff date, the AGE( ) function allows you to use a field for the cutoff date.

For example:

```
AGE(Payment_date, Due_date)
```

Using the AGE( ) function in this manner is equivalent to calculating the difference between two date fields by subtracting them in an expression.

For example:

```
Due_date – Payment_date
```

Parameter details

A datetime field specified for *date/datetime/string* or *cutoff_date* can use any date or datetime format, as long as the field definition correctly defines the format.

Specifying a literal date or datetime value

When specifying a literal date or datetime value for *date/datetime/string* or *cutoff_date*, you are restricted to the formats in the table below, and you must enclose the value in backquotes, or single or double quotation marks – for example, `` `20141231` `` or `"20141231"`

Do not use any separators such as slashes (/) or colons (:) between the individual components of dates or times. Colons are permitted in character time values.

- **Datetime values** – you can use any combination of the date, separator, and time formats listed in the table below. The date must precede the time, and you must use a separator between the two. Valid separators are a single blank space, the letter 't', or the letter 'T'.

Example formats	Example literal values
YYYYMMDD	`20141231` "20141231"
YYMMDD	`141231` "141231"
YYYYMMDD hhmmss	`20141231 235959` "20141231 235959"
YYMMDDthhmm	`141231t2359` "141231t2359"
YYYYMMDDThh	`20141231T23` "20141231T23"
YYYYMMDD hhmmss+/-hhmm (UTC offset)	`20141231 235959-0500` "20141231 235959-0500"
YYMMDD hhmm+/-hh (UTC offset)	`141231 2359+01` "141231 2359+01"
Note Do not use hh alone in the main time format with data that has a UTC offset. For example, avoid: hh+hhmm. Results can be unreliable.	

ALLTRIM( ) function

Returns a string with leading and trailing spaces removed from the input string.

Syntax

```
ALLTRIM(string)
```

Parameters

Name	Type	Description
string	character	The field, expression, or literal value to remove leading and trailing spaces from.

Output

Character.

Examples

Basic examples

Returns "Vancouver":

```
ALLTRIM("   Vancouver   ")
```

Returns "New York":

```
ALLTRIM("   New York   ")
```

Advanced examples

Concatenating character fields

Use ALLTRIM( ) to eliminate spaces when you concatenate character fields, such as first name and last name fields, so that the resulting field does not contain multiple blank spaces between the concatenated values.

```
DEFINE FIELD Full_Name COMPUTED ALLTRIM(First_Name) + " " + ALLTRIM
(Last_Name)
```

Removing non-breaking spaces

Non-breaking spaces are not removed by the ALLTRIM( ) function.

If you need to remove leading or trailing non-breaking spaces, create a computed field using the following expression:

```
DEFINE FIELD Description_cleaned COMPUTED ALLTRIM(REPLACE(Description,
CHR(160), CHR(32)))
```

The REPLACE( ) function replaces any non-breaking spaces with regular spaces, and then ALLTRIM( ) removes any leading or trailing regular spaces.

Remarks

How it works

The ALLTRIM( ) function removes the leading and trailing spaces from a string. Spaces inside the string are not removed.

Related functions

Use the LTRIM( ) function if you want to remove only leading spaces from a string, or the TRIM( ) function if you want to remove only trailing spaces.

ASCII( ) function

Returns the ASCII code for a specified character.

Syntax

```
ASCII(character)
```

Parameters

Name	Type	Description
character	character	The character to identify the ASCII code for. You can specify a quoted character, or a multi-character string, field, or expression. If you specify multiple characters, only the first character is evaluated.

Output

Numeric.

Examples

Basic examples

Returns 65:

```
ASCII("A")
```

Returns 49:

```
ASCII("1")
```

Advanced examples

Extracting a record that starts with a tab character

Extract records that have a tab character at the beginning of a field called "Description". The ASCII code for a tab character is "9".

```
EXTRACT RECORD TO "Tab_Entries.acl" IF ASCII(Description) = 9
```

Remarks

Testing for non-printable characters

You can use ASCII( ) to test for non-printable characters such as:

- **Nul** - ASCII "0"
- **Tab** - ASCII "9"
- **Line Feed (LF)** - ASCII "10"
- **Carriage Return (CR)** - ASCII "13"

Related functions

ASCII( ) is the inverse of the CHR( ) function.

AT( ) function

Returns a number specifying the starting location of a particular occurrence of a substring within a character value.

Syntax

```
AT(occurrence_num, search_for_string, within_text)
```

Parameters

Name	Type	Description
occurance_num	numeric	The occurrence (instance) of *search_for_string* to return the location of. For example, specify `1` to return the starting location of the first occurrence of *search_for_string*.
search_for_string	character	The substring to search for in *within_text*. This value is case-sensitive. If *search_for_string* includes double quotation marks, you need to enclose the value in single quotation marks: `AT(1,'"test"', Description)`
within_text	character	The value to search in. You can concatenate two or more fields in the *within_text* parameter if you want to search in more than one field in a table: `AT(1,'"test"', Description+Summary)`

Output

Numeric. Returns the starting byte position of the specified occurrence of the *search_for_string* value, or 0 if no matches are found.

Examples

Basic examples

Occurrences found

Returns 4:

```
AT(1, "-", "604-669-4225")
```

Returns 8:

```
AT(2, "-", "604-669-4225")
```

Occurrences not found

Returns 0, because there is not a third hyphen in the value:

```
AT(3, "-", "604-669-4225")
```

Returns 0, because there is not a fourth lowercase "a" in the value:

```
AT(4, "a", "Alabama")
```

Groups of characters

Returns 5:

```
AT(2, "iss", "Mississippi")
```

Searching a field

Returns the byte position of the first hyphen in each value in the **Invoice_Number** field:

```
AT(1, "-", Invoice_Number)
```

Advanced examples

Finding invoice numbers in which the second hyphen occurs after the tenth byte position

You can analyze the consistency of invoice numbers in a table by using the AT( ) function to create a filter like the one below. This filter isolates all records in which the invoice number contains two or more hyphens, and the second hyphen occurs after the tenth byte position:

```
SET FILTER TO AT(2, "-", Invoice_Number) > 10
```

Remarks

When to use AT( )

Use this function to retrieve the following starting positions within a character value:

- the starting position of a substring
- the starting position of a subsequent occurrence of the substring

If you only want to confirm multiple occurrences of the same substring in a field, the OCCURS( ) function is a better alternative. For more information, see "OCCURS( ) function" on page 2263.

Return value when *occurrence_num* exceeds the number of occurrences

If *occurrence_num* is greater than the actual number of substring occurrences in *within_text*, the function returns 0 because it cannot find that occurrence of the substring.

Concatenated fields and return values

When you search in more than one field, the value returned for the instance is the starting location of *search_for_string* across all fields that you specify. The concatenated fields are treated like a single field that includes leading and trailing spaces from the individual fields, unless you use the ALLTRIM ( ) function to remove spaces.

For example, if you search for the first occurrence of a string in two fields with a width of eight characters each, and the string is found at the beginning of the second field, the return value is 9.

BETWEEN( ) function

Returns a logical value indicating whether the specified value falls within a range.

Syntax

```
BETWEEN(value, min, max)
```

Parameters

Name	Type	Description
value	character numeric datetime	The field, expression, or literal value to test.
min	character numeric datetime	The minimum value of the range. Can be a field, expression, or literal value.
max	character numeric datetime	The maximum value of the range. Can be a field, expression, or literal value.

> **Note**
> The range evaluating to **T** (true) includes the *min* and *max* values.
> For information regarding character ranges, see "SET EXACT behavior" on page 2050.

Output

Logical. Returns **T** (true) if *value* is greater than or equal to the *min* value, and less than or equal to the *max* value. Returns **F** (false) otherwise.

Examples

Basic examples

Numeric input

Returns T:

```
BETWEEN(500,400,700)
```

Returns F:

```
BETWEEN(100,400,700)
```

Character input

Returns T:

```
BETWEEN("B","A","C")
```

Returns F, because the character comparison is case-sensitive, and lowercase "b" does not fall between uppercase "A" and "C":

```
BETWEEN("b","A","C")
```

Datetime input

Returns T:

```
BETWEEN(`141230`,`141229`,`141231`)
```

Returns T for all values in the **Login_time** field from 07:00:00 AM to 09:00:00 AM, inclusive, and F otherwise:

```
BETWEEN(Login_time,`t070000`,`t090000`)
```

SET EXACT behavior

Returns T for all values in the **Last_Name** field that begin with the letters from "C" to "K", inclusive, and F otherwise (SET EXACT must be OFF):

```
BETWEEN(Last_Name, "C", "K")
```

Returns T for all values in the **Last_Name** field that begin with the letters from "C" to "J", inclusive, and F otherwise (SET EXACT must be ON). Also returns T for the single letter "K":

```
BETWEEN(Last_Name, "C", "K")
```

Field input

Returns T for all values in the **Invoice_Date** field from 30 Sep 2014 to 30 Oct 2014, inclusive, and F otherwise:

```
BETWEEN(Invoice_Date, `20140930`, `20141030`)
```

Returns T for all records in which the invoice date does not fall between the PO date and the paid date, inclusive, and F otherwise:

```
NOT BETWEEN(Invoice_Date, PO_Date, Paid_Date)
```

Returns T for all values in the **Invoice_Amount** field from $1000 to $5000, inclusive, and F otherwise:

```
BETWEEN(Invoice_Amount, 1000, 5000)
```

Advanced examples

Create a filter to view a salary range

The following example opens the **Employee_List** sample table and applies a filter that limits the records displayed to include only employees that earn a salary greater than or equal to $40,000.00, and less than or equal to $50,000.00.

```
OPEN Employee_List
SET FILTER TO BETWEEN(Salary, 40000.00, 50000.00)
```

Create a filter to find dates within a range that changes

You have created a table that joins data from your company's travel and expense system with company credit card data. You want to find any instances where an employee was reimbursed for a hotel room charge that was also charged to the company credit card.

You join the two sets of data on the Amount field and plan to use the dates of the hotel stay and the T&E expense date to confirm that the two amounts refer to the same expense. The problem is that the date in the T&E system can differ by a day or two from the hotel dates in the company credit card data.

The example below opens the **Joined_expense_data** table and applies a filter that finds T&E dates within a range of hotel room dates. By using fields rather than actual date values, the ranges shift with the data.

```
OPEN Joined_expense_data
SET FILTER TO BETWEEN(T_E_date, Arrival_date-2, Arrival_date+2) OR
BETWEEN(T_E_date, Departure_date-2, Departure_date+2)
```

Remarks

Supported data types

Inputs to the BETWEEN( ) function can be numeric, character, or datetime data. You cannot mix data types. All three inputs must belong to the same data category.

Use BETWEEN( ) instead of the AND operator

You can use the BETWEEN( ) function instead of expressions that use the AND operator.

For example:

```
BETWEEN(Invoice_Amount, 1000, 5000)
```

is equivalent to

```
Invoice_Amount >= 1000 AND Invoice_Amount <= 5000
```

The order of *min* and *max*

The order of *min* and *max* in the BETWEEN( ) function does not matter because Analytics automatically identifies which value is the minimum and which value is the maximum.

Both of the examples below return T:

```
BETWEEN(2500, 1000, 5000)
```

```
BETWEEN(2500, 5000, 1000)
```

Decimal precision of numeric inputs

When the numeric inputs being compared have a different decimal precision, the comparison uses the higher level of precision.

Returns T, because 1.23 is equal to 1.23:

```
BETWEEN(1.23, 1.23, 1.25)
```

Returns F, because 1.23 is less than 1.234 once the third decimal place is considered:

```
BETWEEN(1.23, 1.234, 1.25)
```

Character data

Case sensitivity

The BETWEEN( ) function is case-sensitive when used with character data. When it compares characters, "a" is not equivalent to "A".

Returns F:

```
BETWEEN("B","a","C")
```

If you are working with data that includes inconsistent case, you can use the UPPER( ) function to convert the values to consistent case before using BETWEEN( ).

Returns T:

```
BETWEEN(UPPER("B"), UPPER("a"), UPPER("C"))
```

Partial matching

Partial matching is supported for character comparisons.

value can be contained by *min*.

Returns T, even though *value* "AB" appears to be less than *min* "ABC":

```
BETWEEN("AB", "ABC", "Z")
```

max can be contained by *value*.

Returns T, even though *value* "ZZ" appears to be greater than *max* "Z":

```
BETWEEN("ZZ", "ABC", "Z")
```

> **Note**
> The shorter value in the character comparison must appear at the start of the longer value to constitute a match.

Partial matching and SET EXACT

Partial matching is enabled when SET EXACT = OFF, which is the Analytics default setting. If SET EXACT = ON, partial matching is disabled and the comparison values must exactly match to constitute a match.

Both examples above are False when SET EXACT is ON.

For more information about SET EXACT (the **Exact Character Comparisons** option), see "SET command" on page 1960.

Turning SET EXACT off or on

If you want to ensure that the **Exact Character Comparisons** option is not used with the BETWEEN ( ) function, check that the option is deselected in the **Table** tab in the **Options** dialog box (**Tools > Options**).

If you are using a script, you can add the `SET EXACT OFF` command before the BETWEEN( ) function appears. If required, you can restore the previous state with the `SET EXACT ON` command.

Datetime parameters

A date, datetime, or time field specified as a function input can use any date, datetime, or time format, as long as the field definition correctly defines the format.

Mixing date, datetime, and time inputs

You are not prevented from mixing date, datetime, and time values in the BETWEEN( ) function's three inputs, but mixing these Datetime subtypes can give results that are not meaningful.

Analytics uses serial number equivalents to process datetime calculations, so even if you are interested in only the date portion of a datetime value, the time portion still forms part of the calculation.

Consider the following examples:

Returns T, because 31 December 2014 falls within the range specified by *min* and *max*:

```
BETWEEN(`20141231`,`20141229`,`20141231`)
```

Returns F, even though 12:00 PM on 31 December 2014 appears to fall within the range specified by *min* and *max*:

```
BETWEEN(`20141231 120000`,`20141229`,`20141231`)
```

If we look at the serial number equivalent of these two expressions, we can see why the second one evaluates to false.

Returns T, because the serial number *value* is equal to the serial number *max*:

```
BETWEEN(42003.000000, 42001.000000, 42003.000000)
```

Returns F, because the serial number *value* is greater than the serial number *max*:

```
BETWEEN(42003.500000, 42001.000000, 42003.000000)
```

The serial number `42003.500000` is greater than `42003.000000` and therefore is out of range, even though the two dates are identical. `0.500000` is the serial number equivalent of 12:00 PM.

Harmonize Datetime subtypes

To avoid the problems that can be caused by mixing Datetime subtypes, you can use functions to harmonize the subtypes.

For example, this expression, which uses the same initial values as the second example above, returns T rather than F:

```
BETWEEN(CTOD(DATE(`20141231
120000`,"YYYYMMDD"),"YYYYMMDD"),`20141229`,`20141231`)
```

Specifying a literal date, datetime, or time value

When specifying a literal date, datetime, or time value for any of the function inputs, you are restricted to the formats in the table below, and you must enclose the value in backquotes – for example, `` `20141231` ``.

Do not use any separators such as slashes (/) or colons (:) between the individual components of dates or times.

- **Datetime values** – you can use any combination of the date, separator, and time formats listed in the table below. The date must precede the time, and you must use a separator between the two. Valid separators are a single blank space, the letter 't', or the letter 'T'.
- **Time values** – you must specify times using the 24-hour clock. Offsets from Coordinated Universal Time (UTC) must be prefaced by a plus sign (+) or a minus sign (-).

Example formats	Example literal values
YYYYMMDD	`20141231`
YYMMDD	`141231`
YYYYMMDD hhmmss	`20141231 235959`
YYMMDDthhmm	`141231t2359`
YYYYMMDDThh	`20141231T23`
YYYYMMDD hhmmss+/-hhmm (UTC offset)	`20141231 235959-0500`
YYMMDD hhmm+/-hh (UTC offset)	`141231 2359+01`
thhmmss	`t235959`
Thhmm	`T2359`

Example formats	Example literal values
Note Do not use hh alone in the main time format with data that has a UTC offset. For example, avoid: hh+hhmm. Results can be unreliable.	

BINTOSTR( ) function

Returns Unicode character data converted from ZONED or EBCDIC character data. Abbreviation for "Binary to String".

> **Note**
> This function is specific to the Unicode edition of Analytics. It is not a supported function in the non-Unicode edition.

Syntax

```
BINTOSTR(string, string_type)
```

Parameters

Name	Type	Description
string	character	The ZONED or EBCDIC value that you want to convert to Unicode character encoding.
string_type	character	The format to convert from. You must specify one of the following values: ◦ `"A"` - convert from ZONED (ASCII) data ◦ `"E"` - convert from EBCDIC data

Output

Character.

Examples

Basic examples

The expression ZONED(-6448,4) converts the value -6448 to the character format "644Q", however the Unicode edition of Analytics requires that you convert the output of ZONED( ) to Unicode

characters using BINTOSTR( ).

Returns "644Q" in Unicode format:

```
BINTOSTR(ZONED(-6448,4), "A")
```

Remarks

When to use BINTOSTR( )

Use this function to convert return values from the ZONED( ) and EBCDIC( ) functions to a Unicode value.

> **Note**
> If this function is not applied to the return values of ZONED( ) and EBCDIC( ) in Unicode editions of Analytics, then they are displayed incorrectly because the encoding is not interpreted correctly.

BIT( ) function

Returns the binary representation for the specified byte position in the current record as an eight character string.

Syntax

```
BIT(byte_location)
```

Parameters

Name	Type	Description
byte_location	numeric	The byte position to return as a binary value.

Output

Character.

Examples

Basic examples

Returns "00110001" if the eighth byte contains "1":

```
BIT(8)
```

Returns "01000001" if the ninth byte contains "A":

```
BIT(9)
```

Returns "01100001" if the seventeenth byte contains "a":

```
BIT(17)
```

Advanced examples

Using BIT ( ) and SUBSTRING ( ) to extract a value

Assume that byte position 17 contains a set of 8 credit flags.

To extract all customer records that have the third bit set to one (meaning "do not ship"), specify:

```
EXTRACT IF SUBSTRING(BIT(17), 3, 1) = "1"
```

In this example, the SUBSTRING( ) function is used to extract the value of the third bit.

Remarks

How it works

BIT( ) converts the byte at the specified byte position into an eight character string of ones and zeros.

When to use BIT( )

Use BIT( ) to examine the individual bits in a byte.

Related functions

If you want to retrieve the character at the specified byte location, use the BYTE( ) function.

BLANKS( ) function

Returns a string containing a specified number of blank spaces.

Syntax

```
BLANKS(count)
```

Parameters

Name	Type	Description
count	numeric	The number of blank spaces to insert.

Output

Character.

Examples

Basic examples

Returns " ":

```
BLANKS(5)
```

Returns "ABC Corporation":

```
"ABC" + BLANKS(1) + "Corporation"
```

Remarks

When to use BLANKS( )

Use the BLANKS( ) function to harmonize fields, to initialize variables in scripts, or to insert blank spaces when formatting fields or concatenating strings.

BYTE( ) function

Returns the character stored in the specified byte position in the current record.

Syntax

```
BYTE(byte_location)
```

Parameters

Name	Type	Description
byte_location	numeric	The byte position to return as a character value. The value refers to a position in the record (counting from 1), irrespective of any field definitions.

Output

Character.

Examples

Basic examples

Returns "1" from a record that begins with an ID field containing "1":

```
byte(112)
```

Advanced examples

Identify records in print files or PDFs based on consistent formatting

Use BYTE( ) to identify records in a data file where a particular character is present in a particular byte position. This is typically the case in Print Image (Report) files or Adobe Acrobat (PDF) files where data is formatted in a consistent way throughout the document.

For example, to locate and extract records that include a period at byte position 113:

```
EXTRACT RECORD IF BYTE(113) = "." TO "Output.fil"
```

Remarks

When to use BYTE( )

Use BYTE( ) to examine the contents of a position in a record, without having to define a field for the purpose.

Using BYTE( ) on EBCDIC data

If you use this function on EBCDIC data, the value returned will also be EBCDIC. You may not be able to compare this to character values.

Related functions

If you want to retrieve the binary representation for specified byte location, use the BIT( ) function.

CDOW( ) function

Returns the name of the day of the week for a specified date or datetime. Abbreviation for "Character Day of Week".

Syntax

```
CDOW(date/datetime, length)
```

Parameters

Name	Type	Description
date/datetime	datetime	The field, expression, or literal value to return the day name for.
length	numeric	A value between 1 and 9 that specifies the length of the output string. To display abbreviated day names, specify a smaller value.

Output

Character.

Examples

Basic examples

Returns "Wednesday" because December 31, 2014 falls on a Wednesday, and *length* is 9:

```
CDOW(`20141231`, 9)
```

Returns "Wed" because December 31, 2014 falls on a Wednesday, and *length* is 3:

```
CDOW(`20141231 235959`, 3)
```

Returns the full day name for each value in the **Invoice_date** field:

```
CDOW(Invoice_date, 9)
```

Returns the abbreviated day name for each value in the **Receipt_timestamp** field:

```
CDOW(Receipt_timestamp, 3)
```

Advanced examples

Adding a field that identifies the days of the week for dates

Use the CDOW( ) function to create a computed field that identifies the days of the week for all the dates in a date field. Once you have created the computed field, you can add it to the view beside the date column:

```
DEFINE FIELD Name_of_Day COMPUTED CDOW(Trans_Date, 3)
```

Creating a filter to test for transactions that occurred on a weekend

Use the CDOW( ) function to create a filter that isolates transactions that occurred on a weekend:

```
SET FILTER TO CDOW(Trans_Date, 3) = "Sat" OR CDOW(Trans_Date, 3) = "Sun"
```

Remarks

Parameter details

A field specified for *date/datetime* can use any date or datetime format, as long as the field definition correctly defines the format.

If the *length* parameter is shorter than the day name, the day name is truncated to the specified length. If the *length* parameter is longer than the day name, the day name is padded with blank spaces.

Specifying a literal date or datetime value

When specifying a literal date or datetime value for *date/datetime*, you are restricted to the formats in the table below, and you must enclose the value in backquotes – for example, `` `20141231` ``.

Do not use any separators such as slashes (/) or colons (:) between the individual components of dates or times.

- **Datetime values** – you can use any combination of the date, separator, and time formats listed in the table below. The date must precede the time, and you must use a separator between the two. Valid separators are a single blank space, the letter 't', or the letter 'T'.
- **Time values** – you must specify times using the 24-hour clock. Offsets from Coordinated Universal Time (UTC) must be prefaced by a plus sign (+) or a minus sign (-).

Example formats	Example literal values
YYYYMMDD	`20141231`
YYMMDD	`141231`
YYYYMMDD hhmmss	`20141231 235959`
YYMMDDthhmm	`141231t2359`
YYYYMMDDThh	`20141231T23`
YYYYMMDD hhmmss+/-hhmm (UTC offset)	`20141231 235959-0500`
YYMMDD hhmm+/-hh (UTC offset)	`141231 2359+01`

Example formats	Example literal values
Note Do not use hh alone in the main time format with data that has a UTC offset. For example, avoid: hh+hhmm. Results can be unreliable.	

Related functions

If you need to return the day of the week as a number (1 to 7), use DOW( ) instead of CDOW( ).

CHR( ) function

Returns the character associated with the specified ASCII code.

Syntax

```
CHR(number)
```

Parameters

Name	Type	Description
number	numeric	A valid numeric expression between 1 and 255.

Output

Character.

Examples

Basic examples

Returns "A":

```
CHR(65)
```

Returns "1":

```
CHR(49)
```

Advanced examples

Adding the UK pound symbol (£) to each of the values in a currency field

Create a computed field that adds the pound symbol (ASCII code 163) before amounts in the **Invoice_Amount** field. The numeric **Invoice_Amount** field is first converted to a character field, and leading and trailing blank spaces are trimmed.

```
DEFINE FIELD Currency_UK COMPUTED CHR(163)+ALLTRIM(STRING(Invoice_
Amount, 12))
```

Remarks

When to use CHR( )

Use the CHR( ) function to return the character associated with any ASCII code, including those characters that cannot be entered directly from a keyboard or displayed on screen. With CHR( ), you can search fields or records for the existence of these specific characters.

Referencing NUL

Referencing the ASCII NUL (null) character, `CHR(0)`, may produce unpredictable results because it is used by Analytics as a text qualifier, and should be avoided if possible.

Related functions

CHR( ) is the inverse of the ASCII( ) function.

CLEAN( ) function

Replaces the first invalid character in a string, and all subsequent characters, with blanks.

Syntax

```
CLEAN(string <,extra_invalid_characters>)
```

Parameters

Name	Type	Description
string	character	The value from which to remove default and any extra invalid characters.

extra_invalid_characters
optional
	character	Invalid characters you want to remove from *string* in addition to the default invalid characters. You may specify more than one extra invalid character:
		`" ,;\"`
		Tab characters, null characters, and carriage return and line feed characters are default invalid characters that are automatically removed and do not need to be specified.
		To specify the double quotation mark as an extra invalid character, enclose *extra_invalid_characters* in single quotation marks:
		`'"'`

Output

Character.

Examples

Basic examples

Returns "ABC " ("ABC" followed by four blank spaces):

```
CLEAN("ABC%DEF","%")
```

Returns "1234.56 " ("1234.56" followed by six blank spaces):

```
CLEAN("1234.56,111,2", ",")
```

Remarks

When to use CLEAN( )

Use this function to ensure that fields containing invalid data are printed correctly. You can also use this function to isolate parts of a field, such as the last name in a customer field that includes both the first and last name of the customer.

Specifying invalid single and double quotation marks

If you need to specify both single and double quotation marks as invalid characters, you must nest the CLEAN( ) function within itself:

```
CLEAN(CLEAN(string, '"'), "'")
```

Automatic CLEAN( )

In an Analytics script, you can apply the CLEAN( ) function automatically to all character fields by adding `SET CLEAN ON` to the script. You cannot specify extra individual characters using this option.

CMOY( ) function

Returns the name of the month of the year for a specified date or datetime. Abbreviation for "Character Month of Year".

Syntax

```
CMOY(date/datetime, length)
```

Parameters

Name	Type	Description
date/datetime	datetime	The field, expression, or literal value to return the month name for.
length	numeric	A value between 1 and 9 that specifies the length of the output string. To display abbreviated month names, specify a smaller value.

Output

Character.

Examples

Basic examples

Returns "December":

```
CMOY(`20141231`, 9)
```

Returns "Dec":

```
CMOY(`20141231 235959`, 3)
```

Returns the abbreviated month name for each value in the **Receipt_timestamp** field:

```
CMOY(Receipt_timestamp, 3)
```

Returns the full month name for each value in the **Invoice_date** field:

```
CMOY(Invoice_date, 9)
```

Returns the full name of the month 15 days after each value in the **Invoice_date** field:

```
CMOY(Invoice_date + 15, 9)
```

Remarks

Parameter details

A field specified for *date/datetime* can use any date or datetime format, as long as the field definition correctly defines the format.

If the *length* parameter is shorter than the month name, the month name is truncated to the specified length. If the *length* parameter is longer than the month name, the month name is padded with blank spaces.

Specifying a literal date or datetime value

When specifying a literal date or datetime value for *date/datetime*, you are restricted to the formats in the table below, and you must enclose the value in backquotes – for example, `` `20141231` ``.

Do not use any separators such as slashes (/) or colons (:) between the individual components of dates or times.

- **Datetime values** – you can use any combination of the date, separator, and time formats listed in the table below. The date must precede the time, and you must use a separator between the two. Valid separators are a single blank space, the letter 't', or the letter 'T'.
- **Time values** – you must specify times using the 24-hour clock. Offsets from Coordinated Universal Time (UTC) must be prefaced by a plus sign (+) or a minus sign (-).

Example formats	Example literal values
YYYYMMDD	`20141231`

Example formats	Example literal values
YYMMDD	`141231`
YYYYMMDD hhmmss	`20141231 235959`
YYMMDDthhmm	`141231t2359`
YYYYMMDDThh	`20141231T23`
YYYYMMDD hhmmss+/-hhmm (UTC offset)	`20141231 235959-0500`
YYMMDD hhmm+/-hh (UTC offset)	`141231 2359+01`
Note Do not use hh alone in the main time format with data that has a UTC offset. For example, avoid: hh+hhmm. Results can be unreliable.	

Related functions

If you need to return the month of the year as a number (1 to 12), use MONTH( ) instead of CMOY( ).

COS( ) function

Returns the cosine of an angle expressed in radians, with a precision of 15 decimal places.

Syntax

```
COS(radians)
```

Parameters

Name	Type	Description
radians	numeric	The measurement of the angle in radians.

Output

Numeric.

Examples

Basic examples

Returns 0.500000000000000 (the specified number of radians):

```
COS(1.047197551196598)
```

Advanced examples

Using degrees as input

Returns 0.500000000000000 (the cosine of 60 degrees):

```
COS(60 * PI( )/180)
```

Rounding to 3 decimal places

Returns 0.500 (the cosine of 60 degrees, rounded to 3 decimal places):

```
DEC(COS(60 * PI( )/180),3)
```

Remarks

Performing the Mantissa Arc Test

The three trigonometric functions in Analytics – SIN( ), COS( ), and TAN( ) – support performing the Mantissa Arc Test associated with Benford's Law.

Converting degrees to radians

If your input is in degrees you can use the PI( ) function to convert the input to radians: (*degrees* * PI ( )/180) = *radians*. If required, you can round or truncate the return value using the DEC( ) function.

CTOD( ) function

Converts a character or numeric date value to a date. Can also extract the date from a character or numeric datetime value and return it as a date. Abbreviation for "Character to Date".

Syntax

```
CTOD(string/number <,format>)
```

Parameters

Name	Type	Description
string/number	character numeric	The field, expression, or literal value to convert to a date, or from which to extract the date.
format optional	character	The date format of *string/number*. The *format* is required for values that use any date format other than YYYYMMDD or YYMMDD, for example `"DD/MM/YYYY"`. **Note** If you use the CTOD function with a datetime value that requires the *format* parameter, specify only the date portion of the format, and not the time portion. For example: `CTOD("31/12/2014 23:59:59", "DD/MM/YYYY")` Specifying the time portion prevents the results from appearing.

Output

Datetime. The date value is output using the current Analytics date display format.

Examples

Basic examples

Character literal input

Returns `20141231` displayed as 31 Dec 2014 assuming a current Analytics date display format of DD MMM YYYY:

```
CTOD("20141231")
```

```
CTOD("31/12/2014", "DD/MM/YYYY")
```

```
CTOD("20141231 235959")
```

Numeric literal input

Returns `20141231` displayed as 31 Dec 2014 assuming a current Analytics date display format of DD MMM YYYY:

```
CTOD(20141231)
```

```
CTOD(31122014, "DDMMYYYY")
```

```
CTOD(20141231.235959)
```

Character field input

Returns each value in the specified character field as a date, using the current Analytics date display format:

```
CTOD(Invoice_date, "DD/MM/YYYY")
```

```
CTOD(Receipt_timestamp)
```

Numeric field input

Returns each value in the specified numeric field as a date, using the current Analytics date display format:

```
CTOD(Due_date, "DDMMYYYY")
```

```
CTOD(Payment_timestamp)
```

Advanced examples

Compare a character or numeric field to a date

Use the CTOD( ) function to compare a date against character or numeric fields that contain values representing dates.

The filter below compares two values:

- the numeric **Due_date** field that stores dates as numbers in the format DDMMYYYY
- the literal date value July 1, 2014

```
SET FILTER TO CTOD(Due_date, "DDMMYYYY") < `20140701`
```

Remarks

Required date formats

Character and numeric fields containing date or datetime values must match the formats in the table below. Datetime values can use any combination of the date, separator, and time formats valid for their data type. The date must precede the time, and there must be a separator between the two.

Dates, or the date portion of datetime values, can use any date format supported by Analytics, and valid for the data type, as long as formats other than YYYYMMDD or YYMMDD are correctly defined by *format*.

Date formats	Separator formats	Time formats
Character fields		

Date formats	Separator formats	Time formats
YYYYMMDD	single blank space	hhmmss hh:mm:ss
YYMMDD	the letter 't'	hhmm hh:mm
any Analytics-supported date format, valid for the data type, if defined by *format*	the letter 'T'	hh
		+/-hhmm +/-hh:mm (UTC offset)
		+/-hh (UTC offset)
		Note Do not use hh alone in the main time format with data that has a UTC offset. For example, avoid: hh+hhmm. Results can be unreliable.
Numeric fields		
YYYYMMDD	decimal point	hhmmss
YYMMDD		hhmm
any Analytics-supported date format, valid for the data type, if defined by *format*		hh

Other datetime conversion functions

Character or Numeric to Datetime conversion

Function	Description
CTODT( )	Converts a character or numeric datetime value to a datetime. Abbreviation for "Character to Datetime".
CTOT( )	Converts a character or numeric time value to a time. Can also extract the time from a character or numeric datetime value and return it as a time. Abbreviation for "Character to

Function	Description
	Time".

Datetime to Character conversion

Function	Description
DATE( )	Extracts the date from a specified date or datetime and returns it as a character string. Can also return the current operating system date.
DATETIME( )	Converts a datetime to a character string. Can also return the current operating system datetime.
TIME( )	Extracts the time from a specified time or datetime and returns it as a character string. Can also return the current operating system time.

Serial to Datetime conversion

Function	Description
STOD( )	Converts a serial date - that is, a date expressed as an integer - to a date value. Abbreviation for "Serial to Date".
STODT( )	Converts a serial datetime - that is, a datetime expressed as an integer, and a fractional portion of 24 hours - to a datetime value. Abbreviation for "Serial to Datetime".
STOT( )	Converts a serial time - that is, a time expressed as a fractional portion of 24 hours, with 24 hours equaling 1 - to a time value. Abbreviation for "Serial to Time".

CTODT( ) function

Converts a character or numeric datetime value to a datetime. Abbreviation for "Character to Datetime".

Syntax

```
CTODT(string/number <,format>)
```

Parameters

Name	Type	Description
string/number	character numeric	The field, expression, or literal value to convert to a datetime.
format optional	character	The date format of *string/number*. The *format* is required for values that use any date format other than YYYYMMDD or YYMMDD for the date portion of the value, for example `"DD/MM/YYYY"`.

Output

Datetime. The datetime value is output using the current Analytics date and time display formats.

Examples

Basic examples

Character literal input

Returns `20141231t235959` displayed as 31 Dec 2014 23:59:59 assuming a current Analytics date and time display formats of DD MMM YYYY and hh:mm:ss:

```
CTODT("20141231 235959")
```

```
CTODT("31/12/2014 23:59:59", "DD/MM/YYYY hh:mm:ss")
```

Numeric literal input

Returns `20141231t235959` displayed as 31 Dec 2014 23:59:59 assuming a current Analytics date and time display formats of DD MMM YYYY and hh:mm:ss:

```
CTODT(20141231.235959)
```

```
CTODT(31122014.235959, "DDMMYYYY.hhmmss")
```

Character field input

Returns each value in the **Receipt_timestamp** character field as a datetime, using the current Analytics date display format:

```
CTODT(Receipt_timestamp, "DD/MM/YYYY hh:mm:ss")
```

Numeric field input

Returns each value in the **Payment_timestamp** numeric field as a datetime, using the current Analytics date display format:

```
CTODT(Payment_timestamp, "DD/MM/YYYY hh:mm:ss")
```

Advanced examples

Compare a character or numeric field to a datetime

Use the CTODT( ) function to compare a datetime against character or numeric fields that contain values representing datetimes.

The filter below compares two values:

- the character **Receipt_timestamp** field that stores datetimes as character data in the format DD/MM/YYYY hh:mm:ss
- the literal datetime value July 1, 2014 13:30:00

```
SET FILTER TO CTODT(Receipt_timestamp, "DD/MM/YYYY hh:mm:ss") <
`20140701t133000`
```

Remarks

Required datetime formats

Character and numeric fields containing datetime values must match the formats in the table below. The datetime values can use any combination of the date, separator, and time formats valid for their data type. The date must precede the time, and there must be a separator between the two.

The date portion of values can use any date format supported by Analytics, and valid for the data type, as long as formats other than YYYYMMDD or YYMMDD are correctly defined by *format*. If you use *format*, you must also specify the time format, which must be one of the time formats that appear in the table below.

Analytics automatically recognizes the separator between the date and time portions of datetime values, so there is no need to specify the separator in *format*. You can specify the separator if you want to.

Date formats	Separator formats	Time formats
Character fields		
YYYYMMDD	single blank space	hhmmss hh:mm:ss
YYMMDD	the letter 't'	hhmm hh:mm
any Analytics-supported date format, valid for the data type, if defined by *format*	the letter 'T'	hh
		+/-hhmm +/-hh:mm (UTC offset)
		+/-hh (UTC offset)

Date formats	Separator formats	Time formats
		Note Do not use hh alone in the main time format with data that has a UTC offset. For example, avoid: hh+hhmm. Results can be unreliable.
Numeric fields		
YYYYMMDD	decimal point	hhmmss
YYMMDD		hhmm
any Analytics-supported date format, valid for the data type, if defined by *format*		hh

Other datetime conversion functions

Character or Numeric to Datetime conversion

Function	Description
CTOD( )	Converts a character or numeric date value to a date. Can also extract the date from a character or numeric datetime value and return it as a date. Abbreviation for "Character to Date".
CTOT( )	Converts a character or numeric time value to a time. Can also extract the time from a character or numeric datetime value and return it as a time. Abbreviation for "Character to Time".

Datetime to Character conversion

Function	Description
DATE( )	Extracts the date from a specified date or datetime and returns it as a character string. Can also return the current operating system date.
DATETIME( )	Converts a datetime to a character string. Can also return the current operating system datetime.
TIME( )	Extracts the time from a specified time or datetime and returns it as a character string. Can also return the current operating system time.

Serial to Datetime conversion

Function	Description
STOD( )	Converts a serial date - that is, a date expressed as an integer - to a date value. Abbreviation for "Serial to Date".
STODT( )	Converts a serial datetime - that is, a datetime expressed as an integer, and a fractional portion of 24 hours - to a datetime value. Abbreviation for "Serial to Datetime".
STOT( )	Converts a serial time - that is, a time expressed as a fractional portion of 24 hours, with 24 hours equaling 1 - to a time value. Abbreviation for "Serial to Time".

CTOT( ) function

Converts a character or numeric time value to a time. Can also extract the time from a character or numeric datetime value and return it as a time. Abbreviation for "Character to Time".

Syntax

```
CTOT(string/number)
```

Parameters

Name	Type	Description
string/number	character numeric	The field, expression, or literal value to convert to a time, or from which to extract the time.

Output

Datetime. The time value is output using the current Analytics time display format.

Examples

Basic examples

Character literal input

Returns `t235959` displayed as 23:59:59 assuming a current Analytics time display format of hh:mm:ss:

```
CTOT("t235959")
```

```
CTOT("23:59:59")
```

```
CTOT("20141231 235959")
```

Numeric literal input

Returns `t235959` displayed as 23:59:59 assuming a current Analytics time display format of hh:mm:ss:

```
CTOT(.235959)
```

```
CTOT(0.235959)
```

```
CTOT(20141231.235959)
```

Character field input

Returns each value in the **Login_time** character field as a time, using the current Analytics time display format:

```
CTOT(Login_time)
```

Numeric field input

Returns each value in the **Payment_datetime** numeric field as a time, without any date portion, using the current Analytics time display format:

```
CTOT(Payment_datetime)
```

Advanced examples

Compare a character or numeric field to a time

Use the CTOT( ) function to compare a time against character or numeric fields that contain values representing times.

The filter below compares two values:

- the numeric **Login_time** field that stores times as numeric data
- the literal time value 09:30:00

```
SET FILTER TO CTOT(Login_time) > `t093000`
```

Remarks

Required datetime formats

Character and numeric fields containing time or datetime values must match the formats in the table below.

Time values can use any combination of separator and time format. There must be a separator before the time value, or colons between the time components, for the function to operate correctly.

Datetime values can use any combination of the date, separator, and time formats valid for their data type. The date must precede the time, and there must be a separator between the two.

Use the CTOD( ) function if you want to convert a character or numeric date value to a date, or extract the date from a character or numeric datetime value and return it as a date.

Use the CTODT( ) function if you want to convert a character or numeric datetime value to a datetime.

Date formats	Separator formats	Time formats
Character fields		
YYYYMMDD	single blank space	hhmmss hh:mm:ss
YYMMDD	the letter 't'	hhmm hh:mm
	the letter 'T'	hh
		+/-hhmm +/-hh:mm (UTC offset)
		+/-hh (UTC offset)

Date formats	Separator formats	Time formats
		Note Do not use hh alone in the main time format with data that has a UTC offset. For example, avoid: hh+hhmm. Results can be unreliable.)
Numeric fields		
YYYYMMDD	decimal point	hhmmss
YYMMDD		hhmm
		hh

Other datetime conversion functions

Character or Numeric to Datetime conversion

Function	Description
CTOD( )	Converts a character or numeric date value to a date. Can also extract the date from a character or numeric datetime value and return it as a date. Abbreviation for "Character to Date".
CTODT( )	Converts a character or numeric datetime value to a datetime. Abbreviation for "Character to Datetime".

Datetime to Character conversion

Function	Description
DATE( )	Extracts the date from a specified date or datetime and returns it as a character string. Can also return the current operating system date.
DATETIME( )	Converts a datetime to a character string. Can also return the current operating system datetime.
TIME( )	Extracts the time from a specified time or datetime and returns it as a character string. Can also return the current operating system time.

Serial to Datetime conversion

Function	Description
STOD( )	Converts a serial date - that is, a date expressed as an integer - to a date value. Abbreviation for "Serial to Date".
STODT( )	Converts a serial datetime - that is, a datetime expressed as an integer, and a fractional portion of 24 hours - to a datetime value. Abbreviation for "Serial to Datetime".
STOT( )	Converts a serial time - that is, a time expressed as a fractional portion of 24 hours, with 24 hours equaling 1 - to a time value. Abbreviation for "Serial to Time".

CUMIPMT( ) function

Returns the cumulative interest paid on a loan during a range of periods.

Syntax

```
CUMIPMT(rate, periods, amount, start_period, end_period <,type>)
```

Parameters

Name	Type	Description
rate	numeric	The interest rate per period.
periods	numeric	The total number of payment periods.
amount	numeric	The principal amount of the loan.
start_period	numeric	The first period in the calculation. *start_period* cannot be 0.
end_period	numeric	The last period in the calculation. *end_period* cannot be greater than the total number of payment periods.
type optional	numeric	The timing of payments: ◦ `0` - payment at the end of a period ◦ `1` - payment at the beginning of a period If omitted, the default value of 0 is used.

> **Note**
>
> You must use consistent time periods when specifying *rate* and *periods* to ensure that you are specifying interest rate **per period**.
>
> For example:
>
> - for monthly payments on a two-year loan or investment with interest of 5% per annum, specify 0.05/12 for *rate* and 2 * 12 for *periods*
> - for annual payments on the same loan or investment, specify 0.05 for *rate* and 2 for *periods*

Output

Numeric.

Examples

Basic examples

Returns 17437.23, the total amount of interest paid in the second year of a twenty-five year, $275,000 loan at 6.5% per annum, with payments due at the end of the month:

```
CUMIPMT(0.065/12, 12*25, 275000, 13, 24, 0)
```

Returns 17741.31, the total amount of interest paid on the same loan in the first year of the loan:

```
CUMIPMT(0.065/12, 12*25, 275000, 1, 12, 0)
```

Remarks

Related functions

The CUMPRINC( ) function is the complement of the CUMIPMT( ) function.

The IPMT( ) function calculates interest paid for a single period.

CUMPRINC( ) function

Returns the cumulative principal paid on a loan during a range of periods.

Syntax

```
CUMPRINC(rate, periods, amount, start_period, end_period <,type>)
```

Parameters

Name	Type	Description
rate	numeric	The interest rate per period.
periods	numeric	The total number of payment periods.
amount	numeric	The principal amount of the loan.
start_period	numeric	The first period in the calculation. *start_period* cannot be 0.
end_period	numeric	The last period in the calculation. *end_period* cannot be greater than the total number of payment periods.
type optional	numeric	The timing of payments: ◦ `0` - payment at the end of a period ◦ `1` - payment at the beginning of a period If omitted, the default value of 0 is used.

> **Note**
>
> You must use consistent time periods when specifying *rate* and *periods* to ensure that you are specifying interest rate **per period**.
>
> For example:
>
> - for monthly payments on a two-year loan or investment with interest of 5% per annum, specify 0.05/12 for *rate* and 2 * 12 for *periods*
> - for annual payments on the same loan or investment, specify 0.05 for *rate* and 2 for *periods*

Output

Numeric.

Examples

Basic examples

Returns 4844.61, the total amount of principal paid in the second year of a twenty-five year, $275,000 loan at 6.5% per annum, with payments due at the end of the month:

```
CUMPRINC(0.065/12, 12*25, 275000, 13, 24, 0)
```

Returns 367.24, the amount of principal paid on the same loan in the first month of the loan:

```
CUMPRINC(0.065/12, 12*25, 275000, 1, 1, 0)
```

Remarks

Related functions

The CUMIPMT( ) function is the complement of the CUMPRINC( ) function.

The PPMT( ) function calculates principal paid for a single period.

DATE( ) function

Extracts the date from a specified date or datetime and returns it as a character string. Can also return the current operating system date.

Syntax

```
DATE(<date/datetime> <,format>)
```

Parameters

Name	Type	Description
date/datetime optional	datetime	The field, expression, or literal value to extract the date from. If omitted, the current operating system date is returned.
format optional	character	The format to apply to the output string, for example `"DD/MM/YYYY"`. If omitted, the current Analytics date display format is used. You cannot specify a *format* if you have omitted *date/datetime*.

Output

Character.

Examples

Basic examples

Returns "20141231" in the current Analytics date display format:

```
DATE(`20141231 235959`)
```

Returns "31-Dec-2014":

```
DATE(`20141231 235959`, "DD-MMM-YYYY")
```

Returns the current operating system date as a character string, using the current Analytics date display format:

```
DATE()
```

Returns each value in the **Receipt_timestamp** field as a character string using the current Analytics date display format:

```
DATE(Receipt_timestamp)
```

Returns each value in the **Receipt_timestamp** field as a character string using the specified date display format:

```
DATE(Receipt_timestamp, "DD/MM/YYYY")
```

Remarks

Output string length

The length of the output string is always 12 characters. If the specified output *format*, or the Analytics date display format, is less than 12 characters, the output string is padded with trailing blank spaces.

Parameter details

A field specified for *date/datetime* can use any date or datetime format, as long as the field definition correctly defines the format.

If you use *format* to control how the output string is displayed, you can use any supported Analytics date display format. For example:

- DD/MM/YYYY
- MM-DD-YY
- DD MMM YYYY

format must be specified using single or double quotation marks – for example, `"DD MMM YYYY"`.

Specifying a literal date or datetime value

When specifying a literal date or datetime value for *date/datetime*, you are restricted to the formats in the table below, and you must enclose the value in backquotes – for example, `` `20141231` ``.

Do not use any separators such as slashes (/) or colons (:) between the individual components of dates or times.

- **Datetime values** – you can use any combination of the date, separator, and time formats listed in the table below. The date must precede the time, and you must use a separator between the two. Valid separators are a single blank space, the letter 't', or the letter 'T'.
- **Time values** – you must specify times using the 24-hour clock. Offsets from Coordinated Universal Time (UTC) must be prefaced by a plus sign (+) or a minus sign (-).

Example formats	Example literal values
YYYYMMDD	`20141231`
YYMMDD	`141231`
YYYYMMDD hhmmss	`20141231 235959`
YYMMDDthhmm	`141231t2359`
YYYYMMDDThh	`20141231T23`
YYYYMMDD hhmmss+/-hhmm (UTC offset)	`20141231 235959-0500`
YYMMDD hhmm+/-hh (UTC offset)	`141231 2359+01`
Note Do not use hh alone in the main time format with data that has a UTC offset. For example, avoid: hh+hhmm. Results can be unreliable.	

Related functions

If you need to return the current operating system date as a datetime value, use TODAY( ) instead of DATE( ).

Other datetime conversion functions

Datetime to Character conversion

Function	Description
DATETIME( )	Converts a datetime to a character string. Can also return the current operating system datetime.
TIME( )	Extracts the time from a specified time or datetime and returns it as a character string. Can also return the current operating system time.

Character or Numeric to Datetime conversion

Function	Description
CTOD( )	Converts a character or numeric date value to a date. Can also extract the date from a character or numeric datetime value and return it as a date. Abbreviation for "Character to Date".
CTODT( )	Converts a character or numeric datetime value to a datetime. Abbreviation for "Character to Datetime".
CTOT( )	Converts a character or numeric time value to a time. Can also extract the time from a character or numeric datetime value and return it as a time. Abbreviation for "Character to Time".

Serial to Datetime conversion

Function	Description
STOD( )	Converts a serial date – that is, a date expressed as an integer – to a date value. Abbreviation for "Serial to Date".
STODT( )	Converts a serial datetime – that is, a datetime expressed as an integer, and a fractional portion of 24 hours – to a datetime value. Abbreviation for "Serial to Datetime".
STOT( )	Converts a serial time – that is, a time expressed as a fractional portion of 24 hours, with 24 hours equaling 1 – to a time value. Abbreviation for "Serial to Time".

DATETIME( ) function

Converts a datetime to a character string. Can also return the current operating system datetime.

Syntax

```
DATETIME(<datetime> <,format>)
```

Parameters

Name	Type	Description
datetime optional	datetime	The field, expression, or literal value to convert. If omitted, the current operating system date is returned.
format optional	character	The format to apply to the output string, for example `"DD/MM/YYYY"`. If omitted, the current Analytics date display format is used. You cannot specify a *format* if you have omitted *date/datetime*.

Output

Character.

Examples

Basic examples

Literal datetime input

Returns "20141231 235959" in the current Analytics date and time display formats:

```
DATETIME(`20141231 235959`)
```

Returns "31-Dec-2014 11:59 P":

```
DATETIME(`20141231 235959`, "DD-MMM-YYYY hh:mm A")
```

Returns the current operating system date and time as a character string, using the current Analytics date and time display formats:

```
DATETIME()
```

Field input

Returns each value in the **Receipt_timestamp** field as a character string using the current Analytics date and time display formats:

```
DATETIME(Receipt_timestamp)
```

Returns each value in the **Receipt_timestamp** field as a character string using the specified date and time display formats:

```
DATETIME(Receipt_timestamp, "DD/MM/YYYY hh:mm:ss")
```

Remarks

Output string length

The length of the output string is always 27 characters. If the specified output *format*, or the Analytics date and time display formats, are less than 27 characters, the output string is padded with trailing blank spaces.

Parameter details

A field specified for *datetime* can use any datetime format, as long as the field definition correctly defines the format.

If you use *format* to control how the output string is displayed, you are restricted to the formats in the table below.

- You can use any combination of date, time, and AM/PM formats.
- The date must precede the time. Placing a separator between the two is not required as Analytics automatically uses a single space as a separator in the output string.
- The AM/PM format is optional, and is placed last.

- *format* must be specified using single or double quotation marks.

  For example: `"DD-MMM-YYYY hh:mm:ss AM"`

Date formats	Time formats	AM/PM formats	Examples
all supported Analytics date display formats	hh:mm:ss	none 24-hour clock	"DD/MM/YYYY hh:mm:ss"
	hhmmss	AM, or PM 12-hour clock	"MMDDYY hhmmss PM"
	hh:mm	A, or P 12-hour clock	"DD-MMM-YYYY hh:mm A"
	hhmm		
	hh		

Specifying a literal datetime value

When specifying a literal datetime value for *datetime*, you are restricted to the formats in the table below, and you must enclose the value in backquotes – for example, `` `20141231 235959` ``.

Do not use any separators such as slashes (/) or colons (:) between the individual components of dates or times.

- **Datetime values** – you can use any combination of the date, separator, and time formats listed in the table below. The date must precede the time, and you must use a separator between the two. Valid separators are a single blank space, the letter 't', or the letter 'T'.
- **Time values** – you must specify times using the 24-hour clock. Offsets from Coordinated Universal Time (UTC) must be prefaced by a plus sign (+) or a minus sign (-).

Example formats	Example literal values
YYYYMMDD hhmmss	`` `20141231 235959` ``
YYMMDDthhmm	`` `141231t2359` ``
YYYYMMDDThh	`` `20141231T23` ``
YYYYMMDD hhmmss+/-hhmm (UTC offset)	`` `20141231 235959-0500` ``
YYMMDD hhmm+/-hh (UTC offset)	`` `141231 2359+01` ``

Example formats	Example literal values
Note Do not use hh alone in the main time format with data that has a UTC offset. For example, avoid: hh+hhmm. Results can be unreliable.	

Other datetime conversion functions

Datetime to Character conversion

Function	Description
DATE( )	Extracts the date from a specified date or datetime and returns it as a character string. Can also return the current operating system date.
TIME( )	Extracts the time from a specified time or datetime and returns it as a character string. Can also return the current operating system time.

Character or Numeric to Datetime conversion

Function	Description
CTOD( )	Converts a character or numeric date value to a date. Can also extract the date from a character or numeric datetime value and return it as a date. Abbreviation for "Character to Date".
CTODT( )	Converts a character or numeric datetime value to a datetime. Abbreviation for "Character to Datetime".
CTOT( )	Converts a character or numeric time value to a time. Can also extract the time from a character or numeric datetime value and return it as a time. Abbreviation for "Character to Time".

Serial to Datetime conversion

Function	Description
STOD( )	Converts a serial date - that is, a date expressed as an integer - to a date value. Abbreviation for "Serial to Date".
STODT( )	Converts a serial datetime - that is, a datetime expressed as an integer, and a fractional

Function	Description
	portion of 24 hours - to a datetime value. Abbreviation for "Serial to Datetime".
STOT( )	Converts a serial time - that is, a time expressed as a fractional portion of 24 hours, with 24 hours equaling 1 - to a time value. Abbreviation for "Serial to Time".

DAY( ) function

Extracts the day of the month from a specified date or datetime and returns it as a numeric value (1 to 31).

Syntax

```
DAY(date/datetime)
```

Parameters

Name	Type	Description
date/datetime	datetime	The field, expression, or literal value to extract the day from.

Output

Numeric.

Examples

Basic examples

Returns 31:

```
DAY(`20141231`)
```

```
DAY(`20141231  235959`)
```

Returns the day of the month for each value in the **Invoice_date** field:

```
DAY(Invoice_date)
```

Remarks

Parameter details

A field specified for *date/datetime* can use any date or datetime format, as long as the field definition correctly defines the format.

Specifying a literal date or datetime value

When specifying a literal date or datetime value for *date/datetime*, you are restricted to the formats in the table below, and you must enclose the value in backquotes – for example, `` `20141231` ``.

Do not use any separators such as slashes (/) or colons (:) between the individual components of dates or times.

- **Datetime values** – you can use any combination of the date, separator, and time formats listed in the table below. The date must precede the time, and you must use a separator between the two. Valid separators are a single blank space, the letter 't', or the letter 'T'.
- **Time values** – you must specify times using the 24-hour clock. Offsets from Coordinated Universal Time (UTC) must be prefaced by a plus sign (+) or a minus sign (-).

Example formats	Example literal values
YYYYMMDD	`20141231`
YYMMDD	`141231`
YYYYMMDD hhmmss	`20141231 235959`
YYMMDDthhmm	`141231t2359`
YYYYMMDDThh	`20141231T23`
YYYYMMDD hhmmss+/-hhmm (UTC offset)	`20141231 235959-0500`
YYMMDD hhmm+/-hh (UTC offset)	`141231 2359+01`
Note Do not use hh alone in the main time format with data that has a UTC offset. For example, avoid: hh+hhmm. Results can be unreliable.	

Related functions

If you need to return:

- the day of the week as a number (1 to 7), use DOW( ) instead of DAY( )
- the name of the day of the week, use CDOW( ) instead of DAY( )

DBYTE( ) function

Returns the Unicode character located at the specified byte position in a record.

> **Note**
> This function is specific to the Unicode edition of Analytics. It is not a supported function in the non-Unicode edition.

Syntax

```
DBYTE(byte_location)
```

Parameters

Name	Type	Description
byte_location	numeric	The byte position to return as a character value. To return a meaningful value, you must specify the starting point of the double-byte character, which means that you should only specify odd numbers in the *byte_location* parameter.

Output

Character.

Examples

Basic examples

The examples illustrate the behavior of the function when applied to the following Unicode value, which contains 11 characters (22 bytes) 美丽 **10072DOE**:

Returns "丽":

```
DBYTE(3)
```

Returns "D":

```
DBYTE(17)
```

Returns "E":

```
DBYTE(21)
```

Remarks

When to use DBYTE( )

Use DBYTE( ) to examine the contents of a position in a record, without having to define a field for this purpose.

DEC( ) function

Returns a value, or the result of a numeric expression, with the specified number of decimal places.

Syntax

```
DEC(number, decimals)
```

Parameters

Name	Type	Description
number	numeric	The value or result to adjust the number of decimal places for. ◦ **integers** - decimal places are added to the end of *number* as trailing zeros. ◦ **fractional numbers** - If the number of decimal places is reduced, *number* is rounded, not truncated. If the number of decimal places is increased, trailing zeros are added to the end of *number*.
decimals	numeric	The number of decimal places to use in the return value. **Note** You cannot use DEC( ) to increase the decimal precision of results. For information about how to increase decimal precision, see "Controlling rounding and decimal precision in numeric expressions" on page 803.

Output

Numeric.

Examples

Basic examples

Returns 7.00:

```
DEC(7, 2)
```

Returns 7.565:

```
DEC(7.5647, 3)
```

Returns 7.56470:

```
DEC(7.5647, 5)
```

Advanced examples

Calculating daily interest

Calculates the daily interest to six decimal places for a field called **Annual_rate**:

```
DEC(Annual_rate, 6) / 365
```

Remarks

When to use DEC( )

Use this function to adjust the number of decimal places in a field, or when you want to round a value or a result to a specified number of decimal places.

DEC( ) cannot reverse fixed-point rounding

You cannot use the DEC( ) function to reverse the standard rounding that fixed-point arithmetic performs in numeric expressions.

Example

Consider the following series of expressions in Analytics:

```
1.1 * 1.1 = 1.2
1.1 * 1.10 = 1.21
DEC(1.1 * 1.1, 2) = 1.20
```

Fixed-point rounding means that the result of `1.1 * 1.1` is 1.2, not 1.21, which is the unrounded result. Using DEC( ) to specify a two-decimal-place result does not create two-decimal-place precision. Instead, it adds a trailing zero to create the specified number of decimal places, without increasing precision.

For information about how to increase decimal precision, see "Controlling rounding and decimal precision in numeric expressions" on page 803.

Related functions

If you want to round a value to the nearest whole number, use the "ROUND( ) function" on page 2363.

DHEX( ) function

Converts a Unicode string to a hexadecimal string.

> **Note**
> This function is specific to the Unicode edition of Analytics. It is not a supported function in the non-Unicode edition.

Syntax

```
DHEX(field)
```

Parameters

Name	Type	Description
field	character	The Unicode string to convert to a hexadecimal string.

Output

Character.

Examples

Basic examples

Returns "004100420043003100320033":

```
DHEX("ABC123")
```

Remarks

How it works

DHEX( ) displays each double-byte character using big-endian format, with the most significant double-byte stored first.

Each character is represented by a four-character code. The output string is four times as long as the *field* value, and includes the digits between 0 and 9 and letters between A and F that make up the hexadecimal values.

Related functions

DHEX( ) is the inverse of the HTOU( ) function, which converts a hexadecimal string to a Unicode string.

DICECOEFFICIENT( ) function

Returns the Dice's coefficient of two specified strings, which is a measurement of how similar the two strings are.

Syntax

```
DICECOEFFICIENT(string1, string2 <,ngram>)
```

Parameters

Name	Type	Description
string1	character	The first string in the comparison.
string2	character	The second string in the comparison.
ngram optional	numeric	The *n*-gram length to use. Specify a whole number, 1 or greater. Increasing the *ngram* length makes the criterion for similarity between two strings stricter. If you do not specify a length, the default length of 2 is used. *N*-grams are overlapping substrings (character blocks) into which the comparison strings are divided as part of the Dice's coefficient calculation. For detailed information, see "Remarks" on page 2119.

Output

Numeric. The value is the Dice's coefficient of the two strings, which represents the percentage of the total number of *n*-grams in the two strings that are identical. The range is 0.0000 to 1.0000, inclusive.

Examples

Basic examples

How the *n*-gram length affects the result

The three examples below compare the same two strings. The degree of similarity returned varies depending on the specified *n*-gram length.

Returns 0.9167 (using the default *n*-gram length (2), the *n*-grams in the two strings are 92% identical):

```
DICECOEFFICIENT("125 SW 39TH ST, Suite 100","Suite 100, 125 SW 39TH ST")
```

Returns 1.0000 (using an *n*-gram length of 1, the *n*-grams in the two strings are 100% identical):

```
DICECOEFFICIENT("125 SW 39TH ST, Suite 100","Suite 100, 125 SW 39TH ST", 1)
```

Returns 0.8261 (using an *n*-gram length of 3, the *n*-grams in the two strings are 83% identical):

```
DICECOEFFICIENT("125 SW 39TH ST, Suite 100","Suite 100, 125 SW 39TH ST", 3)
```

Field input

Returns the Dice's coefficient of each value in the **Address** field when compared to the string "125 SW 39TH ST, Suite 100" (based on the default *n*-gram length of 2):

```
DICECOEFFICIENT(Address,"125 SW 39TH ST, Suite 100")
```

Advanced examples

Working with transposed elements

By reducing the *n*-gram length, and removing non-essential characters, you can optimize DICECOEFFICIENT( ) when searching for transposed elements.

Returns 0.7368 (using the default *n*-gram length (2), the *n*-grams in the two strings are 74% identical):

```
DICECOEFFICIENT("John Smith","Smith, John")
```

Returns 1.0000 (by excluding the comma between last name and first name, and by using an *n*-gram length of 1, the *n*-grams in the two strings are 100% identical):

```
DICECOEFFICIENT("John Smith", EXCLUDE("Smith, John", ","), 1)
```

Ranking values against "125 SW 39TH ST, Suite 100"

Create the computed field **Dice_Co** to display the Dice's coefficient between "125 SW 39TH ST, Suite 100" and each value in the **Address** field:

```
DEFINE FIELD Dice_Co COMPUTED DICECOEFFICIENT(Address,"125 SW 39TH ST,
Suite 100")
```

Add the computed field **Dice_Co** to the view, and then quick sort it in descending order, to rank all values in the **Address** field based on their similarity to "125 SW 39TH ST, Suite 100".

Isolating fuzzy duplicates for "125 SW 39TH ST, Suite 100"

Create a filter that isolates all values in the **Address** field that are within a specified degree of similarity to "125 SW 39TH ST, Suite 100":

```
SET FILTER TO DICECOEFFICIENT(Address,"125 SW 39TH ST, Suite 100") > 0.5
```

Changing the number in the expression allows you to adjust the degree of similarity in the filtered values.

Remarks

When to use DICECOEFFICIENT( )

Use the DICECOEFFICIENT( ) function to find nearly identical values (fuzzy duplicates). You can also use DICECOEFFICIENT( ) to find values with identical or near-identical content, but transposed elements. For example:

- telephone numbers, or social security numbers, with transposed digits
- versions of the same address, formatted differently

How it works

DICECOEFFICIENT( ) returns the Dice's coefficient of the two evaluated strings, which is a measurement of the degree of similarity between the strings, on a scale from 0.0000 to 1.0000. The greater the returned value the more similar the two strings:

- **1.0000** – means that each string is composed of an identical set of characters, although the characters may be in a different order, and may use different case.
- **0.7500** – means the *n*-grams in the two strings are 75% identical.
- **0.0000** – means the two strings have no shared *n*-grams (explained below), or the specified length of the *n*-gram used in the calculation is longer than the shorter of the two strings being compared.

Usage tips

- **Filtering or sorting** – Filtering or sorting the values in a field based on their Dice's coefficient identifies those values that are most similar to the comparison string.
- **Case-sensitivity** – The function is not case-sensitive, so "SMITH" is equivalent to "smith."
- **Leading and trailing blanks** – The function automatically trims leading and trailing blanks in fields, so there is no need to use the TRIM( ) or ALLTRIM( ) functions when specifying a field as a parameter.

How Dice's coefficient is calculated

Dice's coefficient represents the percentage of the total number of *n*-grams in two strings that are identical.

Dice's coefficient is calculated by first dividing the strings being compared into *n*-grams. *N*-grams (also referred to as *q*-grams) are overlapping substrings, or overlapping character blocks, with a length of *n*. You can specify the length of *n* using the *ngram* parameter, or accept the default length of 2.

Two names divided into *n*-grams

Here are the names "John Smith" and "Smith, John D." divided into *n*-grams with a length of 2, and *n*-grams with a length of 3. Underscores indicate spaces. Internal spaces and punctuation are counted as characters.

n-gram length	"John Smith" *n*-grams	"Smith, John D." *n*-grams
2	Jo | oh | hn | n_ | _S | Sm | mi | it | th	Sm | mi | it | th | h, | ,_ | _J | Jo | oh | hn | n_ | _D | D.
3	Joh | ohn | hn_ | n_S | _Sm | Smi | mit | ith	Smi | mit | ith | th, | h,_ | ,_J | _Jo | Joh | ohn | hn_ | n_D | _D.

The Dice's coefficient formula

Once the *n*-grams have been established for two strings being compared, the calculation is completed using the following formula:

- *2 x the number of shared n-grams / the total number of n-grams in both strings*

Shared *n*-grams are *n*-grams that appear in both strings. For example, "ABC" and "BCD" share the *n*-gram "BC", assuming an *n*-gram length of 2 (AB \| **BC** and **BC** \| CD).

Examples of calculating the Dice's coefficient

The table below illustrates calculating the Dice's coefficient for the two strings, "John Smith" and "Smith, John D.", using different *n*-gram lengths.

Note that as the *n*-gram length increases for the same pair of strings, the Dice's coefficient value decreases, indicating less similarity. Although the strings remain the same, the criterion for similarity becomes stricter because dividing the strings into longer *n*-grams means that longer sequences of characters must match for an *n*-gram to qualify as shared.

Another way of thinking about this point is that the relative position of characters is weighted more heavily as you increase the *n*-gram length. By contrast, the relative position of characters is not considered when using an *n*-gram length of 1. Relative position refers to the position of characters in relation to one another, rather than to their absolute position within a string.

> **Tip**
> If you are specifically looking for transposition, use an *n*-gram length of 1.

n-gram length	"John Smith" *n*-grams	"Smith, John D." *n*-grams	Shared *n*-grams	Dice's coefficient
1	J | o | h | n | _ | S | m | i | t | h (10 *n*-grams)	S | m | i | t | h | , | _ | J | o | h | n | _ | D | . (14 *n*-grams)	10	2x10 / (10+14) = **0.8333**

n-gram length	"John Smith" *n*-grams	"Smith, John D." *n*-grams	Shared *n*-grams	Dice's coefficient
2 (default)	Jo | oh | hn | n_ | _S | Sm | mi | it | th (9 *n*-grams)	Sm | mi | it | th | h, | ,_ | _J | Jo | oh | hn | n_ | _D | D. (13 *n*-grams)	8	2x8 / (9+13) = **0.7273**
3	Joh | ohn | hn_ | n_S | _ Sm | Smi | mit | ith (8 *n*-grams)	Smi | mit | ith | th, | h,_ | ,_J | _Jo | Joh | ohn | hn_ | n_D | _D. (12 *n*-grams)	6	2x6 / (8+12) = **0.6000**
4	John | ohn_ | hn_S | n_ Sm | _Smi | Smit | mith (7 *n*-grams)	Smit | mith | ith, | th,_ | h,_J | ,_Jo | _ Joh | John | ohn_ | hn_D | n_D. (11 *n*-grams)	4	2x4 / (7+11) = **0.4444**

DICECOEFFICIENT( ) compared to ISFUZZYDUP ( ) and LEVDIST( )

One of the key differences between the DICECOEFFICIENT( ) function and the ISFUZZYDUP( ) and LEVDIST( ) functions, which use Levenshtein distance, is that DICECOEFFICIENT( ) de-emphasizes or completely ignores the relative position of characters or character blocks in the two strings being compared. Relative position is significant in the functions based on Levenshtein distance.

Comparison values with transposition

If you are comparing strings such as addresses, in which entire elements might be transposed, DICECOEFFICIENT( ) might be a better choice. For example, the same address with the "Suite" element transposed is identified as highly similar by DICECOEFFICIENT( ), but highly different by LEVDIST( ):

Address pair	Dice's coefficient (default *n*-gram of 2)	Levenshtein distance
◦ 125 SW 39TH ST, Suite 100 ◦ Suite 100, 125 SW 39TH ST	0.9167	22 (the greater the Levenshtein distance, the more two strings differ)

Comparison values without transposition

If transposition is not an issue, LEVDIST( ) may give more useful results. For example, the same corporation name with different punctuation is identified as highly different by DICECOEFFICIENT( ), but highly similar by LEVDIST( ):

Corporation name pair	Dice's coefficient (default *n*-gram of 2)	Levenshtein distance
◦ AVS, Inc ◦ A.V.S. Inc	0.3750	3

DIGIT( ) function

Returns the upper or lower digit of a specified Packed data type byte.

Syntax

```
DIGIT(byte_location, position)
```

Parameters

Name	Type	Description
byte_location	numeric	The byte position in the record.
position	numeric	The digit to return: ◦ specify `1` to return the upper half of the byte ◦ specify `2` to return the lower half of the byte

Output

Numeric.

Examples

Basic examples

A packed field with the value 123.45 (00 12 34 5C), containing two decimals, and starting in byte position 10, appears in the data record in the following format:

	Byte 10	Byte 11	Byte 12	Byte 13
UPPER(1)	0	1	3	5
LOWER(2)	0	2	4	C

Returns 3 (finds the digit that appears in the 12th position in the upper half of the byte):

```
DIGIT(12, 1)
```

Returns 4 (finds digit that appears in the 12th position in the lower half of the byte):

```
DIGIT(12, 2)
```

Remarks

How it works

DIGIT( ) separates individual halves of a byte, and returns the value of the byte specified in the position parameter as a digit between 0 and 15.

When to use DIGIT( )

Use DIGIT( ) to access individual half-bytes. This is required if you work with applications that use half-byte-aligned packed fields, such as Unisys applications.

DOW( ) function

Returns a numeric value (1 to 7) representing the day of the week for a specified date or datetime. Abbreviation for "Day of Week".

Syntax

```
DOW(date/datetime)
```

Parameters

Name	Type	Description
date/datetime	datetime	The field, expression, or literal value to extract the numeric day of the week from.

Output

Numeric.

Examples

Basic examples

Returns 4, because December 31, 2014 falls on a Wednesday, the 4th day of the week:

```
DOW(`20141231`)
```

```
DOW(`20141231 235959`)
```

Returns the numeric day of the week for each value in the **Invoice_date** field:

```
DOW(Invoice_date)
```

Advanced examples

Identifying transactions occurring on a weekend

Use the DOW( ) function to identify transactions that occur on a weekend. The filter below isolates dates in the **Trans_Date** field that occur on a Saturday or a Sunday:

```
SET FILTER TO DOW(Trans_Date) = 7 OR DOW(Trans_Date) = 1
```

Remarks

Parameter details

A field specified for *date/datetime* can use any date or datetime format, as long as the field definition correctly defines the format.

Specifying a literal date or datetime value

When specifying a literal date or datetime value for *date/datetime*, you are restricted to the formats in the table below, and you must enclose the value in backquotes – for example, `` `20141231` ``.

Do not use any separators such as slashes (/) or colons (:) between the individual components of dates or times.

- **Datetime values** – you can use any combination of the date, separator, and time formats listed in the table below. The date must precede the time, and you must use a separator between the two. Valid separators are a single blank space, the letter 't', or the letter 'T'.
- **Time values** – you must specify times using the 24-hour clock. Offsets from Coordinated Universal Time (UTC) must be prefaced by a plus sign (+) or a minus sign (-).

Example formats	Example literal values
YYYYMMDD	`20141231`
YYMMDD	`141231`

Example formats	Example literal values
YYYYMMDD hhmmss	`20141231 235959`
YYMMDDthhmm	`141231t2359`
YYYYMMDDThh	`20141231T23`
YYYYMMDD hhmmss+/-hhmm (UTC offset)	`20141231 235959-0500`
YYMMDD hhmm+/-hh (UTC offset)	`141231 2359+01`
Note Do not use hh alone in the main time format with data that has a UTC offset. For example, avoid: hh+hhmm. Results can be unreliable.	

Related functions

If you need to return:

- the name of the day of the week, use CDOW( ) instead of DOW( )
- the day of the month as a number (1 to 31), use DAY( ) instead of DOW( )

DTOU( ) function

Converts an Analytics date value to a Unicode string in the specified language and locale format. Abbreviation for "Date to Unicode".

> **Note**
> This function is specific to the Unicode edition of Analytics. It is not a supported function in the non-Unicode edition.

Syntax

```
DTOU(<date> <,locale> <,style>)
```

Parameters

Name	Type	Description
date optional	datetime	The field, expression, or literal value to convert to a Unicode string. If omitted, the current operating system date is used. The *date* can contain a datetime value, but the time portion of the value is ignored. Standalone time values are not supported. You can specify a field or a literal date value: ◦ **Field** - can use any date format, as long as the field definition correctly defines the format ◦ **Literal** - must use one of the YYYYMMDD or YYMMDD formats, for example `` `20141231` `` The minimum supported *date* value is 31 December 1969.
locale optional	character	The locale code that specifies the language of the output string, and optionally the version of the language associated with a particular country or region. For example, `"zh"` specifies Chinese, and `"pt_BR"` specifies Brazilian Portuguese. If omitted, the default locale for your computer is used. If a language is specified, but no country is specified, the default country for the language is used. You cannot specify *locale* if you have not specified *date*. For information about locale codes, see www.unicode.org.
style	numeric	The date format style to use for the Unicode string. The format

Name	Type	Description
optional		style matches the standard for the locale you specify: ◦ `0` - full specification format, such as "Sunday, September 18, 2016" ◦ `1` - long format, such as "September 18, 2016" ◦ `2` - medium format, such as "Sep 18, 2016" ◦ `3` - short, numeric format such as "9/18/16" If omitted, the default value of 2 is used. You cannot specify *style* if you have not specified *date* and *locale*.

Output

Character.

Examples

Basic examples

Literal input values

Returns "31 de dezembro de 2014":

```
DTOU(`20141231`, "pt_BR", 1)
```

Returns "31 grudnia 2014":

```
DTOU(`20141231`, "pl", 1)
```

Field input values

Returns each numeric date in the **Invoice_date** field as a Unicode string:

```
DTOU(Invoice_date, "zh", 1)
```

Output uses full date style

Returns "星期三, 2014 十二月 31" (no region identifier specified):

```
DTOU(`20141231`, "zh", 0)
```

Returns "2014年12月31日星期三" (region identifier specified):

```
DTOU(`20141231`, "zh_CN", 0)
```

Output uses long date style

Returns "2014 十二月 31" (no region identifier specified):

```
DTOU(`20141231`, "zh", 1)
```

Returns "2014年12月31日" (region identifier specified):

```
DTOU(`20141231`, "zh_CN", 1)
```

Remarks

Related functions

DTOU( ) is the inverse of the UTOD( ) function, which converts a Unicode string to a date.

EBCDIC( ) function

Returns a string that has been converted to EBCDIC character encoding.

Syntax

```
EBCDIC(string)
```

Parameters

Name	Type	Description
string	character	The value to convert to EBCDIC.

Output

Character.

Examples

Basic examples

Returns "ñòó@Æ'...@â£K":

```
EBCDIC("123 Fake St.")
```

Advanced examples

Creating an EBCDIC-encoded field to export

To create a field containing the EBCDIC encoded value of a **Name** field for export to an application that requires EBCDIC encoding, specify the following:

```
DEFINE FIELD Name_Exp COMPUTED EBCDIC(Name)
```

Remarks

When to use EBCDIC( )

Use this function to convert data to the Extended Binary Coded Decimal Interchange Code (EBCDIC) character encoding. EBCDIC character encoding is used primarily on IBM mainframe operating systems, such as z/OS.

EFFECTIVE( ) function

Returns the effective annual interest rate on a loan.

Syntax

```
EFFECTIVE(nominal_rate, periods)
```

Parameters

Name	Type	Description
nominal_rate	numeric	The nominal annual interest rate.
periods	numeric	The number of compounding periods per year. **Note** Specify an integer. If you specified a decimal portion, it is truncated.

Output

Numeric. The rate is calculated to eight decimals places.

Examples

Basic examples

Returns 0.19561817 (19.56%), the effective annual rate of interest on the unpaid balance of a credit card that charges 18% per annum, compounded monthly:

```
EFFECTIVE(0.18, 12)
```

Remarks

What is the effective annual interest rate?

The effective annual interest rate on a loan is the actual annual rate of interest paid, taking into account interest on the remaining balance, compounded monthly or daily.

Related functions

The NOMINAL( ) function is the inverse of the EFFECTIVE( ) function.

EOMONTH( ) function

Returns the date of the last day of the month that is the specified number of months before or after a specified date.

Syntax

```
EOMONTH(<date/datetime> <,months>)
```

Parameters

Name	Type	Description
date/datetime optional	datetime	The field, expression, or literal value from which to calculate the end-of-month date. If omitted, the end-of-month date is calculated from the current operating system date. **Note** You can specify a datetime value for *date/datetime* but the time portion of the value is ignored.
months optional	numeric	The number of months before or after *date/datetime*. If omitted, the default of 0 (zero) is used. You cannot specify *months* if you have omitted *date/datetime*.

Output

Datetime. The date value is output using the current Analytics date display format.

Examples

Basic examples

No input

Returns the last day of the month for the current operating system date:

```
EOMONTH()
```

Literal input values

Returns `20140131` displayed as 31 Jan 2014 assuming a current Analytics date display format of DD MMM YYYY:

```
EOMONTH(`20140115`)
```

Returns `20140430` displayed as 30 Apr 2014 assuming a current Analytics date display format of DD MMM YYYY:

```
EOMONTH(`20140115`, 3)
```

Returns `20131031` displayed as 31 Oct 2013 assuming a current Analytics date display format of DD MMM YYYY:

```
EOMONTH(`20140115`, -3)
```

Field input values

Returns the last day of the month that falls three months after each date in the **Invoice_date** field:

```
EOMONTH(Invoice_date, 3)
```

Returns the last day of the month that falls three months after each date in the **Invoice_date** field plus a grace period of 15 days:

```
EOMONTH(Invoice_date + 15, 3)
```

Returns the first day of the month in which the invoice date falls:

```
EOMONTH(Invoice_date, -1) + 1
```

Remarks

Datetime formats

A field specified for *date/datetime* can use any date or datetime format, as long as the field definition correctly defines the format.

A literal date value must use one of the following formats:

- YYYYMMDD
- YYMMDD

You must enclose literal date values in backquotes. For example: `` `20141231` ``

How the *months* value works

- **Positive value** – the output date is more recent than the specified *date/datetime*
- **Negative value** – the output date is prior to the specified *date/datetime*
- **Value omitted, or '0' (zero)** – the output date is the last day of the month in which the *date/datetime* occurs

Return the date of the first day of a month

Add 1 day to the result of the EOMONTH( ) function to return the date of the first day of a month.

Returns `20140201` displayed as 01 Feb 2014 assuming a current Analytics date display format of DD MMM YYYY:

```
EOMONTH(`20140115`) + 1
```

Related functions

Use the GOMONTH( ) function if you want to return the exact date, rather than the date of the last day of the month, that is the specified number of months before or after a specified date.

EXCLUDE( ) function

Returns a string that excludes the specified characters.

Syntax

```
EXCLUDE(string, characters_to_exclude)
```

Parameters

Name	Type	Description
string	character	The field, expression, or literal value from which to exclude characters.
characters_to_exclude	character	The list of characters to exclude. If you specify double quotation marks in *characters_to_exclude*, you must enclose the list of characters in single quotation marks. For example: `'"-/'`

Output

Character.

Examples

Basic examples

Returns " Alberni Street", which is the input string with all numbers excluded:

```
EXCLUDE("1550 Alberni Street", "0123456789")
```

Returns all the values in the **Product_Number** field with the forward slash and number sign excluded:

```
EXCLUDE(Product_Number, "/#")
```

Remarks

How it works

The EXCLUDE( ) function compares each character in *string* with the characters listed in *characters_to_exclude*. If a match occurs, the character is excluded from the output string.

For example, the output for `EXCLUDE("123-45-4536", "-")` is "123454536".

No matching characters

If there are no matches between *string* and *characters_to_exclude*, then *string* and the output of the function are the same.

For example, the output for `EXCLUDE("ABC", "D")` is "ABC".

Case sensitivity

The EXCLUDE( ) function is case-sensitive. If you specify "ID" in *characters_to_exclude*, these characters are not excluded from "id#94022". If there is a chance the case may be mixed, use the UPPER( ) function to convert *string* to uppercase.

For example:

```
EXCLUDE(UPPER("id#94022"), "ID#")
```

Usage tip

Use EXCLUDE( ) if the set of characters you want to exclude is small, and the set you want to include is large.

Excluding both single and double quotation marks

Quotation marks are used as string delimiters, therefore to exclude both single and double quotation marks you must nest EXCLUDE( ) so that there is a single function for each type of quotation mark:

```
EXCLUDE(EXCLUDE(field_to_process, "'"), '"')
```

Related functions

The EXCLUDE( ) function is the opposite of the INCLUDE( ) function.

EXP( ) function

Returns the exponential value (base 10) of a numeric expression with a specified number of decimal places.

Syntax

```
EXP(number, decimals)
```

Parameters

Name	Type	Description
number	numeric	The numeric field, expression, or value to return the exponential value of.
decimals	numeric	The number of decimals to include in the return value.

Output

Numeric.

Examples

Basic examples

Returns 1000.00:

```
EXP(3, 2)
```

Returns 72443.596007:

```
EXP(4.86, 6)
```

Advanced examples

Finding the cube root

Creates a field that is the cube root of the field X to two decimal places:

```
DEFINE FIELD cube_root COMPUTED EXP(LOG(X, 6) / 3, 2)
```

> Tip
>
> You can determine the *n*th root by dividing the log of the value by *n* and taking the exponential of the result.

Remarks

How it works

This function returns the exponential value (base 10) of a numeric expression, which is defined as 10 raised to the *n*th power. For example, the exponential value of 3 is 10^3, or 1000.

When to use EXP( )

Use EXP( ) for financial applications requiring complex mathematical calculations. EXP( ) performs the same operation as the exponentiation operator (^), but can be useful in applications that also use the LOG( ) function.

Related functions

The inverse of an exponent is its logarithm, so EXP( ) is the opposite of the LOG( ) function.

FILESIZE( ) function

Returns the size of the specified file in bytes or -1 if the file does not exist.

Syntax

```
FILESIZE(filename)
```

Parameters

Name	Type	Description
filename	character	The name of the file. If the file is in the same folder as the Analytics project, you do not need to specify the file path. For files in other folders, specify either a relative path or an absolute path. For example: ◦ `"results\test_output.fil"` ◦ `"c:\results\test_output.fil"` **Note** You need to specify the physical data file name (.fil) for Analytics tables, not the table name.

Output

Numeric.

Examples

Basic examples

Returns 14744:

```
FILESIZE("Inventory.fil")
```

If the file you are checking is not in the same folder as the Analytics project, you must specify either the relative path or absolute path to the file.

Returns 6018:

```
FILESIZE("C:\ACL Data\Sample Data Files\Backup\Ap_Trans.fil")
```

Advanced examples

Executing a script if a file does not exist

Only executes the script **import_data** if the file `Metaphor_Inventory_2002.fil` does not exist:

```
DO SCRIPT import_data IF FILESIZE("Metaphor_Inventory_2002.fil") = -1
```

Recording a file's size in the Analytics command log

Use the CALCULATE command to record the size of `Metaphor_Inventory_2002.fil` in the Analytics command log:

```
CALCULATE FILESIZE("Metaphor_Inventory_2002.fil")
```

FIND( ) function

Returns a logical value indicating whether the specified string is present in a particular field, or anywhere in an entire record.

> **Note**
> The FIND( ) function and the "FIND command" on page 1721 are two separate Analytics features with significant differences.

Syntax

```
FIND(string <,field_to_search_in>)
```

Parameters

Name	Type	Description
string	character	The character string to search for. This search is not case-sensitive.
field_to_search_in optional	character	The field, or variable, to search in. If omitted, the entire record is searched, including any undefined portion of the record.

Output

Logical. Returns **T** (true) if the specified *string* value is found, and **F** (false) otherwise.

Examples

Basic examples

Searching an entire record

Returns T for all records that contain the string "New York" in any field, across any field boundaries, and in any undefined portion of the record. Returns F otherwise:

```
FIND("New York")
```

Searching a single field

Returns T for all records that contain the string "New York" in the **City** field. Returns F otherwise.

```
FIND("New York", City)
```

Returns T for all records that contain the string "Ne" in the **City** field. Returns F otherwise:

```
FIND("Ne", City)
```

Returns T for all records that contain the string "New York" preceded by one or more spaces in the **City** field. Returns F otherwise:

```
FIND(" New York", City)
```

Returns T for all records that have a value in the **Description** field that matches, or contains, the value in the *v_search_term* variable. Returns F otherwise:

```
FIND(v_search_term, Description)
```

Searching multiple fields

Returns T for all records that contain the string "New York" in either the **City** or the **City_2** fields. Returns F otherwise:

```
FIND("New York", City+City_2)
```

Returns T for all records that contain the string "New York" in either the **City** or the **City_2** fields. Returns F otherwise:

```
FIND("New York", City) OR FIND("New York", City_2)
```

Combining with other functions

Returns T for all records that have a value in the **Last_Name_2** field that matches, or contains, the trimmed value from the **Last_Name** field. Returns F otherwise:

```
FIND(ALLTRIM(Last_Name), Last_Name_2)
```

Remarks

When to use FIND( )

Use the FIND( ) function to test for the presence of the specified *string* in a field, two or more fields, or an entire record.

How matching works

The *string* value can be exactly matched or it can be contained within a longer string. Leading spaces in fields do not affect the search unless you include one or more leading spaces in the *string* value.

Search an entire record

If the optional *field_to_search_in* is not specified, the entire record is searched, including any undefined portion of the record. Field boundaries are ignored when the entire record is searched, and trailing spaces in fields are treated as characters.

> **Note**
> When you search an entire record, the physical record is searched. Any computed fields or related fields are not searched.

Search a subset of fields

You can concatenate two or more fields in the *field_to_search_in* if you want to search in a subset of the fields in a table. For example, to search both the **City** and **City_2** fields for the string "New York":

```
FIND("New York", City+City_2)
```

The concatenated fields are treated like a single field that includes leading and trailing spaces from the individual fields, unless you use the ALLTRIM( ) function to remove spaces.

You can also build an expression that searches each field individually:

```
FIND("New York", City) OR FIND("New York", City_2)
```

If *string* includes a leading space, search results from the two approaches can differ.

Case sensitivity and Exact Character Comparisons

The FIND( ) function is not case-sensitive, and finds both ASCII and EBCDIC characters. The function is not affected by the **Exact Character Comparisons** option (SET EXACT ON/OFF).

Search in a computed field

To search in a computed field you must specify the name of the field in *field_to_search_in*. For example, if **Vendor_City** is a computed field that isolates the city in an address:

```
FIND("New York", Vendor_City)
```

Search in a related field

To search in a related field you must specify the fully qualified name of the field (that is, *table.field name*) in the *field_to_search_in* value:

```
FIND("New York", Vendor.Vendor_City)
```

Search datetime or numeric data

It is possible to use the FIND( ) function to search datetime or numeric data at the record level. Specifying the *field_to_search_in* is not supported for datetime or numeric searching.

The numeric or datetime *string* must be enclosed in quotation marks, and needs to exactly match the source data formatting rather than the formatting in the view.

Using the FIND( ) function to search datetime or numeric data in computed or related fields is not supported.

> **Note**
>
> Using the FIND( ) function to search datetime or numeric data is not recommended because it can be difficult to do successfully.

FINDMULTI( ) function

Returns a logical value indicating whether any string in a set of one or more specified strings is present in a particular field, or anywhere in an entire record.

Syntax

```
FINDMULTI({search_in|RECORD}, string_1 <,...n>)
```

Parameters

Name	Type	Description
search_in | RECORD	character	The field, or variable, to search in. Specify the keyword `RECORD` to search the entire record, including any undefined portion of the record. You can also specify a list of fields by concatenating field names: `Field_1+Field_2+Field_3`
string_1 <,...n>	character	One or more character strings to search for. Separate multiple search strings with commas: `FINDMULTI(RECORD, "Joa", "Jim", "Joh")` The search is not case-sensitive.

Output

Logical. Returns **T** (true) if any of the specified *string* values are found, and **F** (false) otherwise.

Examples

Basic examples

Searching an entire record

Returns T for all records that contain "New York" or "Chicago" in any field, across any field boundaries, and in any undefined portion of the record. Returns F otherwise:

```
FINDMULTI(RECORD, "New York", "Chicago")
```

Searching a single field

Returns T for all records that contain "New York" or "Chicago" in the **City** field. Returns F otherwise:

```
FINDMULTI(City, "New York", "Chicago")
```

Returns T for all records that contain the string "Ne" or "Chi" in the **City** field. Returns F otherwise:

```
FINDMULTI(City, "Ne", "Chi")
```

Returns T for all records that contain "New York" or "Chicago" preceded by one or more spaces in the **City** field. Returns F otherwise:

```
FINDMULTI(City, " New York", " Chicago")
```

Returns T for all records that have a value in the **Description** field that matches, or contains, any of the values in the *v_search_term* variables . Returns F otherwise:

```
FINDMULTI(Description, v_search_term_1, v_search_term_2, v_search_term_3)
```

Searching multiple fields

Returns T for all records that contain the string "New York" or "Chicago" in either the **City** or the **City_2** fields. Returns F otherwise:

```
FINDMULTI(City+City_2, "New York", "Chicago")
```

Returns T for all records that contain the string "New York" or "Chicago" in either the **City** or the **City_2** fields. Returns F otherwise:

```
FINDMULTI(City, "New York", "Chicago") OR FINDMULTI(City_2, "New York",
"Chicago")
```

Combining with other functions

Returns T for all records that have a value in the **Last_Name_1** field that matches, or contains, the trimmed value from the **Last_Name_2** or **Last_Name_3** fields. Returns F otherwise:

```
FINDMULTI(Last_Name_1, ALLTRIM(Last_Name_2), ALLTRIM(Last_Name_3))
```

Remarks

When to use FINDMULTI( )

Use the FINDMULTI( ) function to test for the presence of any of the specified strings in a field, two or more fields, or an entire record.

How matching works

The *string* value can be exactly matched or it can be contained within a longer string. Leading spaces in fields do not affect the search unless you include one or more leading spaces in the *string* value.

Search an entire record

If you specify RECORD instead of a *search_in* field, the entire record is searched, including any undefined portion of the record. Field boundaries are ignored when the entire record is searched, and trailing spaces in fields are treated as characters.

> **Note**
> When you search an entire record, the physical record is searched. Any computed fields or related fields are not searched.

Search a subset of fields

You can concatenate two or more fields in the *search_in* parameter if you want to search in a subset of the fields in a table. For example, to search both the **City** and the **City_2** fields for the strings "New York" or "Chicago":

```
FINDMULTI(City+City_2, "New York", "Chicago")
```

The concatenated fields are treated like a single field that includes leading and trailing spaces from the individual fields, unless you use the ALLTRIM( ) function to remove spaces.

You can also build an expression that searches each field individually:

```
FINDMULTI(City, "New York", "Chicago") OR FINDMULTI(City_2, "New York",
"Chicago")
```

If a *string* value includes a leading space, search results from the two approaches can differ.

Case sensitivity and Exact Character Comparisons

The FINDMULTI( ) function is not case-sensitive, and finds both ASCII and EBCDIC characters. The function is not affected by the **Exact Character Comparisons** option (SET EXACT ON/OFF).

Search in a computed field

To search in a computed field you must specify the name of the field in *search_in*. For example, if **Vendor_City** is a computed field that isolates the city in an address:

```
FINDMULTI(Vendor_City, "New York", "Chicago")
```

Search in a related field

To search in a related field you must specify the fully qualified name of the field (that is, *table.field name*) in the *search_in* value:

```
FINDMULTI(Vendor.Vendor_City, "New York", "Chicago")
```

Search datetime or numeric data

It is possible to use the FINDMULTI( ) function to search datetime or numeric data at the record level, when specifying RECORD. Specifying a *search_in* field is not supported for datetime or numeric searching.

The numeric or datetime *string* values must be enclosed in quotation marks, and need to exactly match the source data formatting rather than the formatting in the view.

Using the FINDMULTI( ) function to search datetime or numeric data in computed or related fields is not supported.

> **Note**
> Using the FINDMULTI( ) function to search datetime or numeric data is not recommended because it can be difficult to do successfully.

FREQUENCY( ) function

Returns the expected Benford frequency for sequential leading positive numeric digits to a precision of eight decimal places.

Syntax

```
FREQUENCY(digit_string)
```

Parameters

Name	Type	Description
digit_string	character	A character string containing digits (0-9) to identify the frequency for. *digit_string* must be a positive number, and leading zeros are ignored.

Output

Numeric.

Examples

Basic examples

Returns 0.00998422:

```
FREQUENCY("43")
```

Returns 0.00000000:

```
FREQUENCY("87654321")
```

> **Note**
> The result is 0.00000000495, but because Analytics computes to a precision of eight decimal places, a zero value is returned.

Remarks

How it works

FREQUENCY( ) returns the expected Benford frequency for sequential leading positive numeric digits to a precision of eight digits. It lets you perform limited Benford tests for specific situations.

Using this function for specific digit combinations

You can use this function instead of the BENFORD command if you want to focus on specific digit combinations. For example, when auditing insurance claims that have approval limits at specified claim amounts, you could use the FREQUENCY( ) function to investigate amounts just under an approval threshold.

To investigate claims valued close to an approval limit of $5,000, you could select the range from $4,900 through $4,999. First, count the total number of records, then use a filter to count the records for which LEADING( ) returns 49, and compare the ratio of the two counts to the value you get for FREQUENCY("49").

This is faster than running a complete analysis on a table of a million records, and it does not generate a large table or lengthy entries in the command log.

Specifying strings longer than six digits

Specifying strings longer than six digits can result in zero values. Calculations for strings longer than six digits may require greater precision than Analytics's limit of eight decimal places.

FTYPE( ) function

Returns a character identifying the data category of a field or variable, or the type of an Analytics project item.

Syntax

```
FTYPE(field_name_string)
```

Parameters

Name	Type	Description
field_name_string	character	A field name, variable name, or Analytics project item name. Enclose *field_name_string* in quotation marks: `FTYPE("Amount")`

Output

Character. This function returns one of the following characters, which indicates the field, variable, or Analytics project item type:

- **"C"** - Character field
- **"N"** - Numeric field
- **"D"** - Datetime field
- **"L"** - Logical field
- **"c"** - Character variable
- **"n"** - Numeric variable
- **"d"** - Datetime variable
- **"l"** - Logical variable
- **"b"** - Analytics script
- **"y"** - Analytics table layout
- **"w"** - Analytics workspace
- **"i"** - Analytics index
- **"r"** - Analytics report
- **"a"** - Analytics log file
- **"U"** - Undefined

Examples

Basic examples

The following example assigns a value of 4 to the *num* variable and then checks the type.

Returns "n":

```
ASSIGN num = 4
FTYPE("num")
```

Advanced examples

Testing for the data type of a field

You have a script or analytic that requires a numeric **Amount** field, and you need to test that the field is the correct type before running the script.

The following command only runs Script_1 if **Amount** is a numeric field:

```
OPEN Invoices
DO Script_1 IF FTYPE("Amount") = "N"
```

Testing if a table or Analytics project item exists

The following command only runs Script_1 if a table named Invoices is present in the project:

```
DO Script_1 IF FTYPE("Invoices") <> "U"
```

Testing the runtime environment

You can use FTYPE to determine if an analytic is running in Analytics, or on Analytics Exchange or in the Analysis App window.

If an analytic is running on Analytics Exchange, or in the Analysis App window, 'ax_main' is equal to 'b':

```
IF FTYPE('ax_main') = 'b' v_running_in_ax_or_analysis_app = T
```

If an analytic is running in Analytics, 'ax_main' is not equal to 'b':

```
IF FTYPE('ax_main') <> 'b' v_running_in_ax_or_analysis_app = F
```

The ability to detect the runtime environment allows you to design a single script that executes different blocks of codes depending on which application it is running in.

FVANNUITY( ) function

Returns the future value of a series of payments calculated using a constant interest rate. Future value is the sum of the payments plus the accumulated compound interest.

Syntax

```
FVANNUITY(rate, periods, payment <,type>)
```

Parameters

Name	Type	Description
rate	numeric	The interest rate per period.
periods	numeric	The total number of payment periods.
payment	numeric	The payment per period. The payment amount must remain constant over the term of the annuity.
type optional	numeric	The timing of payments: ◦ `0` - payment at the end of a period ◦ `1` - payment at the beginning of a period If omitted, the default value of 0 is used.

> **Note**
> You must use consistent time periods when specifying *rate*, *periods*, and *payment* to ensure that you are specifying interest rate **per period**.
>
> For example:
>
> - for a monthly *payment* on a two-year loan or investment with interest of 5% per annum, specify 0.05/12 for *rate* and 2 * 12 for *periods*
> - for an annual *payment* on the same loan or investment, specify 0.05 for *rate* and 2 for *periods*

Output

Numeric. The result is calculated to two decimal places.

Examples

Basic examples

Monthly payments

Returns 27243.20, the future value of $1,000 paid at the beginning of each month for 2 years at 1% per month, compounded monthly:

```
FVANNUITY(0.01, 2*12, 1000, 1)
```

Returns 12809.33, the future value of the same annuity after the first year:

```
FVANNUITY(0.01, 12, 1000, 1)
```

Annual payments

Returns 25440.00, the future value of $12,000 paid at the end of each year for 2 years at 12% per annum, compounded annually:

```
FVANNUITY(0.12, 2, 12000, 0)
```

Advanced examples

Annuity calculations

Annuity calculations involve four variables:

- **present value, or future value** - $21,243.39 and $ 26,973.46 in the examples below
- **payment amount per period** - $1,000.00 in the examples below
- **interest rate per period** - 1% per month in the examples below
- **number of periods** - 24 months in the examples below

If you know the value of three of the variables, you can use an Analytics function to calculate the fourth.

I want to find:	Analytics function to use:
Present value	PVANNUITY( ) Returns 21243.39: `PVANNUITY(0.01, 24, 1000)`
Future value	FVANNUITY( ) Returns 26973.46: `FVANNUITY(0.01, 24, 1000)`
Payment amount per period	PMT( ) Returns 1000: `PMT(0.01, 24, 21243.39)`
Interest rate per period	RATE( ) Returns 0.00999999 (1%): `RATE(24, 1000, 21243.39)`
Number of periods	NPER( ) Returns 24.00: `NPER(0.01, 1000, 21243.39)`

Annuity formulas

The formula for calculating the **present value** of an ordinary annuity (payment at the end of a period):

$$PV_A = Pmt\left[\frac{1-\frac{1}{(1+i)^N}}{i}\right]$$

$$21243.39 = 1000\left[\frac{1-\frac{1}{(1+0.01)^{24}}}{0.01}\right]$$

The formula for calculating the **future value** of an ordinary annuity (payment at the end of a period):

$$FV_A = Pmt\left[\frac{(1+i)^N - 1}{i}\right]$$

$$26973.46 = 1000\left[\frac{(1+0.01)^{24} - 1}{0.01}\right]$$

Remarks

Related functions

The PVANNUITY( ) function is the inverse of the FVANNUITY( ) function.

FVLUMPSUM( ) function

Returns the future value of a current lump sum calculated using a constant interest rate.

Syntax

```
FVLUMPSUM(rate, periods, amount)
```

Parameters

Name	Type	Description
rate	numeric	The interest rate per period.
periods	numeric	The total number of periods.
amount	numeric	The investment made at the start of the first period.

> **Note**
> You must use consistent time periods when specifying *rate* and *periods* to ensure that you are specifying interest rate **per period**.
>
> For example:
>
> - for monthly payments on a two-year loan or investment with interest of 5% per annum, specify 0.05/12 for *rate* and 2 * 12 for *periods*
> - for annual payments on the same loan or investment, specify 0.05 for *rate* and 2 for *periods*

Output

Numeric. The result is calculated to two decimal places.

Examples

Basic examples

Interest compounded monthly

Returns 1269.73, the future value of a lump sum of $1,000 invested for 2 years at 1% per month, compounded monthly:

```
FVLUMPSUM(0.01, 2*12, 1000)
```

Returns 1126.83, the future value of the same investment after the first year:

```
FVLUMPSUM(0.01, 12, 1000)
```

Returns 27243.20, the future value of $21,455.82 invested for 2 years at 1% per month, compounded monthly:

```
FVLUMPSUM(0.01, 2*12, 21455.82)
```

Interest compounded semi-annually

Returns 1262.48, the future value of a lump sum of $1,000 invested for 2 years at 12% per annum, compounded semi-annually:

```
FVLUMPSUM(0.12/2, 2*2, 1000)
```

Interest compounded annually

Returns 1254.40, the future value of a lump sum of $1,000 invested for 2 years at 12% per annum, compounded annually:

```
FVLUMPSUM(0.12, 2, 1000)
```

Remarks

What is future value?

The future value of an invested lump sum is the initial investment principal plus the accumulated compound interest.

Related functions

The PVLUMPSUM( ) function is the inverse of the FVLUMPSUM( ) function.

FVSCHEDULE( ) function

Returns the future value of a current lump sum calculated using a series of interest rates.

Syntax

```
FVSCHEDULE(principal, rate1 <,rate2...>)
```

Parameters

Name	Type	Description
principal	numeric	The amount of the initial investment.
rate1, rate2...	numeric	A series of interest rates for equal-length periods. **Note** The periods can represent months or years, or some other time period, as long as the type of time period is consistent. You must specify the interest rates **per period**. So, if one of the interest rates is 5% per annum and the periods are months, specify 0.05/12.

Output

Numeric. The result is calculated to two decimal places.

Examples

Basic examples

Returns 1282.93, the future value of a lump sum of $1000 invested for 3 years at 10% for the first year, 9% for the second year, and 7% for the third year, compounded annually:

```
FVSCHEDULE(1000, 0.1, 0.09, 0.07)
```

Remarks

The future value of an invested lump sum is the initial investment principal plus the accumulated compound interest.

GETOPTIONS( ) function

Returns the current setting for the specified Analytics option (**Options** dialog box setting).

Syntax

```
GETOPTIONS(option)
```

Parameters

Name	Type	Description
option	character	The Analytics option to return a setting for. The name of the option must be specified exactly as it appears in the list below, and it must be enclosed in quotation marks: ◦ **separators** – returns the current settings for the three Analytics separator characters, in the following order: • decimal place symbol • thousands separator • list separator **Note** Currently, `"separators"` is the only *option* that can be specified for the GETOPTIONS( ) function.

Output

Character.

Examples

Basic examples

Returns the current settings for the three Analytics separator characters. For example, ".,,":

```
GETOPTIONS("separators")
```

Advanced examples

Using GETOPTIONS( ) in a script

If a script needs to change one or more of the Analytics separator characters, the GETOPTIONS( ) function provides a method for discovering the current settings. The current settings can be stored in a variable and then reinstated at the end of the script.

```
ASSIGN v_SeparatorsSetting = GETOPTIONS("separators")
SET SEPARATORS ",.;"
<script content>
SET SEPARATORS "%v_SeparatorsSetting%"
```

Remarks

The three Analytics separator characters are specified in the following options in the **Options** dialog box:

- **Decimal Place Symbol**
- **Thousands Separator**
- **List Separator**

GOMONTH( ) function

Returns the date that is the specified number of months before or after a specified date.

Syntax

```
GOMONTH(date/datetime, months)
```

Parameters

Name	Type	Description
date/datetime	datetime	The field, expression, or literal value from which to calculate the output date.
months	numeric	The number of months before or after *date/datetime*. **Note** You can specify a datetime value for *date/datetime* but the time portion of the value is ignored.

Output

Datetime. The date value is output using the current Analytics date display format.

Examples

Basic examples

Literal input values

Returns `20140415` displayed as 15 Apr 2014 assuming a current Analytics date display format of DD MMM YYYY:

```
GOMONTH(`20140115`, 3)
```

Returns `20131015` displayed as 15 Oct 2013 assuming a current Analytics date display format of DD MMM YYYY:

```
GOMONTH(`20140115`, -3)
```

Returns `20140430` displayed as 30 Apr 2014 assuming a current Analytics date display format of DD MMM YYYY (date rounding prevents returning 31 Apr 2014, which is an invalid date):

```
GOMONTH(`20140330`, 1)
```

```
GOMONTH(`20140331`, 1)
```

Returns `20140501` displayed as 01 May 2014 assuming a current Analytics date display format of DD MMM YYYY:

```
GOMONTH(`20140401`, 1)
```

Field input values

Returns the date three months after each date in the **Invoice_date** field:

```
GOMONTH(Invoice_date, 3)
```

Returns the date three months after each date in the **Invoice_date** field plus a grace period of 15 days:

```
GOMONTH(Invoice_date + 15, 3)
```

Remarks

Datetime formats

A field specified for *date/datetime* can use any date or datetime format, as long as the field definition correctly defines the format.

A literal date value must use one of the following formats:

- YYYYMMDD
- YYMMDD

You must enclose literal date values in backquotes. For example: `` `20141231` ``

How the *months* value works

- **Positive value** – the output date is more recent than the specified *date/datetime*
- **Negative value** – the output date is prior to the specified *date/datetime*
- **Value omitted, or '0' (zero)** – the output date is the same as the *date/datetime*

Date rounding to avoid non-existent dates

If the combination of *date/datetime* and *months* would produce a non-existent date, the GOMONTH( ) function uses 'date rounding' to return the closest valid date within the same month.

Returns `20140430` (30 Apr 2014) because 31 Apr 2014 is an invalid date:

```
GOMONTH(`20140331`,1)
```

Related functions

Use the EOMONTH( ) function if you want to return the date of the last day of the month, rather than the exact date, that is the specified number of months before or after a specified date.

HASH( ) function

Returns a salted cryptographic hash value based on the input value.

Syntax

```
HASH(field <,salt_value>)
```

Parameters

Name	Type	Description
field	character numeric datetime logical	The value to hash.
salt_value optional	character numeric	The salt value to use. You can specify a `PASSWORD` identifier number from 1 to 10, or a character string. If omitted, the Analytics default salt value is used. The salt value is limited to 128 characters, and is automatically truncated to 128 characters if you specify a longer salt value. For more information, see "The salt value" on page 2178.

Output

Character.

Examples

Basic examples

With the Analytics default salt value

Returns "819A974BB91215D58E7753FD5A42226150100A0763087CA7DECD93F3C3090405":

```
HASH("555-44-3322")
```

Returns the hash value for each number in the **Credit_card_num** field:

```
HASH(Credit_card_num)
```

With a user-specified salt value

Returns "AD1E7D9B97B6F6B5345AB13471A74C31EBE6630CA2622BB7E8C280E9FBEE1F17":

```
HASH("555-44-3322", "my salt value 123")
```

Advanced examples

Ensuring hash values are identical

Use other functions in conjunction with HASH( ) to standardize clear text values that should produce identical hash values.

Consider the following set of examples. Note how the case of the clear text values completely alters the output hash value in the first two examples.

Returns "DF6789E1EC65055CD9CA17DD5B0BEA5892504DFE7661D258737AF7CB9DC46462":

```
HASH("John Smith")
```

Returns "3E12EABB5940B7A2AD90A6B0710237B935FAB68E629907927A65B3AA7BE6781D":

```
HASH("JOHN SMITH")
```

By using the UPPER( ) function to standardize case, an identical hash value results.

Returns "3E12EABB5940B7A2AD90A6B0710237B935FAB68E629907927A65B3AA7BE6781D":

```
HASH(UPPER("John Smith"))
```

Using HASH( ) to compare large blocks of text

Use HASH( ) to test if blocks of text in two comment fields are identical.

To perform this test, create two computed fields similar to the ones shown below, and then create a filter to find any text blocks that are not identical.

```
DEFINE FIELD Hash_1 COMPUTED HASH(Comment_Field_1)
DEFINE FIELD Hash_2 COMPUTED HASH(Comment_Field_2)
SET FILTER TO Hash_1 <> Hash_2
```

If the comment fields are in separate tables, create a computed HASH( ) field in each table and then use the computed fields as a common key field to do an unmatched join of the two tables. The records in the joined output table represent text blocks that are not identical.

Remarks

When to use HASH( )

Use the HASH( ) function to protect sensitive data, such as credit card numbers, salary information, or social security numbers.

How it works

HASH( ) provides one-way encoding. Data in clear text can be used to produce a hash value, however the hash value cannot subsequently be unencoded or decrypted.

A specific clear text value always produces the same hash value, so you can search a field of hashed credit card numbers for duplicates, or join two fields of hashed credit card numbers, and the results are the same as if you had performed the operation on the equivalent clear text fields.

Protecting sensitive data

To avoid storing sensitive data on a server, you can create a computed field locally using the HASH( ) function, and then create a new table by extracting the hashed field and any other required fields, while excluding the clear text field. You can use the new table on the server for your analysis, and once you have the results, refer back to the original table if you need to see the clear text version of any of the hashed data.

If storing sensitive data locally, beyond initial use, is prohibited, you can delete the original table after you have created the new table with the hashed values, and refer to the original source system for the clear text values.

Clear text values must be exactly identical

In order to produce identical hash values, two clear text values must be exactly identical. For example, different hash values result from the same credit card number with or without hyphens, or the same name in title case or all upper case.

You may need to incorporate functions such as INCLUDE( ), EXCLUDE( ), or UPPER( ) in the HASH ( ) function to standardize clear text values.

Leading and trailing blanks are automatically trimmed by the HASH( ) function, so there is no need to use the TRIM( ) or ALLTRIM( ) functions.

What if leading or trailing blanks are meaningful?

If you have data in which leading or trailing blanks represent meaningful differences between values you need to replace the blanks with another character before hashing the values.

Replaces blanks in the field values with the underscore character (_) before hashing:

```
HASH(REPLACE(field_name, " ", "_"))
```

The cryptographic algorithm used by HASH( )

HASH( ) uses an SHA-2 cryptographic hash algorithm that produces a fixed-length hashed output of 64 bytes, regardless of the length of the input value. The clear text input value can be longer than 64 bytes.

The salt value

How it works

The protection offered by the HASH( ) function is strengthened by the automatic addition of a salt value prior to hashing. The salt value is an alphanumeric string that is concatenated with the source data value. The entire concatenated string is then used to produce the salted, hashed value. This approach makes the hashed values more resistant to decoding techniques.

Optionally specify your own salt value

A fixed, default salt value is automatically used unless you specify a salt value. You can use either of the following methods to specify a salt value:

- **Salt value as clear text string**

  Specify an alphanumeric string. For example:

  ```
  HASH(Credit_card_num, "my salt value")
  ```

- **Salt value as password**

  Use the PASSWORD command in conjunction with the HASH( ) function and specify a PASSWORD identifier number from 1 to 10. For example:

  ```
  PASSWORD 3 "Enter a salt value"
  EXTRACT FIELDS HASH(Credit_card_num, 3) TO "Protected_table"
  ```

  > **Note**
  > The PASSWORD salt value must be entered before the field in the HASH ( ) function can be extracted.

  The benefit of using a PASSWORD identifier number with HASH( ) is that you do not have to expose a clear text salt value.

  For more information, see "PASSWORD command" on page 1893.

Password method guidelines

The password method is intended for use in scripts that prompt for the password at the beginning of the script, or prior to the HASH( ) function appearing in the script.

The password method is not suitable for use in computed fields because PASSWORD assignments are deleted when you close Analytics.

In addition, computed fields that use a password-based salt value are automatically removed from views when you reopen Analytics. This removal is necessary to avoid the recalculation of hash values using the default salt value. The recalculated values would differ from the original hash values calculated with a user-supplied salt value.

HEX( ) function

Converts an ASCII string to a hexadecimal string.

Syntax

```
HEX(field)
```

Parameters

Name	Type	Description
field	character	The ASCII string to convert to a hexadecimal string.

Output

Character.

Examples

Basic examples

Returns "3132333435":

```
HEX("12345")
```

Returns the values in the **Count** field as hexadecimal strings:

```
HEX(Count)
```

Remarks

How it works

This function returns the hexadecimal string that is equivalent to the field value or expression you specify. You can use the function when you need to identify the exact contents of a field, including characters that cannot be displayed on screen, such as CR (carriage return), LF (line feed), and NUL (null).

Return value length

The return value is a string that is double the length of the *field* value. The digits 0 to 9 and the letters A to F (for the digits 10 to 15) represent the 16 hexadecimal values.

Use fields as input rather than expressions

In general, you should apply this function to fields rather than expressions because it displays a representation of the internal storage format of expressions, which is not meaningful in most instances.

HOUR( ) function

Extracts the hour from a specified time or datetime and returns it as a numeric value using the 24-hour clock.

Syntax

```
HOUR(time/datetime)
```

Parameters

Name	Type	Description
time/datetime	datetime	The field, expression, or literal value to extract the hour from.

Output

Numeric.

Examples

Basic examples

Returns 23:

```
HOUR(`t235959`)
```

```
HOUR(`20141231 235959`)
```

Returns the hour for each value in the **Call_start_time** field:

```
HOUR(Call_start_time)
```

Remarks

Parameter details

A field specified for *time/datetime* can use any time or datetime format, as long as the field definition correctly defines the format.

Specifying a literal time or datetime value

When specifying a literal time or datetime value for *time/datetime*, you are restricted to the formats in the table below, and you must enclose the value in backquotes – for example, `` `20141231 235959` ``.

Do not use any separators such as slashes (/) or colons (:) between the individual components of dates or times.

- **Time values** – you can use any of the time formats listed in the table below. You must use a separator before a standalone time value for the function to operate correctly. Valid separators are the letter 't', or the letter 'T'. You must specify times using the 24-hour clock. Offsets from Coordinated Universal Time (UTC) must be prefaced by a plus sign (+) or a minus sign (-).
- **Datetime values** – you can use any combination of the date, separator, and time formats listed in the table below. The date must precede the time, and you must use a separator between the two. Valid separators are a single blank space, the letter 't', or the letter 'T'.

Example formats	Example literal values
thhmmss	`t235959`
Thhmm	`T2359`
YYYYMMDD hhmmss	`20141231 235959`
YYMMDDthhmm	`141231t2359`
YYYYMMDDThh	`20141231T23`
YYYYMMDD hhmmss+/-hhmm (UTC offset)	`20141231 235959-0500`
YYMMDD hhmm+/-hh (UTC offset)	`141231 2359+01`
Note Do not use hh alone in the main time format with data that has a UTC offset. For example, avoid: hh+hhmm. Results can be unreliable.	

HTOU( ) function

Converts a hexadecimal string to a Unicode string. Abbreviation for "Hexadecimal to Unicode".

> **Note**
> This function is specific to the Unicode edition of Analytics. It is not a supported function in the non-Unicode edition.

Syntax

```
HTOU(hex_string)
```

Parameters

Name	Type	Description
hex_string	character	The hexadecimal string to convert to a Unicode string. The string can only contain hexadecimal values, for example "20AC".

Output

Character.

Examples

Basic examples

Returns "ABC123":

```
HTOU("004100420043003100320033")
```

Advanced examples

Adding a currency symbol to a value

You need to extract a monetary field to a new table. The field should display the numeric **Amount** field's value and prepend it with a Euro currency symbol (€):

```
EXTRACT HTOU("20AC") + STRING(Amount, 10) AS "Currency_Amount" TO Dis-
play_Table
```

When the EXTRACT command runs, HTOU( ) returns the Euro symbol "€" and concatenates it with the **Amount** value that STRING( ) converts to a character. If the original value of **Amount** was 2000, then the value in **Currency_Amount** is "€2000".

Remarks

Related functions

HTOU( ) is the inverse of the DHEX( ) function, which converts a Unicode string to a hexadecimal string.

INCLUDE( ) function

Returns a string that includes only the specified characters.

Syntax

```
INCLUDE(string, characters_to_include)
```

Parameters

Name	Type	Description
string	character	The field, expression, or literal value to restrict to included characters.
characters_to_include	character	The list of characters to include. If you specify double quotation marks in *characters_to_include*, you must enclose the list of characters in single quotation marks. For example: `'"-/'` **Note** If a character you specify to include does not appear in *string*, it is not included in the return value.

Output

Character.

Examples

Basic examples

Returns "123", which is the input string with only numbers included:

```
INCLUDE("123 Main St.", "0123456789")
```

Returns "1231234", which is the input string with only numbers included:

```
INCLUDE("123-123-4", "1243")
```

Returns "" (nothing), because the input string does not contain "D":

```
INCLUDE("ABC", "D")
```

Remarks

How it works

The INCLUDE( ) function compares each character in *string* with the characters listed in *characters_to_include*. If a match occurs, the character is included in the output string.

No matching characters

If there are no matches between *string* and *characters_to_include* the output of the function is blank.

Case sensitivity

The INCLUDE( ) function is case-sensitive. If you specify "ID" in *characters_to_include*, these characters are not included in "id#94022". If there is a chance the case may be mixed, use the UPPER( ) function to convert *string* to uppercase.

For example:

```
INCLUDE(UPPER("id#94022"), "ID0123456789")
```

Usage tip

Use INCLUDE( ) if the set of characters you want to include is small, and the set you want to exclude is large.

Related functions

The INCLUDE( ) function is the opposite of the EXCLUDE( ) function.

INSERT( ) function

Returns the original string with specified text inserted at a specific byte location.

Syntax

```
INSERT(string, insert_text, location)
```

Parameters

Name	Type	Description
string	character	The field, expression, or literal value to insert the text into.
insert_text	character	The text to insert.
location	numeric	The character position at which to insert *insert_text* into the *string*.

Output

Character.

Examples

Basic examples

Returns "aXXXbcde":

```
INSERT("abcde", "XXX", 2)
```

Returns "XXXabcde":

```
INSERT("abcde", "XXX", 0)
```

Returns "abcdeXXX", with `"XXX"` inserted at byte position 6 instead of 8, because `"abcde"` is only 5 bytes long::

```
INSERT("abcde", "XXX", 8)
```

Remarks

How it works

The INSERT( ) function inserts specified characters or spaces into a character string, beginning at a specified position in the string.

When to use INSERT( )

Use INSERT( ) to normalize data for formatting, for duplicate matching, and for the JOIN and DEFINE RELATION commands, which require identical fields.

For example, part numbers in one file may be in the format "12345", and in another file, "12-345." In the first file, you can use INSERT( ) to insert a hyphen (-) in position 3.

Location guidelines

- If the *location* value is greater than the length of *string*, the *insert_text* value is inserted at the end of the string.
- If *location* is 0 or 1, *insert_text* is inserted at the beginning of the string.

Inserting double quotation marks

If you specify double quotation marks in *insert_text*, you must enclose them in single quotation marks.

For example: `'"'`

INT( ) function

Returns the integer value of a numeric expression or field value.

Syntax

```
INT(number)
```

Parameters

Name	Type	Description
number	numeric	The field or numeric expression to convert to an integer. If the value specified includes decimals, the decimals are truncated without rounding.

Output

Numeric.

Examples

Basic examples

Returns 7:

```
INT(7.9)
```

Returns -7:

```
INT(-7.9)
```

IPMT( ) function

Returns the interest paid on a loan for a single period.

Syntax

```
IPMT(rate, specified_period, periods, amount <,type>)
```

Parameters

Name	Type	Description
rate	numeric	The interest rate per period.
specified_period	numeric	The period for which you want to find the interest payment.
periods	numeric	The total number of payment periods.
amount	numeric	The principal amount of the loan.
type optional	numeric	The timing of payments: ◦ `0` - payment at the end of a period ◦ `1` - payment at the beginning of a period If omitted, the default value of 0 is used.

> **Note**
> You must use consistent time periods when specifying *rate* and *periods* to ensure that you are specifying interest rate **per period**.
>
> For example:
>
> - for monthly payments on a two-year loan or investment with interest of 5% per annum, specify 0.05/12 for *rate* and 2 * 12 for *periods*
> - for annual payments on the same loan or investment, specify 0.05 for *rate* and 2 for *periods*

Output

Numeric.

Examples

Basic examples

Returns 1489.58, the interest paid in the first month of a twenty-five year, $275,000 loan at 6.5% per annum, with payments due at the end of the month:

```
IPMT(0.065/12, 1, 12*25, 275000, 0)
```

Returns 10.00, the interest paid on the same loan in the last month of the loan:

```
IPMT(0.065/12, 300, 12*25, 275000, 0)
```

Remarks

Related functions

The PPMT( ) function is the complement of the IPMT( ) function.

The CUMIPMT( ) function calculates interest paid during a range of periods.

ISBLANK( ) function

Returns a logical value indicating whether the input value is blank.

Syntax

```
ISBLANK(string)
```

Parameters

Name	Type	Description
string	character	The value to test for blank data.

Output

Logical. Returns **T** (true) if the *string* parameter value is blank, and **F** (false) otherwise.

Examples

Basic examples

Returns F:

```
ISBLANK("  A")
```

Returns T:

```
ISBLANK("  ")
```

```
ISBLANK("")
```

Returns T for all values in the **Address** field that are blank, and F otherwise:

```
ISBLANK(Address)
```

Remarks

When to use ISBLANK( )

Use ISBLANK( ) during the data integrity phase of an analysis project to identify fields with missing data, which may indicate issues with the source data.

What is blank input?

For the function to evaluate to true, the input value must be one of the following:

- entirely blank (that is, contain only spaces)
- a zero-length string

The function only identifies true blanks in source data, not invalid characters that appear as blanks in a view.

Null characters

ISBLANK( ) may not return useful results when used with character fields that contain null characters. Analytics uses the null character to indicate the end of a string, and for this reason the ISBLANK( ) function will not read any character data that follows a null character, including blanks.

ISDEFINED( ) function

Returns **T** (true) if the specified field or variable is defined, and **F** (false) otherwise.

Syntax

```
ISDEFINED(string)
```

Parameters

Name	Type	Description
string	character	The name of the field or variable to check for the existence of. The value must be entered as a quoted string: `ISDEFINED("v_numeric_limit")`

Output

Logical.

Examples

Basic examples

Returns **T** if `v_numeric_limit` is defined as a variable or field, otherwise returns **F**:

```
ISDEFINED("v_numeric_limit")
```

Advanced examples

Using ISDEFINED( ) to test a field

The following example uses the ISDEFINED( ) function to test if the **Limit** field is defined in the table before extracting records based on the value in the field:

```
OPEN Metaphor_Employees_US
IF ISDEFINED("Limit") EXTRACT RECORD IF Limit > 50000 TO "HighLimit.fil"
```

ISFUZZYDUP( ) function

Returns a logical value indicating whether a string is a fuzzy duplicate of a comparison string.

Syntax

```
ISFUZZYDUP(string1, string2, levdist <,diffpct>)
```

Parameters

Name	Type	Description
string1	character	The first string in the comparison.
string2	character	The second string in the comparison.
levdist	numeric	The maximum allowable Levenshtein distance between the two strings for them to be identified as fuzzy duplicates. The *levdist* value cannot be less than 1 or greater than 10. Increasing the *levdist* value increases the number of results by including values with a greater degree of fuzziness - that is, values that are more different from each other.
diffpct optional	numeric	The upper threshold for the 'difference percentage'. Difference percentage is explained in "How it works" on page 2201. The *diffpct* value cannot be less than 1 or greater than 99. Increasing the *diffpct* value increases the number of results by including values with a greater proportion of difference relative to their length. If omitted, difference percentage is not considered during processing of the ISFUZZYDUP( ) function.

Output

Logical. Returns **T** (true) if *string* values are fuzzy duplicates, and **F** (false) otherwise.

Examples

Basic examples

Returns F, because two edits are required to transform "Smith" into "Smythe", but the *levdist* value is only 1:

```
ISFUZZYDUP("Smith","Smythe", 1, 99)
```

Returns T, because two edits are required to transform "Smith" into "Smythe", and the *levdist* value is 2:

```
ISFUZZYDUP("Smith","Smythe", 2, 99)
```

Returns T, because zero edits are required to transform "SMITH" into "smith", and the *levdist* value is 1 (the ISFUZZYDUP( ) function is not case-sensitive):

```
ISFUZZYDUP("SMITH","smith", 1, 99)
```

Returns a logical value (T or F) indicating whether individual values in the **Last_Name** field are fuzzy duplicates for the string "Smith":

```
ISFUZZYDUP(Last_Name,"Smith", 3, 99)
```

Advanced examples

Working with difference percentage

The difference percentage gives you a tool for reducing the number of false positives returned by ISFUZZYDUP( ).

No *diffpct* specified

Returns T, because five edits are required to transform "abc" into "Smith", and the *levdist* value is 5:

```
ISFUZZYDUP("abc", "Smith", 5)
```

***diffpct* specified**

Returns F, even though "abc" is within the specified Levenshtein distance of "Smith", because *5 edits/a string length of 3* results in a difference percentage of 167%, which exceeds the specified *diffpct* of 99%:

```
ISFUZZYDUP("abc", "Smith", 5, 99)
```

Difference percentage is fully explained in "How it works" on the facing page.

Isolating fuzzy duplicates for "Smith"

Create a filter that isolates all values in the **Last_Name** field that are fuzzy duplicates for "Smith":

```
SET FILTER TO ISFUZZYDUP(Last_Name, "Smith", 3, 99)
```

Changing the *levdist* or *diffpct* values allows you to adjust the amount of difference in the filtered values.

Isolating fuzzy duplicates for a vendor name

Create a filter that isolates all values in the **Vendor_Name** field that are fuzzy duplicates for "Pacific Lighting and Electrical Supply, Inc.":

```
SET FILTER TO ISFUZZYDUP(Vendor_Name, "Pacific Lighting and Electrical
Supply, Inc.", 2, 99)
```

Improve the effectiveness of the filter by using additional functions with the ISFUZZYDUP( ) function.

Using ISFUZZYDUP( ) with OMIT( ) returns:

- Pacific Lighting and Electrical Supply, Inc.
- Pacific Lighting and Electrical Supply
- Pacific Lighting & Electrical Supply, Inc.

```
SET FILTER TO ISFUZZYDUP(OMIT(Vendor_Name, ".,&,and,Inc,Ltd,"), "Pacific
Lighting Electrical Supply", 2, 99)
```

Using ISFUZZYDUP( ) with SORTWORDS( ) and UPPER( ) returns:

- Pacific Lighting and Electrical Supply, Inc.
- Pacific Electrical and Lighting Supply, Inc.

```
SET FILTER TO ISFUZZYDUP(SORTWORDS(UPPER(Vendor_Name)), SORTWORDS(UPPER
("Pacific Lighting and Electrical Supply, Inc.")), 2, 99)
```

Using ISFUZZYDUP( ) with SORTWORDS( ), UPPER( ), and OMIT( ) returns:

- Pacific Lighting and Electrical Supply, Inc.
- Pacific Lighting and Electrical Supply
- Pacific Lighting & Electrical Supply, Inc.
- Pacific Electrical and Lighting Supply, Inc.

```
SET FILTER TO ISFUZZYDUP(SORTWORDS(UPPER(OMIT(Vendor_Name,
".,&,and,Inc,Ltd,"))), SORTWORDS(UPPER("Pacific Lighting Electrical Sup-
ply")), 2, 99)
```

Remarks

When to use ISFUZZYDUP( )

Use the ISFUZZYDUP( ) function to find nearly identical values (fuzzy duplicates) or locate inconsistent spelling in manually entered data.

How it works

The ISFUZZYDUP( ) function calculates the Levenshtein distance between two strings, and calculates the difference percentage.

ISFUZZYDUP( ) evaluates to T (true) if:

- The Levenshtein distance is less than or equal to the *levdist* value.
- The difference percentage is less than or equal to the *diffpct* value (if specified).

Levenshtein distance

The Levenshtein distance is a value representing the minimum number of single character edits required to make one string identical to the other string.

For more information, see "LEVDIST( ) function" on page 2214.

Difference percentage

The difference percentage is the percentage of the shorter of the two evaluated strings that is different.

The difference percentage is the result of the following internal Analytics calculation, which uses the Levenshtein distance between the two strings:

Levenshtein distance / number of characters in the shorter string × 100 = difference percentage

Using the optional difference percentage helps reduce the number of false positives returned by ISFUZZYDUP( ):

- The upper threshold for *diffpct* is 99%, which prevents the entire replacement of a string in order to make it identical.
- Strings that require a large number of edits in relation to their length are excluded.

Usage tips

- **Case-sensitivity** – The function is not case-sensitive, so "SMITH" is equivalent to "smith."
- **Trailing blanks** – The function automatically trims trailing blanks in fields, so there is no need to use the TRIM( ) function when specifying a field as a parameter.
- **Sorting elements** – The SORTWORDS( ) function can improve the effectiveness of the ISFUZZYDUP( ) function by sorting individual elements in field values into a sequential order.

  Sorting elements, such as the components of an address, can make two strings with the same information, but a different format, more closely resemble each other. A closer resemblance improves the chances that a pair of strings are selected as fuzzy duplicates of each other.
- **Removing generic elements** – The OMIT( ) and EXCLUDE( ) functions can improve the effectiveness of the ISFUZZYDUP( ) function by removing generic elements such as "Corporation" or "Inc.", or characters such as commas, periods, and ampersands (&), from field values.

  Removal of generic elements and punctuation focuses the ISFUZZYDUP( ) string comparison on just the portion of the strings where a meaningful difference may occur.

How the FUZZYDUP command and the ISFUZZYDUP( ) function differ

The FUZZYDUP command identifies all fuzzy duplicates in a field, organizes them in non-exhaustive groups, and outputs results that in total are exhaustive.

The ISFUZZYDUP( ) function generates a single, exhaustive list of fuzzy duplicates for a specific character value.

The command and the function both identify exact duplicates. Unlike the command, you cannot exclude exact duplicates when using the function.

What exhaustive means

Exhaustive means that all values within the specified degree of difference of the test value are returned, regardless of their position in the test field relative to the test value.

The ISFUZZYDUP( ) function is useful if the non-exhaustive groups produced by the FUZZYDUP command are not convenient for the purposes of your analysis, and you need to directly scrutinize every fuzzy duplicate for a specific character value.

Related functions

- **LEVDIST( )** – provides an alternate method for comparing strings based on Levenshtein distance.

  Unlike ISFUZZYDUP( ), LEVDIST( ) is case-sensitive by default.
- **DICECOEFFICIENT( )** – de-emphasizes or completely ignores the relative position of characters or character blocks when comparing strings.
- **SOUNDSLIKE( ) and SOUNDEX( )** – compare strings based on a phonetic comparison (sound) rather than on an orthographic comparison (spelling).

LAST( ) function

Returns a specified number of characters from the end of a string.

Syntax

```
LAST(string, length)
```

Parameters

Name	Type	Description
string	character	The field, expression, or literal value to return the characters from.
length	numeric	The number of characters to return.

Output

Character.

Examples

Basic examples

Returns "Savings":

```
LAST("Account Type: Savings", 7)
```

Returns "efghi":

```
LAST("abcdefghi", 5)
```

Returns "fghi ":

```
LAST("abcdefghi ", 5)
```

Returns " abc", because the *string* value is shorter than the specified length of 6, so leading spaces are added to the output:

```
LAST("abc", 6)
```

Remarks

Blank results caused by trailing spaces

Trailing spaces in *string* can cause the results produced by the LAST( ) function to be blank.

For example, the output for `LAST("6483-30384   ", 3)` is " ".

You can use the ALLTRIM( ) function in conjunction with LAST( ) to remove any trailing spaces in *string*.

For example, `LAST(ALLTRIM("6483-30384   "), 3)` returns "384".

Returning characters from the start of a string

If you want to return a specified number of characters from the start of a string, use the SUBSTR( ) function. For more information, see "SUBSTR( ) function" on page 2411.

LEADING( ) function

Returns a string containing a specified number of leading digits.

Syntax

```
LEADING(number, leading_digits)
```

Parameters

Name	Type	Description
number	numeric	The value to return the leading digits from.
leading_digits	numeric	The number of leading digits to be returned.

Output

Character.

Examples

Basic examples

Literal numeric input

Returns 623:

```
LEADING(6234.56, 3)
```

Returns 62345:

```
LEADING(-6234.56, 5)
```

Padding with trailing zeros

Returns 000:

```
LEADING(0.00, 3)
```

Returns 00000:

```
LEADING(0.00, 5)
```

Returns 35500:

```
LEADING(3.55, 5)
```

Remarks

Use LEADING( ) to extract digits from a numeric field as a string, and filter out non-digit elements such as decimals or dollar signs.

LEADINGZEROS( ) function

Adds leading zeros to a character string or a number.

Syntax

```
LEADINGZEROS(string/number, length)
```

Parameters

Name	Type	Description
string/number	character numeric	The field, expression, or literal value to add leading zeros to. Leading and trailing spaces are automatically trimmed from character values, and from the result of character expressions, before the leading zeros are added.
length	numeric	The length of the output string. **Note** Any value in *string/number* longer than *length* is truncated from the left.

Output

Character.

Examples

Basic examples

Character input

Returns "000235", because *length* is greater than the number of characters in *string/number* so three leading zeros are added to the result:

```
LEADINGZEROS("235", 6)
```

Returns "35", because *length* is less than the number of characters in *string/number* so the result is truncated from the left:

```
LEADINGZEROS("235", 2)
```

Integer input

Returns "235":

```
LEADINGZEROS(235, 3)
```

Returns "00235", because *length* is greater than the number of digits in *string/number* so two leading zeros are added to the result:

```
LEADINGZEROS(235, 5)
```

Decimal input

Returns "023585", because LEADINGZEROS( ) removes the decimal point:

```
LEADINGZEROS(235.85, 6)
```

Negative input

Returns "0644894", because LEADINGZEROS( ) removes the negative sign:

```
LEADINGZEROS(-6448.94, 7)
```

Advanced examples

Adding leading zeros to a character field containing numbers

The `Employee_Number` field contains the value `"254879"`. You need to convert the value to a 10-digit string with leading zeros.

```
COMMENT returns "0000254879"
ASSIGN v_str_length = 10
LEADINGZEROS(Employee_Number, v_str_length)
```

Harmonizing a key field when relating tables

You have two tables, **Ar** and **Customer**, and you plan to relate them on the customer number key field for further analysis. However, the two key fields have different data formats, which prevents values from matching:

Table	Key field	Data type	Field length	Example
Ar	CustNum	Numeric	4	4455
Customer	CustNo	Character	6 (pads numbers with leading zeros)	"004455"

To harmonize the fields when relating, you create a computed field in the **Ar** table that uses the LEADINGZEROS( ) function. You then relate using the computed field:

```
OPEN Customer
INDEX ON CustNo TO Customer_on_CustNo
OPEN Ar
COMMENT Create the computed field CustNum_Zeros that converts the
CustNum field to the character data type and adds leading zeros.
DEFINE FIELD CustNum_Zeros COMPUTED LEADINGZEROS(CustNum,6)
COMMENT Relate the Ar table using the newly created CustNum_Zeros com-
puted field.
DEFINE RELATION CustNum_Zeros WITH Customer INDEX Customer_on_CustNo
```

Remarks

How it works

This function adds leading zeros to the output if the output length you specify is greater than the length of an input value. The function accepts various kinds of character and numeric input and outputs a character string. The function works the same way in the Unicode and non-Unicode editions of Analytics.

The function is commonly used to normalize fields that require leading zeros, for example, check number, purchase order number, and invoice number fields.

Input length and output values

Input length	Output values
string/number less than *length*	leading zeros added
string/number equal to *length*	no zeros added
string/number greater than *length*	values truncated from the left

Negative sign, thousands separator, and decimal point

Input values	Result
Numeric	LEADINGZEROS( ) removes the negative sign, the thousands separator, and the decimal point from numeric input values. The removed symbols are not included in the length of the input value.
Character	LEADINGZEROS( ) does not remove the negative sign, the thousands separator, or the decimal point from character input values. The removed symbols are included in the length of the input value.

LENGTH( ) function

Returns the number of characters in a string.

Syntax

```
LENGTH(string)
```

Parameters

Name	Type	Description
string	character	The field, expression, or literal value to find the length of.

Output

Numeric.

Examples

Basic examples

Returns 15:

```
LENGTH("ABC Corporation")
```

Returns the length in characters of the **Description** field in the table layout:

```
LENGTH(Description)
```

Advanced examples

Displaying the length of each address in an address field

Create a computed field that displays the length in characters of each address in the **Vendor_Street** field. Leading and trailing blank spaces are first trimmed from the address values so they are not counted in the length.

```
DEFINE FIELD Address_Length COMPUTED LENGTH(ALLTRIM(Vendor_Street))
```

Remarks

How it works

The LENGTH( ) function counts the number of characters in *string*, including any spaces, and returns the number.

Trailing spaces

Trailing spaces are counted as characters. If you do not want trailing spaces to be counted, use the TRIM( ) or ALLTRIM( ) functions to remove them. For example:

```
LENGTH(TRIM(Vendor_Street))
```

If you create a computed field to display the length of the values in a field, and you do not remove trailing spaces, the maximum length of the field is displayed for each value.

LEVDIST( ) function

Returns the Levenshtein distance between two specified strings, which is a measurement of how much the two strings differ.

Syntax

```
LEVDIST(string1, string2 <,case_sensitive>)
```

Parameters

Name	Type	Description
string1	character	The first string in the comparison.
string2	character	The second string in the comparison.
case_sensitive optional	logical	Specify `T` for a case-sensitive comparison of strings, or `F` to ignore case. If omitted, the default value of T is used.

Output

Numeric. The value is the Levenshtein distance between two strings.

Examples

Basic examples

Returns 3, because two substitutions and one insertion are required to transform "smith" into "Smythe":

```
LEVDIST("smith","Smythe")
```

Returns 2, because case is ignored, so only two substitutions are required to transform "smith's" into "Smythes":

```
LEVDIST("smith's","Smythes",F)
```

Returns the Levenshtein distance between each value in the **Last_Name** field and the string "Smith":

```
LEVDIST(TRIM(Last_Name),"Smith")
```

Advanced examples

Ranking values against "Smith"

Create the computed field **Lev_Dist** to display the Levenshtein distance between "Smith" and each value in the **Last_Name** field:

```
DEFINE FIELD Lev_Dist COMPUTED LEVDIST(TRIM(Last_Name),"Smith", F)
```

Add the computed field **Lev_Dist** to the view, and then quick sort it in ascending order, to rank all values in the **Last_Name** field by their amount of difference from "Smith".

Isolating fuzzy duplicates for "Smith"

Create a filter that isolates all values in the **Last_Name** field that are within a specified Levenshtein distance of "Smith":

```
SET FILTER TO LEVDIST(TRIM(Last_Name),"Smith", F) < 3
```

Changing the number in the expression allows you to adjust the amount of Levenshtein distance in the filtered values.

Remarks

When to use LEVDIST( )

Use the LEVDIST( ) function to find nearly identical values (fuzzy duplicates) or locate inconsistent spelling in manually entered data. LEVDIST( ) also identifies exact duplicates.

How it works

The LEVDIST( ) function returns the Levenshtein distance between the two evaluated strings, which is a value representing the minimum number of single character edits required to make one string identical to the other string.

Each required edit increments the value of the Levenshtein distance by 1. The greater the Levenshtein distance, the greater the difference between the two strings. A distance of zero (0) means the strings are identical.

Types of edits

The edits can be of three types:

- insertion
- deletion
- substitution

Transpositions (two adjacent letters reversed) are not recognized by the Levenshtein algorithm, and count as two edits – specifically, two substitutions.

Non-alphanumeric characters

Punctuation marks, special characters, and blanks are treated as single characters, just like letters and numbers.

The case of characters

Changing the case of a character counts as one substitution, unless you turn off case sensitivity using the *case_sensitive* setting.

The position of characters

Levenshtein distance takes the position of characters into account. The same characters ordered differently may result in a different Levenshtein distance.

Returns 2:

```
LEVDIST("abc", "dec")
```

Returns 3:

```
LEVDIST("abc", "cde")
```

Using TRIM( ) with LEVDIST( )

To ensure accurate results when using LEVDIST( ) to compare a literal string such as "Smith" with a character field, you must use the TRIM( ) function to remove trailing blanks from the field.

If you are comparing two fields, you must use the TRIM( ) function with each field.

The Levenshtein algorithm counts blanks as characters, so any trailing blanks are included in the calculation of the number of edits required to make two strings identical.

Related functions

- **ISFUZZYDUP( )** – provides an alternate method for comparing strings based on Levenshtein distance.

  Unlike the default behavior of LEVDIST( ), ISFUZZYDUP( ) is not case-sensitive.
- **DICECOEFFICIENT( )** – de-emphasizes or completely ignores the relative position of characters or character blocks when comparing strings.
- **SOUNDSLIKE( ) and SOUNDEX( )** – compare strings based on a phonetic comparison (sound) rather than on an orthographic comparison (spelling).

LOG( ) function

Returns the logarithm (base 10) of a numeric expression or field value with a specified number of decimal places.

Syntax

```
LOG(number, decimals)
```

Parameters

Name	Type	Description
number	numeric	The value to find the logarithm for.
decimals	numeric	The number of decimal places for the return value.

Output

Numeric.

Examples

Basic examples

Returns 3.0000:

```
LOG(1000, 4)
```

Returns 4.86:

```
LOG(72443, 2)
```

Advanced examples

Finding the cube root

Creates a field that is the cube root of the field X to two decimal places:

```
DEFINE FIELD Cube_root COMPUTED EXP(LOG(X, 6) / 3, 2)
```

> **Note**
> You determine the *n*th root by dividing the log of the value by *n* and taking the exponential value of the result.

Remarks

How it works

The logarithm of a number is the exponent (or power) of 10 needed to generate that number. Therefore, the logarithm of 1000 is 3.

Related functions

The LOG( ) function is the inverse of the EXP( ) function.

LOWER( ) function

Returns a string with alphabetic characters converted to lowercase.

Syntax

```
LOWER(string)
```

Parameters

Name	Type	Description
string	character	The field, expression, or literal value to convert to lowercase.

Output

Character.

Examples

Basic examples

Returns "abc":

```
LOWER("ABC")
```

Returns "abc 123 def":

```
LOWER("abc 123 DEF")
```

Returns "abcd 12":

```
LOWER("AbCd 12")
```

Returns all the values in the **Last_Name** field converted to lowercase:

```
LOWER(Last_Name)
```

Remarks

How it works

The LOWER( ) function converts all alphabetic characters in *string* to lowercase. All non-alphabetic characters are left unchanged.

When to use LOWER( )

Use LOWER( ) when you search for data that is in mixed or unknown case, or when you want data formatted as lowercase text.

LTRIM( ) function

Returns a string with leading spaces removed from the input string.

Syntax

```
LTRIM(string)
```

Parameters

Name	Type	Description
string	character	The field, expression, or literal value to remove leading spaces from.

Output

Character.

Examples

Basic examples

Note that in both examples the trailing spaces are not removed by the LTRIM( ) function.

Returns "Vancouver ":

```
LTRIM("   Vancouver   ")
```

Returns "New York ":

```
LTRIM("   New York   ")
```

Advanced examples

Removing non-breaking spaces

Non-breaking spaces are not removed by the LTRIM( ) function.

If you need to remove leading non-breaking spaces, create a computed field using the following expression:

```
DEFINE FIELD Description_cleaned COMPUTED LTRIM(REPLACE(Description, CHR
(160), CHR(32)))
```

The REPLACE( ) function replaces any non-breaking spaces with regular spaces, and then LTRIM( ) removes any leading regular spaces.

Remarks

How it works

The LTRIM( ) function removes leading spaces only. Spaces inside the string, and trailing spaces, are not removed.

Related functions

LTRIM( ) is related to the TRIM( ) function, which removes any trailing spaces from a string, and to the ALLTRIM( ) function, which removes both leading and trailing spaces.

MAP( ) function

Returns a logical value indicating if a character string matches a specified format string containing wildcard characters, literal characters, or both.

Syntax

```
MAP(string, format)
```

Parameters

Name	Type	Description
string	character	The field, expression, or literal value to test for matches.
format	character	The data pattern, or character string, you want to compare with *string*. *format* can contain wildcard characters, literal characters, or a combination of the two: `"\9\9\9-999-9999"` The following wildcard characters are supported: ◦ `"X"` - matches any alphabetic character (a-z, A-Z, European characters). This wildcard character is not case-sensitive. You can use "X" or "x" ◦ `"9"` - matches any number (0-9) ◦ `"!"` - matches any non-blank character ◦ `"?"` - matches any character, including blanks ◦ `"\"` - escape character that specifies that the character immediately following is a literal. Use the escape character if you want to literally match any of the wildcard characters (X, x, 9, !, ?) ◦ `"\\"` - specifies a literal backslash

Output

Logical. Returns **T** (true) if a match is found, and **F** (false) otherwise.

Examples

Basic examples

Simple search patterns

Returns T:

```
MAP("ABC Plumbing", "xxx")
```

Returns F (*string* has only 3 numbers where a minimum of 4 are required):

```
MAP("045", "9999")
```

Escaping a wildcard

If the goal is to return T for only those values that start with the literal character "X", followed by any second letter, the *format* parameter "\XX" ensures that the first "X" in the parameter is interpreted literally and not as a wildcard.

Returns T:

```
MAP("XA-123", "XX")
```

```
MAP("GC-123", "XX")
```

```
MAP("XA-123", "\XX")
```

Returns F:

```
MAP("GC-123", "\XX")
```

Fields and patterns

Returns T for all records with invoice numbers that consist of, or that start with, two letters followed by five numbers. Returns F otherwise:

```
MAP(Invoice_Number, "XX99999")
```

Returns T for all records with invoice numbers that are exactly "AB12345", or that start with "AB12345". Returns F otherwise:

```
MAP(Invoice_Number, "AB12345")
```

Returns T for all records with invoice numbers that consist of, or that start with, "AB" followed by five numbers. Returns F otherwise:

```
MAP(Invoice_Number, "AB99999")
```

Returns T for all records that do not match the standard format of social security numbers in the SSN field. Returns F otherwise:

```
NOT MAP(SSN, "999-99-9999")
```

Advanced examples

Extracting records with 10-character product codes and with the leading characters "859-"

Use an IF statement and the MAP( ) function to extract only those records that have product codes at least 10 characters long, and the leading characters "859-":

```
EXTRACT RECORD IF MAP(Product_Code, "85\9-999999") TO "Long_Codes_859"
```

Remarks

When to use MAP( )

Use the MAP( ) function to search for patterns or particular formats in alpha-numeric data. The patterns or formats can be made up of wildcard characters, literal characters, or a combination of both.

Case sensitivity

The MAP( ) function is case-sensitive when comparing two literal characters. For example, "a" is not equivalent to "A".

If *string* includes character data with inconsistent case, you can use the UPPER( ) function to convert the values to consistent case before using MAP( ).

For example:

```
MAP(UPPER(Invoice_Number), "AB99999")
```

Partial matching

MAP( ) supports partial matching in one situation but not in the other.

Partial matching in MAP( ) is not affected by the **Exact Character Comparisons** option (SET EXACT ON/OFF).

Partial matching supported

Partial matching is supported if the *format* value is shorter than the *string* value.

Returns T, because *format* is 7 characters and *string* is 9 characters:

```
MAP("AB1234567", "AB99999")
```

> **Note**
> To return True, the *format* value must appear at the start of the *string* value.

Partial matching not supported

Partial matching is not supported if the *format* value is longer than the *string* value.

Returns F, because *format* is 7 characters and *string* is 6 characters:

```
MAP("AB1234", "AB99999")
```

If *format* is longer than *string*, the result is always False.

Accounting for blank spaces

Blank spaces are treated as characters, and can be accounted for in either of two ways:

- match blanks literally, by including blanks in the *format* value at the appropriate position
- use the wildcard `"?"`, which matches any character, including blanks

If required, you can use the TRIM( ), LTRIM( ), or ALLTRIM( ) functions to remove leading or trailing blanks from *string*, which ensures that only text characters and any internal blanks are compared.

Concatenating fields

You can concatenate two or more fields in *string* if you want to search in more than one field in a table. The concatenated fields are treated like a single field that includes leading and trailing blanks from the individual fields, unless you use the ALLTRIM( ) function to remove them.

MASK( ) function

Performs a bitwise AND operation on the first bytes of two character strings.

Syntax

```
MASK(character_value, character_mask)
```

Parameters

Name	Type	Description
character_value	character	The string with the byte to test.
character_mask	character	The string with the byte to test against (the mask value).

Output

Character. The output is the character representation of the binary result of the bitwise AND operation.

Examples

Basic examples

Returns "2" (00110010), the result of a bitwise AND of 3 (00110011) and 6 (00110110):

```
MASK("3", "6")
```

Remarks

When to use MASK( )

Use MASK( ) to identify specific bit patterns in a byte of data, including whether or not a particular bit is set to 1.

How it works

The MASK( ) function performs a bitwise AND operation on the binary representations of the first characters of *character_value* and *character_mask*. The two comparison bytes are compared one bit at a time, resulting in a third binary value.

The result of each comparison of corresponding bits is either 1 or 0:

character_value bit	*character_mask* bit	Result
0	0	0
0	1	0
1	0	0
1	1	1

Comparison strings longer than one byte

If either comparison string is longer than one byte, subsequent characters are ignored.

MATCH( ) function

Returns a logical value indicating whether the specified value matches any of the values it is compared against.

Syntax

```
MATCH(comparison_value, test <,...n>)
```

Parameters

Name	Type	Description
comparison_value	character numeric datetime	The field, expression, or literal value to test for matches.
test <,...n>	character numeric datetime	Any field, expression, or literal value you want to compare with *comparison_value*. You can specify as many test values as necessary, but all specified values must be of the same data type: `MATCH(comparison_value, `20140930`, `20141030`)`

> **Note**
> Inputs to the MATCH( ) function can be character, numeric, or datetime data. You cannot mix data types. All inputs must belong to the same data category.

Output

Logical. Returns **T** (true) if at least one match is found, and **F** (false) otherwise.

Examples

Basic examples

> **Note**
> Return values for character comparisons assume that SET EXACT is OFF (the default setting), except where noted.

Testing literal values

Returns T:

```
MATCH("ABC", "BCD", "CDE", "AB")
```

Returns F:

```
MATCH(98, 99, 100, 101)
```

Testing a field

Returns T for all records that contain "Phoenix", "Austin", or "Los Angeles" in the **Vendor_City** field. Returns F otherwise:

```
MATCH(Vendor_City, "Phoenix", "Austin", "Los Angeles")
```

Returns T for all records that do not contain "Phoenix", "Austin", or "Los Angeles" in the **Vendor_City** field. Returns F otherwise:

```
NOT MATCH(Vendor_City, "Phoenix", "Austin", "Los Angeles")
```

Returns T for all records that contain "PHOENIX", "AUSTIN", or "LOS ANGELES" in the **Vendor_City** field, regardless of the case of any of the characters in the field. Returns F otherwise:

Values in the **Vendor_City** field are converted to uppercase before being compared with the uppercase city names.

```
MATCH(UPPER(Vendor_City), "PHOENIX", "AUSTIN", "LOS ANGELES")
```

Testing multiple fields

Returns T for all records that contain "Phoenix" in the **Vendor_City**, **City**, or **City_2** fields. Returns F otherwise:

```
MATCH("Phoenix", Vendor_City, City, City_2)
```

SET EXACT behavior

Returns T for all records that have product codes "A", "D", or "F", or product codes beginning with "A", "D", or "F", in the **Product_Code** field. Returns F otherwise:

```
MATCH(Product_Code, "A", "D", "F")
```

Returns T for all records that have one-character product codes "A", "D", or "F" in the **Product_Code** field. Returns F otherwise (SET EXACT must be ON):

```
MATCH(Product_Code, "A", "D", "F")
```

Comparing two fields

Returns T for all records that contain identical vendor and employee addresses. Returns F otherwise:

You may need to use additional functions to standardize the format of vendor and employee addresses.

```
MATCH(Vendor_Address, Employee_Address)
```

Comparing dates

Returns T for all records with an invoice date of 30 Sep 2014 or 30 Oct 2014. Returns F otherwise:

```
MATCH(Invoice_Date, `20140930`, `20141030`)
```

Advanced examples

Extracting anomalous inventory records

Use an IF statement and the MATCH( ) function to extract records that contain different amounts in the **Inventory_Value_at_Cost** field and the computed **Cost_x_Quantity** field:

```
EXTRACT RECORD IF NOT MATCH(Inventory_Value_at_Cost, Cost_x_Quantity) TO
"Non_matching_amounts"
```

Extracting records for departments 101, 103, and 107

Use an IF statement and the MATCH( ) function to extract only records associated with departments 101, 103, or 107:

```
EXTRACT RECORD IF MATCH(Dept, "101", "103", "107") TO "Three_Depart-
ments"
```

Remarks

Use MATCH( ) instead of the OR operator

You can use the MATCH( ) function instead of expressions that use the OR operator.

For example:

```
MATCH(City, "Phoenix", "Austin", "Los Angeles")
```

is equivalent to

```
City="Phoenix" OR City="Austin" OR City="Los Angeles"
```

Decimal precision of numeric inputs

When the numeric inputs being compared have a different decimal precision, the comparison uses the higher level of precision.

Returns T, because 1.23 is equal to 1.23:

```
MATCH(1.23, 1.23, 1.25)
```

Returns F, because 1.23 does not equal 1.234 once the third decimal place is considered:

```
MATCH(1.23, 1.234, 1.25)
```

Character parameters

Case sensitivity

The MATCH( ) function is case-sensitive when used with character data. When it compares characters, "a" is not equivalent to "A".

Returns F:

```
MATCH("a","A","B","C")
```

If you are working with data that includes inconsistent case, you can use the UPPER( ) function to convert the values to consistent case before using MATCH( ).

Returns T:

```
MATCH(UPPER("a"), UPPER("A"), UPPER("B"),  UPPER("C"))
```

Partial matching

Partial matching is supported for character comparisons. Either value being compared can be contained by the other value and be considered a match.

Both of these examples return T:

```
MATCH("AB", "ABC")
```

```
MATCH("ABC", "AB")
```

> **Note**
> The shorter value must appear at the start of the longer value to constitute a match.

Partial matching and SET EXACT

Partial matching is enabled when SET EXACT = OFF, which is the Analytics default setting. If SET EXACT = ON, partial matching is disabled and the comparison values must exactly match to constitute a match.

Both examples above are False when SET EXACT is ON.

For more information about SET EXACT (the **Exact Character Comparisons** option), see "SET command" on page 1960.

Turning SET EXACT off or on

If you want to ensure that the **Exact Character Comparisons** option is not used with the MATCH( ) function, check that the option is deselected in the **Table** tab in the **Options** dialog box (**Tools > Options**).

If you are using a script, you can add the `SET EXACT OFF` command before the MATCH( ) function appears. If required, you can restore the previous state with the `SET EXACT ON` command.

Datetime parameters

A date, datetime, or time field specified as a function input can use any date, datetime, or time format, as long as the field definition correctly defines the format.

Mixing date, datetime, and time inputs

You are not prevented from mixing date, datetime, and time values in the MATCH( ) function's inputs, but mixing these Datetime subtypes can give results that are not meaningful.

Analytics uses serial number equivalents to process datetime calculations, so even if you are interested in only the date portion of a datetime value, the time portion still forms part of the calculation.

Consider the following examples:

Returns T, because 31 December 2014 matches the second *test* value:

```
MATCH(`20141231`,`20141229`,`20141231`)
```

Returns F, even though the *comparison_value* and the second *test* value have an identical date of 31 December 2014:

```
MATCH(`20141231 120000`,`20141229`,`20141231`)
```

If we look at the serial number equivalent of these two expressions, we can see why the second one evaluates to false.

Returns T, because the serial number *comparison_value* is equal to the second serial number *test*:

```
MATCH(42003.000000, 42001.000000, 42003.000000)
```

Returns F, because the serial number *comparison_value* does not equal any of the *test* values:

```
MATCH(42003.500000, 42001.000000, 42003.000000)
```

The date portion of the serial numbers `42003.500000` and `42003.000000` match, but the time portions do not. `0.500000` is the serial number equivalent of 12:00 PM.

Harmonize Datetime subtypes

To avoid the problems that can be caused by mixing Datetime subtypes, you can use functions to harmonize the subtypes.

For example, this expression, which uses the same initial values as the second example above, returns T rather than F:

```
MATCH(CTOD(DATE(`20141231 120000`,"YYYYMMDD"),"YYYYMMDD"),`20141229`,
`20141231`)
```

Specifying a literal date, datetime, or time value

When specifying a literal date, datetime, or time value for any of the function inputs, you are restricted to the formats in the table below, and you must enclose the value in backquotes – for example, `` `20141231` ``.

Do not use any separators such as slashes (/) or colons (:) between the individual components of dates or times.

- **Datetime values** – you can use any combination of the date, separator, and time formats listed in the table below. The date must precede the time, and you must use a separator between the two. Valid separators are a single blank space, the letter 't', or the letter 'T'.

- **Time values** – you must specify times using the 24-hour clock. Offsets from Coordinated Universal Time (UTC) must be prefaced by a plus sign (+) or a minus sign (-).

Example formats	Example literal values
YYYYMMDD	`20141231`
YYMMDD	`141231`
YYYYMMDD hhmmss	`20141231 235959`
YYMMDDthhmm	`141231t2359`
YYYYMMDDThh	`20141231T23`
YYYYMMDD hhmmss+/-hhmm (UTC offset)	`20141231 235959-0500`
YYMMDD hhmm+/-hh (UTC offset)	`141231 2359+01`
thhmmss	`t235959`
Thhmm	`T2359`
Note Do not use hh alone in the main time format with data that has a UTC offset. For example, avoid: hh+hhmm. Results can be unreliable.	

MAXIMUM( ) function

Returns the maximum value in a set of numeric values, or the most recent value in a set of datetime values.

Syntax

```
MAXIMUM(value_1, value_2 <,...n>)
```

Parameters

Name	Type	Description
value_1, *value_2 <, ...n>*	numeric datetime	The values to compare, separated by commas. All values must be of the same data type. Additionally, datetime values must be of the same subtype. You cannot mix date, datetime, or time values in a single execution of the function.

Output

Numeric or Datetime.

Examples

Basic examples

Literal numeric input

Returns 7:

```
MAXIMUM(4, 7)
```

Returns 8:

```
MAXIMUM(4, 7, 3, 8)
```

Returns 8.00:

```
MAXIMUM(4, 7.25, 3, 8)
```

Literal datetime input

Returns `20161231`:

```
MAXIMUM(`20161231`, `20161229`, `20161230`)
```

Returns `20161231 23:59:59`:

```
MAXIMUM(`20161231 235959`, `20161229 235959`)
```

Returns `23:59:59`:

```
MAXIMUM(`.235957`, `.235959`, `.235958`)
```

Field input

Returns the most recent date among the three fields for each record:

```
MAXIMUM(PO_Date, Invoice_Date, Payment_Date)
```

Advanced examples

Creating a computed field with a minimum default value

If you have a table of overdue accounts, create an **Interest_Due** computed field with a minimum default value of $1.00:

```
DEFINE FIELD Interest_Due COMPUTED MAXIMUM(BALANCE * ANNUAL_RATE, 1)
```

If the balance multiplied by the interest rate is less than one dollar, MAXIMUM( ) returns 1. Otherwise, MAXIMUM( ) returns the calculated interest amount.

Discovering dates past the end of a quarter

To discover if any dates across multiple fields are past the end of a quarter, create a computed field with an expression like the one below:

```
DEFINE FIELD Past_Qtr COMPUTED MAXIMUM(PO_Date, Invoice_Date, Payment_
Date, `20160331`)
```

- Records with all dates on or before 31 Mar 2016 return `20160331`.
- Records with one or more dates after 31 Mar 2016 return the most recent date among the three fields.

Remarks

How decimal places work in sets of numeric values

If the numeric values being compared do not have the same number of decimal places, the result is adjusted to the largest number of decimal places.

Returns 20.400:

```
MAXIMUM(3.682, 10.88, 20.4)
```

You can use the DECIMALS( ) function to adjust the number of decimals for *value* parameters.

Returns 20.40:

```
MAXIMUM(DECIMALS(3.682, 2), 10.88, 20.4)
```

MINIMUM( ) function

Returns the minimum value in a set of numeric values, or the oldest value in a set of datetime values.

Syntax

```
MINIMUM(value_1, value_2 <,...n>)
```

Parameters

Name	Type	Description
value_1, *value_2<, ...n>*	numeric datetime	The values to compare, separated by commas. All values must be of the same data type. Additionally, datetime values must be of the same subtype. You cannot mix date, datetime, or time values in a single execution of the function.

Output

Numeric or Datetime.

Examples

Basic examples

Literal numeric input

Returns 4:

```
MINIMUM(4, 7)
```

Returns 3:

```
MINIMUM(4, 7, 3, 8)
```

Returns 3.00:

```
MINIMUM(4, 7.25, 3, 8)
```

Literal datetime input

Returns `20161229`:

```
MINIMUM(`20161231`, `20161229`, `20161230`)
```

Returns `20161229 23:59:59`:

```
MINIMUM(`20161231 235959`, `20161229 235959`)
```

Returns `23:59:57`:

```
MINIMUM(`.235957`, `.235959`, `.235958`)
```

Field input

Returns the oldest date among the three fields for each record:

```
MINIMUM(PO_Date, Invoice_Date, Payment_Date)
```

Advanced examples

Identifying the lowest value among multiple fields

Create a computed field to identify the lowest value among the **Cost**, **Sale_Price**, and **Discount_Price** fields:

```
DEFINE FIELD Low_Value COMPUTED MINIMUM(Cost, Sale_Price, Discount_
Price)
```

Discovering dates before the beginning of a quarter

To discover if any dates across multiple fields are before the beginning of a quarter, create a computed field with an expression like the one below:

```
DEFINE FIELD Pre_Qtr COMPUTED MINIMUM(PO_Date, Invoice_Date, Payment_
Date, `20160101`)
```

- Records with all dates on or after 01 Jan 2016 return `20160101`.
- Records with one or more dates before 01 Jan 2016 return the oldest date among the three fields.

Remarks

How decimal places work in sets of numeric values

If the numeric values being compared do not have the same number of decimal places, the result is adjusted to the largest number of decimal places.

Returns 3.600:

```
MINIMUM(3.6,10.88, 20.482)
```

You can use the DECIMALS( ) function to adjust the number of decimals for *value* parameters.

Returns 3.60:

```
MINIMUM(3.6,10.88, DECIMALS(20.482, 2))
```

The MIN( ) abbreviation

In ACLScript, you can use the abbreviation MIN( ) for the MINIMUM( ) function even though it does not uniquely identify the function, which is the normal requirement when abbreviating function names.

MIN( ) could also be an abbreviation for MINUTE( ), however Analytics reserves the abbreviation MIN ( ) for the MINIMUM( ) function.

MINUTE( ) function

Extracts the minutes from a specified time or datetime and returns it as a numeric value.

Syntax

```
MINUTE(time/datetime)
```

Parameters

Name	Type	Description
time/datetime	datetime	The field, expression, or literal value to extract the minutes from.

Output

Numeric.

Examples

Basic examples

Returns 59:

```
MINUTE(`t235930`)
```

```
MINUTE(`20141231 235930`)
```

Returns the minutes for each value in the **Call_start_time** field:

```
MINUTE(Call_start_time)
```

Remarks

Abbreviating MINUTE( ) in scripts

In ACLScript, if you abbreviate the MINUTE( ) function, you must use at least the first four letters (`MINU`). Analytics reserves the abbreviation `MIN` for the MINIMUM( ) function.

Parameter details

A field specified for *time/datetime* can use any time or datetime format, as long as the field definition correctly defines the format.

Specifying a literal time or datetime value

When specifying a literal time or datetime value for *time/datetime*, you are restricted to the formats in the table below, and you must enclose the value in backquotes – for example, `` `20141231 235959` ``.

Do not use any separators such as slashes (/) or colons (:) between the individual components of dates or times.

- **Time values** – you can use any of the time formats listed in the table below. You must use a separator before a standalone time value for the function to operate correctly. Valid separators are the letter 't', or the letter 'T'. You must specify times using the 24-hour clock. Offsets from Coordinated Universal Time (UTC) must be prefaced by a plus sign (+) or a minus sign (-).
- **Datetime values** – you can use any combination of the date, separator, and time formats listed in the table below. The date must precede the time, and you must use a separator between the two. Valid separators are a single blank space, the letter 't', or the letter 'T'.

Example formats	Example literal values
thhmmss	`t235959`
Thhmm	`T2359`
YYYYMMDD hhmmss	`20141231 235959`
YYMMDDthhmm	`141231t2359`
YYYYMMDDThh	`20141231T23`
YYYYMMDD hhmmss+/-hhmm (UTC offset)	`20141231 235959-0500`
YYMMDD hhmm+/-hh (UTC offset)	`141231 2359+01`

Example formats	Example literal values
Note Do not use hh alone in the main time format with data that has a UTC offset. For example, avoid: hh+hhmm. Results can be unreliable.	

MOD( ) function

Returns the remainder from dividing two numbers.

Syntax

```
MOD(number, divisor_number)
```

Parameters

Name	Type	Description
number	numeric	The number to divide.
divisor_number	numeric	The number to divide *number* by. If *number* or *divisor_number*, or both, include decimals, the output has the same decimal precision as the input value with the higher number of decimal places. For example, the output for `MOD(45.35, 5.3)` is "2.95".

Output

Numeric.

Examples

Basic examples

Returns 3:

```
MOD(93, 10)
```

Returns 2.0:

```
MOD(66, 16.00)
```

Returns 3.45:

```
MOD(53.45, 10)
```

Advanced examples

Calculating an anniversary date

Defines a field that shows the number of months since the last anniversary:

```
DEFINE FIELD Months_since_last_anniversary COMPUTED MOD(Months_of_ser-
vice, 12)
```

Remarks

When to use MOD( )

Use the MOD( ) function to test whether two numbers divide evenly, or to isolate the remainder of a division calculation. This function divides one number by another and returns the remainder.

MONTH( ) function

Extracts the month from a specified date or datetime and returns it as a numeric value (1 to 12).

Syntax

```
MONTH(date/datetime)
```

Parameters

Name	Type	Description
date/datetime	datetime	The field, expression, or literal value to extract the month from.

Output

Numeric.

Examples

Basic examples

Returns 12:

```
MONTH(`20141231`)
```

```
MONTH(`20141231 235959`)
```

Returns the month for each value in the **Invoice_date** field:

```
MONTH(Invoice_date)
```

Remarks

Parameter details

A field specified for *date/datetime* can use any date or datetime format, as long as the field definition correctly defines the format.

Specifying a literal date or datetime value

When specifying a literal date or datetime value for *date/datetime*, you are restricted to the formats in the table below, and you must enclose the value in backquotes – for example, `` `20141231` ``.

Do not use any separators such as slashes (/) or colons (:) between the individual components of dates or times.

- **Datetime values** – you can use any combination of the date, separator, and time formats listed in the table below. The date must precede the time, and you must use a separator between the two. Valid separators are a single blank space, the letter 't', or the letter 'T'.
- **Time values** – you must specify times using the 24-hour clock. Offsets from Coordinated Universal Time (UTC) must be prefaced by a plus sign (+) or a minus sign (-).

Example formats	Example literal values
YYYYMMDD	`20141231`
YYMMDD	`141231`
YYYYMMDD hhmmss	`20141231 235959`
YYMMDDthhmm	`141231t2359`
YYYYMMDDThh	`20141231T23`
YYYYMMDD hhmmss+/-hhmm (UTC offset)	`20141231 235959-0500`
YYMMDD hhmm+/-hh (UTC offset)	`141231 2359+01`
Note Do not use hh alone in the main time format with data that has a UTC offset. For example, avoid: hh+hhmm. Results can be unreliable.	

Related functions

If you need to return the name of the month of the year, use CMOY( ) instead of MONTH( ).

NOMINAL( ) function

Returns the nominal annual interest rate on a loan.

Syntax

```
NOMINAL(effective_rate, periods)
```

Parameters

Name	Type	Description
effective_rate	numeric	The effective annual interest rate.
periods	numeric	The number of compounding periods per year. **Note** Specify an integer. If you specified a decimal portion, it is truncated.

Output

Numeric. The rate is calculated to eight decimals places.

Examples

Basic examples

Returns 0.17998457 (18%), the nominal annual rate of interest on the unpaid balance of a credit card that charges an effective annual rate of 19.56%:

```
NOMINAL(0.1956, 12)
```

Remarks

What is the nominal interest rate?

The nominal annual interest rate on a loan is the stated or posted rate of interest, without taking into account interest on the remaining balance, compounded monthly or daily.

Related functions

The EFFECTIVE( ) function is the inverse of the NOMINAL( ) function.

NORMDIST( ) function

Returns the probability that a random variable from a normally distributed data set is less than or equal to a specified value, or exactly equal to a specified value.

Syntax

```
NORMDIST(x, mean, standard_deviation, cumulative)
```

Parameters

Name	Type	Description
x	numeric	The value for which you want to calculate the probability.
mean	numeric	The mean value of the data set.
standard_deviation	numeric	The standard deviation of the data set. The *standard_deviation* value must be greater than 0.
cumulative	logical	Specify T to calculate the probability that a random variable is less than or equal to *x* (cumulative probability), or F to calculate the probability that a random variable is exactly equal to *x* (simple probability).

Output

Numeric.

Examples

Basic examples

Returns 0.908788780274132:

```
NORMDIST(42, 40, 1.5, T)
```

Returns 0.109340049783996:

```
NORMDIST(42, 40, 1.5, F)
```

NORMSINV( ) function

Returns the z-score associated with a specified probability in a standard normal distribution. The z-score is the number of standard deviations a value is from the mean of a standard normal distribution.

Syntax

```
NORMSINV(probability)
```

Parameters

Name	Type	Description
probability	numeric	The probability for which you want to calculate the z-score.

Output

Numeric.

Examples

Basic examples

Returns 1.333401745213610:

```
NORMSINV(0.9088)
```

NOW( ) function

Returns the current operating system time as a Datetime data type.

Syntax

```
NOW()
```

Parameters

This function does not have any parameters.

Output

Datetime.

Examples

Basic examples

Returns the current operating system time as a datetime value, for example, `t235959`, displayed using the current Analytics time display format:

```
NOW()
```

Remarks

Related functions

If you need to return the current operating system time as a character string, use TIME( ) instead of NOW( ).

NPER( ) function

Returns the number of periods required to pay off a loan.

Syntax

```
NPER(rate, payment, amount <,type>)
```

Parameters

Name	Type	Description
rate	numeric	The interest rate per period.
payment	numeric	The payment per period.
amount	numeric	The principal amount of the loan.
type optional	numeric	The timing of payments: ◦ `0` - payment at the end of a period ◦ `1` - payment at the beginning of a period If omitted, the default value of 0 is used.

Output

Numeric.

Examples

Basic examples

Returns 300.00, the number of months required to pay off a $275,000 loan at 6.5% per annum, with payments of $1,856.82 due at the end of each month:

```
NPER(0.065/12, 1856.82, 275000, 0)
```

Returns 252.81, the number of months required to pay off the same loan, with payments of $2,000 due at the end of each month:

```
NPER(0.065/12, 2000, 275000, 0)
```

Returns 249.92, the number of months required to pay off the same loan, with payments of $2,000 due at the beginning of each month:

```
NPER(0.065/12, 2000, 275000, 1)
```

Advanced examples

Annuity calculations

Annuity calculations involve four variables:

- **present value, or future value** – $21,243.39 and $ 26,973.46 in the examples below
- **payment amount per period** – $1,000.00 in the examples below
- **interest rate per period** – 1% per month in the examples below
- **number of periods** – 24 months in the examples below

If you know the value of three of the variables, you can use an Analytics function to calculate the fourth.

I want to find:	Analytics function to use:
Present value	PVANNUITY( ) Returns 21243.39: `PVANNUITY(0.01, 24, 1000)`
Future value	FVANNUITY( ) Returns 26973.46: `FVANNUITY(0.01, 24, 1000)`
Payment amount per period	PMT( ) Returns 1000:

I want to find:	Analytics function to use:
	`PMT(0.01, 24, 21243.39)`
Interest rate per period	RATE( ) Returns 0.00999999 (1%): `RATE(24, 1000, 21243.39)`
Number of periods	NPER( ) Returns 24.00: `NPER(0.01, 1000, 21243.39)`

Annuity formulas

The formula for calculating the **present value** of an ordinary annuity (payment at the end of a period):

$$PV_A = Pmt\left[\frac{1-\frac{1}{(1+i)^N}}{i}\right]$$

$$21243.39 = 1000\left[\frac{1-\frac{1}{(1+0.01)^{24}}}{0.01}\right]$$

The formula for calculating the **future value** of an ordinary annuity (payment at the end of a period):

$$FV_A = Pmt\left[\frac{(1+i)^N-1}{i}\right]$$

$$26973.46 = 1000\left[\frac{(1+0.01)^{24}-1}{0.01}\right]$$

OCCURS( ) function

Returns a count of the number of times a substring occurs in a specified character value.

Syntax

```
OCCURS(string, search_for)
```

Parameters

Name	Type	Description
string	character	The value to search in. You can concatenate two or more fields if you want to search in more than one field in a table: `OCCURS(First_Name+Last_Name,"John")`
search_for	character	The value to search for. The search is case-sensitive.

Output

Numeric.

Examples

Basic examples

Returns 2:

```
OCCURS("abc/abc/a","ab")
```

Returns 3:

```
OCCURS("abc/abc/a","a")
```

Returns the number of times a hyphen occurs in each value in the **Invoice_Number** field:

```
OCCURS(Invoice_Number, "-")
```

Advanced examples

Finding invoice numbers with more than one hyphen

If invoice numbers in a table should have only one hyphen, use the OCCURS( ) function to create a filter that isolates invoice numbers that have two or more hyphens:

```
SET FILTER TO OCCURS(Invoice_Number, "-") > 1
```

Finding occurrences of one field's value in another field

Use OCCURS( ) to find occurrences of one field's value in another field. For example, you could create a filter that isolates records in which **Last_Name** values occur in the **Full_Name** field:

```
SET FILTER TO OCCURS(Full_Name, ALLTRIM(Last_Name)) = 1
```

Including the ALLTRIM( ) function in the expression removes any leading or trailing spaces from the **Last_Name** field, ensuring that only text values are compared.

Performing case-sensitive searches

Unlike the FIND( ) function, the OCCURS( ) function is case sensitive, which allows you to perform case-sensitive searches.

The following expression isolates all records that contain the name "UNITED EQUIPMENT", in uppercase, in the **Vendor_Name** field, while ignoring occurrences of "United Equipment".

```
SET FILTER TO OCCURS(Vendor_Name, "UNITED EQUIPMENT") > 0
```

If you want to find all occurrences of "United Equipment" regardless of casing, use the UPPER( ) function to convert the search field values to uppercase:

```
SET FILTER TO OCCURS(UPPER(Vendor_Name), "UNITED EQUIPMENT") > 0
```

OFFSET( ) function

Returns the value of a field with the starting position offset by a specified number of bytes.

Syntax

```
OFFSET(field, number_of_bytes)
```

Parameters

Name	Type	Description
field	character numeric datetime	A field name.
number_of_bytes	numeric	Any positive numeric expression.

Output

The return value is the same data type as the input *field* parameter.

Examples

Basic examples

If you have a field called "Number" that contains the value "1234567890" and you define an overlapping field called "Offset_Number" that has a starting position of 1, a length of 3, and no decimals places, you can use the OFFSET( ) function to shift the numbers in the field.

Returns 123:

```
OFFSET(Offset_Number,0)
```

Returns 234:

```
OFFSET(Offset_Number,1)
```

Returns 789:

```
OFFSET(Offset_Number,6)
```

Remarks

You can use this function to temporarily offset the starting position of a field. This is useful when you are processing data where the field starting position is variable.

If you use the OFFSET( ) function with conditional computed fields, any fields referenced in the IF test will also be offset.

OMIT( ) function

Returns a string with one or more specified substrings removed.

Syntax

```
OMIT(string1, string2 <,case_sensitive>)
```

Parameters

Name	Type	Description
string1	character	The field, expression, or literal value to remove one or more substrings from.
string2	character	One or more substrings to remove. ◦ Use commas to separate multiple substrings. ◦ Use a space after a comma only if it is part of the substring you want to remove. ◦ If the double quotation mark character occurs in any of the substrings, enclose the entire *string2* parameter in single quotation marks (' ') rather than double quotation marks. ◦ To omit a comma, place a single comma last in the list of substrings, followed immediately by the closing quotation mark (see the final example below).
case_sensitive optional	logical	Specify `T` to make the substrings case-sensitive, or `F` to ignore case. If *case_sensitive* is omitted, the default value of T is used.

Output

Character.

Examples

Basic examples

Literal character input

Returns "Intercity Couriers":

```
OMIT("Intercity Couriers Corporation", " Corporation, Corp.")
```

Returns "Inter-city Couriers":

```
OMIT("Inter-city Couriers Corp.", " Corporation, Corp.")
```

> **Note**
> The Levenshtein distance between the returned values in the first two examples is 1. If the generic elements are not removed, the distance between the two examples is 8, which could allow the values to escape detection as fuzzy duplicates of each other.

Field input

Returns all the values in the **Vendor_Name** field with generic elements such as "Corporation" and "Inc." removed:

```
OMIT(Vendor_Name," Corporation, Corp., Corp, Inc., Inc, Ltd., Ltd")
```

Returns all the values in the **Vendor_Name** field with generic elements such as "Corporation" and "Inc." removed:

```
OMIT(Vendor_Name," ,.,Corporation,Corp,Inc,Ltd")
```

> **Note**
> The two preceding examples both return the same results but the syntax for the second example is more efficient.

Returns all the values in the **Vendor_Name** field with "Corporation" and "Corp" removed, and all commas removed:

```
OMIT(Vendor_Name," Corporation, Corp,")
```

Remarks

OMIT( ) can remove substrings as units

The OMIT( ) function removes one or more substrings from a string. It differs from functions such as CLEAN( ), EXCLUDE( ), INCLUDE( ), and REMOVE( ) because it matches and removes characters on a substring basis rather than on a character-by-character basis. Substring removal allows you to remove specific words, abbreviations, or repeated sequences of characters from a string without affecting the remainder of the string.

A helper function for fuzzy comparisons

OMIT( ) can improve the effectiveness of the ISFUZZYDUP( ) function, or the FUZZYDUP or FUZZYJOIN commands, by removing generic elements such as "Corporation" or "Inc.", or characters such as commas, periods, and ampersands (&), from field values.

Removal of generic elements and punctuation focuses the fuzzy comparison on just the portion of the values where a meaningful difference may occur.

How the order of substrings affects results

If you specify multiple substrings for removal, the order in which you list them in *string2* can affect the output results.

When the OMIT( ) function is processed, the first substring is removed from all values that contain it, the second substring is then removed from all values that contain it, and so on. If one substring forms part of another substring – for example, "Corp" and "Corporation" – removing the shorter substring first also alters values containing the longer substring ("Corporation" becomes "oration") and prevents the longer substring from being found.

To avoid this situation, specify longer substrings before any shorter substrings they contain. For example:

```
OMIT(Vendor_Name," Corporation, Corp., Corp")
```

Try removing special characters first

You can specify single-character substrings, such as punctuation marks, special characters, and a blank space, which further reduces the generic material in strings. It may be more efficient to remove

a single character first, such as a period or a blank, which reduces the number of substring variations you subsequently need to specify. Compare the third and fourth examples above. They both return the same results, but the fourth example is more efficient.

Dealing with blanks or spaces

Blanks or spaces in substrings are treated like any other character. You must explicitly specify each blank you want removed as part of a substring. For example, if you specify an ampersand without any blanks ("&"), "Ricoh Sales & Service" becomes "Ricoh Sales Service". If you include blanks (" & "), "Ricoh Sales & Service" becomes "Ricoh SalesService".

If you specify a blank that is not part of the substring, the substring will not be found. For example, if you specify an ampersand with blanks (" & "), "Ricoh Sales&Service" remains unchanged.

When using commas to separate multiple substrings, follow the comma with a space only if it corresponds with the actual substring you want to remove.

One approach to dealing with blanks is to remove all blanks from a field first, by specifying a blank as a single-character substring prior to specifying any other substrings.

Review the results of using OMIT( )

After using OMIT( ) to created a computed field, review the contents of the field to confirm that you have not inadvertently omitted meaningful portions of strings. For example, omitting "Co" gets rid of a common abbreviation for "Company", but it also removes the letters "Co" from two locations in "Coca-Cola".

PACKED( ) function

Returns numeric data converted to the Packed data type.

Syntax

```
PACKED(number, length_of_result)
```

Parameters

Name	Type	Description
number	numeric	The numeric value or field to convert.
length_of_result	numeric	The number of bytes to use in the output string.

Output

Numeric.

Examples

Basic examples

Integer and decimal input

Returns 00075C:

```
PACKED(75, 3)
```

```
PACKED(7.5, 3)
```

Truncated digits in output

Returns 00000012456D:

```
PACKED(-12.456, 6)
```

Returns 456D:

```
PACKED(-12.456, 2)
```

Advanced examples

Creating an 8-byte field to update a mainframe

You need to create an 8-byte field containing each employee's salary as a PACKED number for uploading to a mainframe:

```
EXTRACT PACKED(SALARY, 8) AS "Salary_Export" TO "export"
```

Remarks

What is Packed data?

The Packed data type is used by mainframe operating systems to store numeric values in a format that uses minimal storage space. The Packed data type stores two digits in each byte, and the last byte indicates whether the value is positive or negative.

When to use PACKED( )

Use the PACKED( ) function to convert numeric data to the Packed format for export to mainframe systems.

Truncated return values

If the *length_of_result* value is shorter than the length of the *number* value, the additional digits are truncated.

PI( ) function

Returns the value of pi to 15 decimal places.

Syntax

```
PI( )
```

Parameters

This function does not have any parameters.

Output

Numeric.

Examples

Basic examples

Returns 3.141592653589793 (the value of pi to 15 decimal places):

```
PI( )
```

Returns 1.047197551196598 (the equivalent in radians of 60 degrees):

```
60 * PI( )/180
```

Advanced examples

Using degrees as input

Returns 0.866025403784439 (the sine of 60 degrees):

```
SIN(60 * PI( )/180)
```

Remarks

When to use PI( )

Use PI( ) to convert degrees to radians: (*degrees* * PI( )/180) = *radians*. Radians are the required input for three of Analytics's math functions: SIN( ), COS( ), and TAN( ).

PMT( ) function

Returns the amount of the periodic payment (principal + interest) required to pay off a loan.

Syntax

```
PMT(rate, periods, amount <,type>)
```

Parameters

Name	Type	Description
rate	numeric	The interest rate per period.
periods	numeric	The total number of payment periods.
amount	numeric	The principal amount of the loan.
type optional	numeric	The timing of payments: ◦ `0` - payment at the end of a period ◦ `1` - payment at the beginning of a period If omitted, the default value of 0 is used.

> **Note**
> You must use consistent time periods when specifying *rate* and *periods* to ensure that you are specifying interest rate **per period**.
>
> For example:
>
> - for monthly payments on a two-year loan or investment with interest of 5% per annum, specify 0.05/12 for *rate* and 2 * 12 for *periods*
> - for annual payments on the same loan or investment, specify 0.05 for *rate* and 2 for *periods*

Output

Numeric.

Examples

Basic examples

Returns 1856.82, the monthly payment (principal + interest) required to pay off a twenty-five year, $275,000 loan at 6.5% per annum, with payments due at the end of the month:

```
PMT(0.065/12, 12*25, 275000, 0)
```

Returns 1846.82, the monthly payment (principal + interest) required to pay off the same loan, with payments due at the beginning of the month:

```
PMT(0.065/12, 12*25, 275000, 1)
```

Advanced examples

Annuity calculations

Annuity calculations involve four variables:

- **present value, or future value** – $21,243.39 and $ 26,973.46 in the examples below
- **payment amount per period** – $1,000.00 in the examples below
- **interest rate per period** – 1% per month in the examples below
- **number of periods** – 24 months in the examples below

If you know the value of three of the variables, you can use an Analytics function to calculate the fourth.

I want to find:	Analytics function to use:
Present value	PVANNUITY( ) Returns 21243.39: `PVANNUITY(0.01, 24, 1000)`
Future value	FVANNUITY( ) Returns 26973.46:

I want to find:	Analytics function to use:
	`FVANNUITY(0.01, 24, 1000)`
Payment amount per period	PMT( ) Returns 1000: `PMT(0.01, 24, 21243.39)`
Interest rate per period	RATE( ) Returns 0.00999999 (1%): `RATE(24, 1000, 21243.39)`
Number of periods	NPER( ) Returns 24.00: `NPER(0.01, 1000, 21243.39)`

Annuity formulas

The formula for calculating the **present value** of an ordinary annuity (payment at the end of a period):

$$PV_A = Pmt\left[\frac{1-\frac{1}{(1+i)^N}}{i}\right]$$

$$21243.39 = 1000\left[\frac{1-\frac{1}{(1+0.01)^{24}}}{0.01}\right]$$

The formula for calculating the **future value** of an ordinary annuity (payment at the end of a period):

$$FV_A = Pmt\left[\frac{(1+i)^N-1}{i}\right]$$

$$26973.46 = 1000\left[\frac{(1+0.01)^{24}-1}{0.01}\right]$$

PPMT( ) function

Returns the principal paid on a loan for a single period.

Syntax

```
PPMT(rate, specified_period, periods, amount <,type>)
```

Parameters

Name	Type	Description
rate	numeric	The interest rate per period.
specified_period	numeric	The period for which you want to find the principal payment.
periods	numeric	The total number of payment periods.
amount	numeric	The principal amount of the loan.
type optional	numeric	The timing of payments: ◦ `0` - payment at the end of a period ◦ `1` - payment at the beginning of a period If omitted, the default value of 0 is used.

> **Note**
> You must use consistent time periods when specifying *rate* and *periods* to ensure that you are specifying interest rate **per period**.
>
> For example:
>
> - for monthly payments on a two-year loan or investment with interest of 5% per annum, specify 0.05/12 for *rate* and 2 * 12 for *periods*
> - for annual payments on the same loan or investment, specify 0.05 for *rate* and 2 for *periods*

Output

Numeric.

Examples

Basic examples

Returns 367.24, the principal paid in the first month of a twenty-five year, $275,000 loan at 6.5% per annum, with payments due at the end of the month:

```
PPMT(0.065/12, 1, 12*25, 275000, 0)
```

Returns 1846.82, the principal paid on the same loan in the last month of the loan:

```
PPMT(0.065/12, 300, 12*25, 275000, 0)
```

Remarks

Related functions

The IPMT( ) function is the complement of the PPMT( ) function.

The CUMPRINC( ) function calculates principal paid during a range of periods.

PROPER( ) function

Returns a string with the first character of each word set to uppercase and the remaining characters set to lowercase.

Syntax

```
PROPER(string)
```

Parameters

Name	Type	Description
string	character	The field, expression, or literal value to convert to proper case.

Output

Character.

Examples

Basic examples

Returns "John Doe":

```
PROPER("JOHN DOE")
```

Returns "John Doe":

```
PROPER("john doe")
```

Returns "1550 Alberni St.":

```
PROPER("1550 ALBERNI st.")
```

Returns "Bill O'Hara":

```
PROPER("BILL O'HARA")
```

Returns all the values in the **Company_Name** field converted to proper case:

```
PROPER(Company_Name)
```

Remarks

How it works

The PROPER( ) function converts the first character in *string*, and any subsequent character preceded by a blank, to uppercase.

Subsequent characters preceded by a hyphen, an apostrophe, an ampersand (&), and several other punctuation marks and special characters are also converted to uppercase. All other alphabetic characters are converted to lowercase.

When to use PROPER( )

The most common use for PROPER( ) is to convert names stored as all uppercase or all lowercase in source data into proper case format, so the name is displayed correctly in a form letter or report.

PROPERTIES( ) function

Returns properties information for the specified Analytics project item.

Syntax

```
PROPERTIES(name, obj_type, info_type)
```

Parameters

Name	Type	Description
name	character	The name of the Analytics project item you want information about. *name* is not case-sensitive. If the project item is an Analytics table, specify the table layout name, not the data file name. For example: `"Invoices"`, not `"january_invoices.fil"` If you are using the PROPERTIES( ) function to return the name of the active table, specify the *name* `"activetable"`
obj_type	character	The type of Analytics project item referred to by *name*. **Note** Currently, `"table"` is the only type of project item supported.
info_type	character	The type of information you want about the Analytics project item. For more information, see "Types of properties information" on page 2287.

Output

Character. The maximum length of the output string is 260 characters. If properties information cannot be found, an empty string is returned.

Examples

Basic examples

Information about the Analytics data file (.fil)

Returns "Ap_Trans.fil":

```
PROPERTIES("Ap_Trans", "table", "filename")
```

Returns "C:\ACL DATA\Sample Data Files":

```
PROPERTIES("Ap_Trans", "table", "filepath")
```

Information about the open Analytics table

Returns "Ap_Trans":

```
PROPERTIES("activetable", "table", "open")
```

Information about an external data source

Returns "Trans_May.xls":

```
PROPERTIES("Trans_May", "table", "sourcename")
```

Returns "C:\Project Data\Monthly Invoices_Excel":

```
PROPERTIES("Trans_May", "table", "sourcepath")
```

Returns "EXCEL":

```
PROPERTIES("Trans_May", "table", "sourcetype")
```

Remarks

File information

Information types starting with "file" provide information about the Analytics data file (.fil) associated with an Analytics table.

Source information

Information types starting with "source" provide information about external data sources that can be associated with an Analytics table. Only those external data sources that support refreshing an Analytics table can be reported on using the PROPERTIES( ) function:

- Microsoft Excel
- Microsoft Access
- Delimited text
- Adobe Acrobat (PDF)
- Print Image (Report)
- SAP private file format/DART
- XML
- XBRL
- ODBC data sources

Types of properties information

The table below lists the types of properties information that can be returned by the PROPERTIES( ) function. Analytics tables are the only Analytics project items that can currently be used with the PROPERTIES( ) function:

obj_type	info_type	Returns:
`"table"`	`"filename"`	The name of the data file associated with the Analytics table.
	`"filepath"`	The path of the data file associated with the Analytics table.
	`"filesize"`	The size, in KB, of the data file associated with the Analytics table.
	`"filemodifiedat"`	The time and date that the data file associated with the Analytics table was last modified.
	`"sourcename"`	The name of the data source associated with the Analytics table. Data sources can be external files such as Excel, Access, PDF, XML, or delimited text files, or ODBC data sources.
	`"sourcepath"`	The path of the data source associated with the Analytics table.

obj_type	info_type	Returns:
		Not supported for ODBC data sources.
	`"sourcetype"`	The type of the data source associated with the Analytics table.
	`"sourcesize"`	The size, in KB, of the data source associated with the Analytics table. Not supported for ODBC data sources.
	`"sourcemodifiedat"`	The time and date that the data source associated with the Analytics table was last modified. Not supported for ODBC data sources.
	`"open"`	The name of the currently active Analytics table. **Note** Multiple Analytics tables can be open at the same time, but in the user interface only one table at a time can be active.

PVANNUITY( ) function

Returns the present value of a series of future payments calculated using a constant interest rate. Present value is the current, lump-sum value.

Syntax

```
PVANNUITY(rate, periods, payment <,type>)
```

Parameters

Name	Type	Description
rate	numeric	The interest rate per period.
periods	numeric	The total number of payment periods.
payment	numeric	The payment per period. The payment amount must remain constant over the term of the annuity.
type optional	numeric	The timing of payments: ◦ `0` - payment at the end of a period ◦ `1` - payment at the beginning of a period If omitted, the default value of 0 is used.

> **Note**
> You must use consistent time periods when specifying *rate*, *periods*, and *payment* to ensure that you are specifying interest rate **per period**.
>
> For example:
>
> - for a monthly *payment* on a two-year loan or investment with interest of 5% per annum, specify 0.05/12 for *rate* and 2 * 12 for *periods*
> - for an annual *payment* on the same loan or investment, specify 0.05 for *rate* and 2 for *periods*

Output

Numeric. The result is calculated to two decimal places.

Examples

Basic examples

Monthly payments

Returns 21455.82, the present value of $1,000 paid at the beginning of each month for 2 years at 1% per month, compounded monthly:

```
PVANNUITY(0.01, 2*12, 1000, 1)
```

Annual payments

Returns 20280.61, the present value of $12,000 paid at the end of each year for 2 years at 12% per annum, compounded annually:

```
PVANNUITY(0.12, 2, 12000, 0)
```

Advanced examples

Annuity calculations

Annuity calculations involve four variables:

- **present value, or future value** – $21,243.39 and $ 26,973.46 in the examples below
- **payment amount per period** – $1,000.00 in the examples below
- **interest rate per period** – 1% per month in the examples below
- **number of periods** – 24 months in the examples below

If you know the value of three of the variables, you can use an Analytics function to calculate the fourth.

I want to find:	Analytics function to use:
Present value	PVANNUITY( ) Returns 21243.39:

I want to find:	Analytics function to use:
	`PVANNUITY(0.01, 24, 1000)`
Future value	FVANNUITY( ) Returns 26973.46: `FVANNUITY(0.01, 24, 1000)`
Payment amount per period	PMT( ) Returns 1000: `PMT(0.01, 24, 21243.39)`
Interest rate per period	RATE( ) Returns 0.00999999 (1%): `RATE(24, 1000, 21243.39)`
Number of periods	NPER( ) Returns 24.00: `NPER(0.01, 1000, 21243.39)`

Annuity formulas

The formula for calculating the **present value** of an ordinary annuity (payment at the end of a period):

$$PV_A = Pmt\left[\frac{1-\frac{1}{(1+i)^N}}{i}\right]$$

$$21243.39 = 1000\left[\frac{1-\frac{1}{(1+0.01)^{24}}}{0.01}\right]$$

The formula for calculating the **future value** of an ordinary annuity (payment at the end of a period):

$$FV_A = Pmt\left[\frac{(1+i)^N - 1}{i}\right]$$

$$26973.46 = 1000\left[\frac{(1+0.01)^{24} - 1}{0.01}\right]$$

Remarks

Related functions

The FVANNUITY( ) function is the inverse of the PVANNUITY( ) function.

PVLUMPSUM( ) function

Returns the present value required to generate a specific future lump sum calculated using a constant interest rate. Present value is the current, lump-sum value.

Syntax

```
PVLUMPSUM(rate, periods, amount)
```

Parameters

Name	Type	Description
rate	numeric	The interest rate per period.
periods	numeric	The total number of periods.
amount	numeric	The value of the future lump sum at the end of the last period.

> **Note**
>
> You must use consistent time periods when specifying *rate* and *periods* to ensure that you are specifying interest rate **per period**.
>
> For example:
>
> - for monthly payments on a two-year loan or investment with interest of 5% per annum, specify 0.05/12 for *rate* and 2 * 12 for *periods*
> - for annual payments on the same loan or investment, specify 0.05 for *rate* and 2 for *periods*

Output

Numeric. The result is calculated to two decimal places.

Examples

Basic examples

Interest compounded monthly

Returns 1000.00, the initial investment principal required to generate a future lump sum of $1,269.73, when invested for 2 years at 1% per month, compounded monthly:

```
PVLUMPSUM(0.01, 2*12, 1269.73)
```

Returns 787.57, the initial investment principal required to generate a future lump sum of $1,000, when invested for 2 years at 1% per month, compounded monthly:

```
PVLUMPSUM(0.01, 2*12, 1000)
```

Returns 21455.82, the initial investment principal required to generate a future lump sum of $27,243.20, when invested for 2 years at 1% per month, compounded monthly:

```
PVLUMPSUM(0.01, 2*12, 27243.20)
```

Interest compounded semi-annually

Returns 792.09, the initial investment principal required to generate a future lump sum of $1,000, when invested for 2 years at 12% per annum, compounded semi-annually:

```
PVLUMPSUM(0.12/2, 2*2, 1000)
```

Interest compounded annually

Returns 797.19, the initial investment principal required to generate a future lump sum of $1,000, when invested for 2 years at 12% per annum, compounded annually:

```
PVLUMPSUM(0.12, 2, 1000)
```

Remarks

What is present value?

The present value of an invested lump sum is the initial principal required to generate a specific future lump sum, within a particular time frame. The future value is the principal plus the accumulated compound interest.

Related functions

The FVLUMPSUM( ) function is the inverse of the PVLUMPSUM( ) function.

PYDATE( ) function

Returns a date value calculated by a function in a external Python script. Data processing in Python is external to Analytics.

Syntax

```
PYDATE("PyFile,PyFunction" <, field|value <,...n>>)
```

Parameters

Name	Type	Description
PyFile,PyFunction	character	The name of the Python script to run followed by a comma and then the name of the function that returns the value: `"myScript,myFunction"` When specifying the Python script, omit the file extension. The function you call may call other functions within the script or within other scripts, however all scripts that run must be placed inside a folder in the `PYTHONPATH` system environment variable prior to running. For more information, see "Configuring Python for use with Analytics" on page 2601. **Note** Your *PyFunction* must return a Python datetime.date object.
field |value <,...n> optional	character numeric datetime logical	This list of fields, expressions, or literal values to use as arguments for the Python function. The values are passed into the function you call in the order you specify them. You may include as many arguments as necessary to satisfy the function definition in the Python script. **Note** Use the `ALLTRIM()` function to remove any leading or trailing spaces from character input: `ALLTRIM(str)`. For more information, see "ALLTRIM( ) function" on page 2041.

Output

Datetime.

Examples

Basic examples

Returns `20160630`:

```
PYDATE('hello,due_date', `20160531`, 30)
```

External Python script that accepts a date and a grace period as a number of days and calculates the date the invoice is due. For an invoice date of **2016-05-31** and a period of 30 days: "2016-06-30":

```
#! python
from datetime import timedelta

def due_date(inv_date, period):
    return(inv_date + timedelta(period))
```

Advanced examples

Defining a computed field

Defines a computed field in the `Ap_Trans` table using the Python script that calculates due date:

```
OPEN Ap_Trans
DEFINE FIELD due_date COMPUTED
WIDTH 27
    PYDATE( "hello,due_date" ,Invoice_Date, Pay_Period)
```

PYDATETIME( ) function

Returns a datetime value calculated by a function in an external Python script. Data processing in Python is external to Analytics.

Syntax

```
PYDATETIME("PyFile,PyFunction" <, field|value <,...n>>)
```

Parameters

Name	Type	Description
PyFile,PyFunction	character	The name of the Python script to run followed by a comma and then the name of the function that returns the value: `"myScript,myFunction"` When specifying the Python script, omit the file extension. The function you call may call other functions within the script or within other scripts, however all scripts that run must be placed inside a folder in the `PYTHONPATH` system environment variable prior to running. For more information, see "Configuring Python for use with Analytics" on page 2601. **Note** Your *PyFunction* must return a Python datetime object.
field |value <,...n> optional	character numeric datetime logical	This list of fields, expressions, or literal values to use as arguments for the Python function. The values are passed into the function you call in the order you specify them. You may include as many arguments as necessary to satisfy the function definition in the Python script. **Note** Use the `ALLTRIM()` function to remove any leading or trailing spaces from character input: `ALLTRIM(str)`. For more information, see "ALLTRIM( ) function" on page 2041.

Output

Datetime.

Examples

Basic examples

Returns `20170101t0500`:

```
PYDATETIME("hello, combine_date_time", `20170101`, `t0500`)
```

External Python script that accepts a date argument and a time argument and returns a combined Datetime object:

```
# hello.py content
from datetime import datetime

def combine_date_time(d,t):
    return datetime.combine(d,t)
```

Advanced examples

Adding time to a datetime

Returns `20160101t2230`:

```
PYDATETIME("hello,add_time", `20160101 150000`, `t073000`)
```

External Python script that accepts a datetime and a time and adds the time to the datetime: 2016-01-01 15:00:00 + 7 hours, 30 minutes, 00 seconds = 2016-01-01 22:30:00.

```
# hello.py content
from datetime import timedelta
```

```
from datetime import datetime
from datetime import time
def add_time(start, time_to_add):
    return start + timedelta(hours=time_to_add.hour, minutes=time_to_
add.minute, seconds=time_to_add.second)
```

PYLOGICAL( ) function

Returns a logical value calculated by a function in an external Python script. Data processing in Python is external to Analytics.

Syntax

```
PYLOGICAL("PyFile,PyFunction" <, field|value <,...n>>)
```

Parameters

Name	Type	Description
PyFile,PyFunction	character	The name of the Python script to run followed by a comma and then the name of the function that returns the value: `"myScript,myFunction"` When specifying the Python script, omit the file extension. The function you call may call other functions within the script or within other scripts, however all scripts that run must be placed inside a folder in the `PYTHONPATH` system environment variable prior to running. For more information, see "Configuring Python for use with Analytics" on page 2601. **Note** Your *PyFunction* must return a Python truth value.
field |value <,...n> optional	character numeric datetime logical	This list of fields, expressions, or literal values to use as arguments for the Python function. The values are passed into the function you call in the order you specify them. You may include as many arguments as necessary to satisfy the function definition in the Python script. **Note** Use the `ALLTRIM()` function to remove any leading or trailing spaces from character input: `ALLTRIM (str)`. For more information, see "ALLTRIM( ) function" on page 2041.

Output

Logical.

Examples

Basic examples

Returns F:

```
PYLOGICAL( "hello,str_compare", "basketball", "baseball", "b" )
```

External Python script that compares *str1* and *str2* using the count of the character that is passed in as *char*:

```
# hello.py content
def str_compare(str1, str2, char):
    return str1.count(char) > str2.count(char)
```

Advanced examples

Using fields

Returns a truth value when comparing Vendor_Name and Vendor_City:

```
PYLOGICAL( "hello,str_compare", Vendor_Name, Vendor_City, "b" )
```

External Python script that compares *str1* and *str2* using the count of the character that is passed in as *char*:

```
# hello.py content
def str_compare(str1, str2, char):
    return str1.count(char) > str2.count(char)
```

PYNUMERIC( ) function

Returns a numeric value calculated by a function in an external Python script. Data processing in Python is external to Analytics.

Syntax

```
PYNUMERIC(PyFile,PyFunction, decimal <, field|value <,...n>>)
```

Parameters

Name	Type	Description
PyFile,PyFunction	character	The name of the Python script to run followed by a comma and then the name of the function that returns the value: `"myScript,myFunction"` When specifying the Python script, omit the file extension. The function you call may call other functions within the script or within other scripts, however all scripts that run must be placed inside a folder in the `PYTHONPATH` system environment variable prior to running. For more information, see "Configuring Python for use with Analytics" on page 2601. **Note** Your *PyFunction* must return a Python numeric type.
decimal	numeric	The number of decimal places to include in the return value. Must be a positive integer.
field |value <,...n> optional	character numeric datetime logical	This list of fields, expressions, or literal values to use as arguments for the Python function. The values are passed into the function you call in the order you specify them. You may include as many arguments as necessary to satisfy the function definition in the Python script.

Name	Type	Description
		Note Use the `ALLTRIM()` function to remove any leading or trailing spaces from character input: `ALLTRIM(str)`. For more information, see "ALLTRIM( ) function" on page 2041.

Output

Numeric.

Examples

Basic examples

Returns 35.00:

```
PYNUMERIC("hello,get_nth_percent", 2, 80, 120, 30, 45, 30, 100, 35, 45)
```

External Python script that returns the value at the requested percentile from a dynamically sized list of values:

```
# hello.py content
from math import ceil
def get_nth_percent(percentage, *values):
    input_length = len(values)
    position = ceil((percentage/100.00) * input_length)
    return values[position-1]
```

PYSTRING( ) function

Returns a character value calculated by a function in an external Python script. Data processing in Python is external to Analytics.

Syntax

```
PYSTRING("PyFile,PyFunction", length <,field|value <,...n>>)
```

Name	Type	Description
PyFile,PyFunction	character	The name of the Python script to run followed by a comma and then the name of the function that returns the value: `"myScript,myFunction"` When specifying the Python script, omit the file extension. The function you call may call other functions within the script or within other scripts, however all scripts that run must be placed inside a folder in the `PYTHONPATH` system environment variable prior to running. For more information, see "Configuring Python for use with Analytics" on page 2601. **Note** Your *PyFunction* must return a Python string object.
length	numeric	The length to allocate for the return string.
field |value <,...n> optional	character numeric datetime logical	This list of fields, expressions, or literal values to use as arguments for the Python function. The values are passed into the function you call in the order you specify them. You may include as many arguments as necessary to satisfy the function definition in the Python script. **Note** Use the `ALLTRIM()` function to remove any leading or trailing spaces from character input: `ALLTRIM (str)`. For more information, see "ALLTRIM( ) function" on page 2041.

Output

Character.

Examples

Basic examples

Returns "my test":

```
PYSTRING('hello,main', 20, "my")
```

External Python script that accepts a string and concatenates " test" to the string:

```
#! python
# hello.py content
def main(str):
    str2 = str + ' test'
    return(str2)
```

Advanced examples

Returning a substring

This example removes the last two characters from the Vendor Name field and returns the substring:

```
PYSTRING( "hello,sub_set", LENGTH(Vendor_Name), ALLTRIM(Vendor_Name),
LENGTH(ALLTRIM(Vendor_Name)), 0, LENGTH(ALLTRIM(Vendor_Name)) - 2)
```

External Python script that accepts a string, a string length, and two character positions. The function returns a substring between position one and position two:

```
#hello.py content
def sub_set(str, length, p1, p2):
    if p1 >= 0 and p2 < length and p1 < p2:
        str2 = str[p1:p2]
    else:
        str2 = str
    return str2
```

PYTIME( ) function

Returns a time value calculated by a function in an external Python script. Data processing in Python is external to Analytics.

Syntax

```
PYTIME("PyFile,PyFunction" <, field|value <,...n>>)
```

Parameters

Name	Type	Description
PyFile,PyFunction	character	The name of the Python script to run followed by a comma and then the name of the function that returns the value: `"myScript,myFunction"` When specifying the Python script, omit the file extension. The function you call may call other functions within the script or within other scripts, however all scripts that run must be placed inside a folder in the `PYTHONPATH` system environment variable prior to running. For more information, see "Configuring Python for use with Analytics" on page 2601. **Note** Your *PyFunction* must return a Python datetime.time object.
field |value <,...n> optional	character numeric datetime logical	This list of fields, expressions, or literal values to use as arguments for the Python function. The values are passed into the function you call in the order you specify them. You may include as many arguments as necessary to satisfy the function definition in the Python script. **Note** Use the `ALLTRIM()` function to remove any leading or trailing spaces from character input: `ALLTRIM(str)`. For more information, see "ALLTRIM( ) function" on page 2041.

Output

Datetime.

Examples

Basic examples

Returns `t2122`:

```
ASSIGN v_time_part = PYTIME("hello,get_time", `20160101 212223`)
```

External Python script:

```
# hello.py content
from datetime import time
from datetime import date

def get_time(timestamp):
   return timestamp.time();
```

RAND( ) function

Returns a random number that falls within a specified boundary.

Syntax

```
RAND(number)
```

Parameters

Name	Type	Description

number	numeric	The numeric boundary for the random number.
		If you specify a number with decimal places the random number generated has the same number of decimal places.
		◦ **If you enter a positive number** - the random number returned is greater than or equal to zero, and less than the number you specified.
		Returns a number from 0 to 99:
		`RAND(100)`
		◦ **If you enter a negative number** - the random number returned is less than zero, and greater than or equal to the number you specified.
		Returns a number from -1 to -100:
		`RAND(-100)`

Output

Numeric.

Examples

Basic examples

Returns 278.61:

```
RAND(1000.00)
```

Returns 3781:

```
RAND(10000)
```

> **Note**
> The return value will differ with each execution of the function.

Remarks

RAND( ) cannot replicate results

If you use the RAND( ) function consecutively with the same *number* value, it produces different results. Unlike the RANDOM command, the RAND( ) function has no seed value.

Duplicate random numbers possible

If you use RAND( ) to create a computed field that assigns a random number to every record in a table, it is possible that duplicate random numbers are generated. There is no guarantee that the random numbers will be unique.

The larger the *number* value in relation to the number of records in the table, the greater the chance that the generated numbers will be unique.

Random numbers dynamically update

A computed field with the RAND( ) function generates a new set of random numbers every time you perform actions such as quick sorting, applying a filter, rearranging columns, or scrolling through the view.

If you want to fix a set of random numbers, extract the data to a new table using the **View** or **Fields** option in the **Extract** dialog box.

RATE( ) function

Returns the interest rate per period.

Syntax

```
RATE(periods, payment, amount)
```

Parameters

Name	Type	Description
periods	numeric	The total number of payment periods.
payment	numeric	The payment per period.
amount	numeric	The principal amount of the loan.

> **Note**
> The RATE( ) function assumes that payments are made at the end of each period.

Output

Numeric. The rate is calculated to eight decimals places.

Examples

Basic examples

Returns 0.00541667 (0.54%), the monthly interest rate implied by a twenty-five year, $275,000 loan with monthly payments of $1,856.82:

```
RATE(12*25, 1856.82, 275000)
```

Returns 0.06500004 (6.5%), the annual interest rate implied by the same loan:

```
RATE(12*25, 1856.82, 275000)*12
```

Advanced examples

Converting the nominal rate to the effective rate

The RATE( ) function calculates the nominal interest rate. You can use the EFFECTIVE ( ) function to convert the result of RATE( ) to the effective interest rate.

Returns 0.06715155 (6.7%), the effective annual interest rate implied by the loan in the examples above:

```
EFFECTIVE((RATE(12*25, 1856.82, 275000)*12), 12*25)
```

Annuity calculations

Annuity calculations involve four variables:

- **present value, or future value** – $21,243.39 and $ 26,973.46 in the examples below
- **payment amount per period** – $1,000.00 in the examples below
- **interest rate per period** – 1% per month in the examples below
- **number of periods** – 24 months in the examples below

If you know the value of three of the variables, you can use an Analytics function to calculate the fourth.

I want to find:	Analytics function to use:
Present value	PVANNUITY( ) Returns 21243.39: `PVANNUITY(0.01, 24, 1000)`
Future value	FVANNUITY( ) Returns 26973.46:

I want to find:	Analytics function to use:
	`FVANNUITY(0.01, 24, 1000)`
Payment amount per period	PMT( ) Returns 1000: `PMT(0.01, 24, 21243.39)`
Interest rate per period	RATE( ) Returns 0.00999999 (1%): `RATE(24, 1000, 21243.39)`
Number of periods	NPER( ) Returns 24.00: `NPER(0.01, 1000, 21243.39)`

Annuity formulas

The formula for calculating the **present value** of an ordinary annuity (payment at the end of a period):

$$PV_A = Pmt\left[\frac{1-\frac{1}{(1+i)^N}}{i}\right]$$

$$21243.39 = 1000\left[\frac{1-\frac{1}{(1+0.01)^{24}}}{0.01}\right]$$

The formula for calculating the **future value** of an ordinary annuity (payment at the end of a period):

$$FV_A = Pmt\left[\frac{(1+i)^N-1}{i}\right]$$

$$26973.46 = 1000\left[\frac{(1+0.01)^{24}-1}{0.01}\right]$$

RDATE( ) function

Returns a date value calculated by an R function or script. Data processing in R is external to Analytics.

Syntax

```
RDATE(rScript|rCode <,field|value <,...n>>)
```

Parameters

Name	Type	Description
rScript | *rCode*	character	The full or relative path to the R script, or a snippet of R code to run. If you enter R code directly rather than use an external file, you cannot use the enclosing quotation character in your code, even if you escape it: ◦ **valid** - `'var <- "\"test\"" '` ◦ **invalid** - `'var <- "\'test\'" '`
field | *value* <,...n> optional	character numeric datetime logical	The list of fields, expressions, or literal values to use as arguments for the R script or code snippet. The values are passed into the function you call in the order you specify them, and you reference them using `value1, value2 ... valueN` in the R code. You may include as many arguments as necessary to satisfy the function definition in the R code. **Note** Use the `ALLTRIM()` function to remove any leading or trailing spaces from character input: `ALLTRIM(str)`. For more information, see "ALLTRIM( ) function" on page 2041.

Output

Datetime.

Examples

Basic examples

Returns `20160530`:

```
RDATE("as.Date(value1,'%m-%d-%Y')", "05-30-16")
```

Advanced examples

Using an external R script

Converts a string to a date and returns it:

```
RDATE("a<-source('c:\\scripts\\r_scripts\\sample.r');a[[1]]", dateText)
```

External R script (sample.r):

```
dateForm <- function(dateText) {
    return(as.Date(dateText,format='%y%m%d'))
}
dateForm(value1)
```

Remarks

Returning data from R

When calling R scripts, use the `source` function and assign the return object to a variable. You can then access the value returned from your R function from the return object:

```
# 'a' holds the response object and a[[1]] access the data value
"a<-source('c:\\scripts\\r_scripts\\sample.r');a[[1]]"
```

R log file

Analytics logs R language messages to an `aclrlang.log` file in the project folder. Use this log file for debugging R errors.

> **Tip**
> The log file is available in the Results folder of Analytics Exchange analytic jobs.

Running external R scripts on AX Server

If you are writing an analysis app to run on AX Server and you want to work with external R scripts:

1. Upload the file as a related file with the analysis app.
2. Use the `FILE` analytic tag to identify the file(s).
3. Reference the file(s) using the relative path `./filename.r`.

> **Note**
> Using a related file ensures that the TomEE application server account has sufficient permissions to access the file when running R with Analytics Exchange.

RDATETIME( ) function

Returns a datetime value calculated by an R function or script. Data processing in R is external to Analytics.

Syntax

```
RDATETIME(rScript|rCode <,field|value <,...n>>)
```

Parameters

Name	Type	Description
rScript | *rCode*	character	The full or relative path to the R script, or a snippet of R code to run. If you enter R code directly rather than use an external file, you cannot use the enclosing quotation character in your code, even if you escape it: ◦ **valid** - `'var <- "\"test\"" '` ◦ **invalid** - `'var <- "\'test\'" '`
field | *value <,...n>* optional	character numeric datetime logical	The list of fields, expressions, or literal values to use as arguments for the R script or code snippet. The values are passed into the function you call in the order you specify them, and you reference them using `value1, value2 ... valueN` in the R code. You may include as many arguments as necessary to satisfy the function definition in the R code. **Note** Use the `ALLTRIM()` function to remove any leading or trailing spaces from character input: `ALLTRIM(str)`. For more information, see "ALLTRIM( ) function" on page 2041.

Output

Datetime.

Examples

Basic examples

Adds 45 minutes to the current date and time:

```
RDATETIME("Sys.time() + value1",2700)
```

Advanced examples

Using an external R script

Adds 45 minutes to a datetime field by passing a field and a literal value to an external R function:

```
RDATETIME("a<-'c:\\scripts\\sample.r');a[[1]]", start_date, 2700)
```

External R script (sample.r):

```
add_time <- function(start, sec) {
    return(start + sec)
}
add_time(value1, value2)
```

Remarks

Returning data from R

When calling R scripts, use the `source` function and assign the return object to a variable. You can then access the value returned from your R function from the return object:

```
# 'a' holds the response object and a[[1]] access the data value
"a<-source('c:\\scripts\\r_scripts\\sample.r');a[[1]]"
```

R log file

Analytics logs R language messages to an `aclrlang.log` file in the project folder. Use this log file for debugging R errors.

> **Tip**
> The log file is available in the Results folder of Analytics Exchange analytic jobs.

Running external R scripts on AX Server

If you are writing an analysis app to run on AX Server and you want to work with external R scripts:

1. Upload the file as a related file with the analysis app.
2. Use the `FILE` analytic tag to identify the file(s).
3. Reference the file(s) using the relative path `./filename.r`.

> **Note**
> Using a related file ensures that the TomEE application server account has sufficient permissions to access the file when running R with Analytics Exchange.

System time zone

Greenwich Mean Time (GMT) is the default current time zone in the R environment used by Analytics.

RECLEN( ) function

Returns the length of the current record.

Syntax

```
RECLEN( )
```

Parameters

This function does not have any parameters.

Output

Numeric.

Examples

Basic examples

The following example extracts all records where the length is exactly 110:

```
EXTRACT RECORD IF RECLEN( ) = 110 TO "Extract.fil"
```

Remarks

You can use the RECLEN( ) function to identify to records of a particular length, or to test for shorter than expected records. This function is useful if you are working with Print Image (Report) files because it provides an easy way to examine the record lengths:

- For fixed-length records, the return value is a constant value (the record length).
- For variable-length records, the return value changes for each record.

RECNO( ) function

Returns the current record number.

Syntax

```
RECNO( )
```

Parameters

This function does not have any parameters.

Output

Numeric.

Examples

Basic examples

The following example extracts records numbered 10 through 20 to a new Analytics table:

```
EXTRACT RECORD IF BETWEEN(RECNO( ),10,20) TO "Subset.fil"
```

Remarks

You can use the RECNO( ) function to output record numbers to a table, or to determine the relative location of a particular record within a table.

Indexed tables vs Non-indexed tables

This function returns the current logical record number:

- If the table is not indexed, RECNO( ) starts with a value of 1 and increases by one for each record in the table. The logical and physical record numbers are identical.
- If the table is indexed, RECNO( ) behaves similarly, but counts the records using the logical, not physical order.

Using the SEEK or FIND command

If the SEEK or FIND commands are used, the record number is reset to 1 after you execute these commands.

Reordering records

When you reorder the records in a table, the record numbers generated by RECNO( ) are not reordered. To keep the record numbers with the records that they were originally associated with, extract the data to a new table using the **Fields** option before you reorder the records.

RECOFFSET( ) function

Returns a field value from a record that is a specified number of records from the current record.

Syntax

```
RECOFFSET(field, number_of_records)
```

Parameters

Name	Type	Description
field	character numeric datetime	The name of the field to retrieve the value from.
number_of_records	numeric	The number of records from the current record. A positive number specifies a record after the current record, and a negative number specifies a record before the current record.

Output

Character, Numeric, or Datetime. The return value belongs to the same data category as the input *field* parameter.

Examples

Basic examples

Returns an *Amount* value from the next record:

```
RECOFFSET(Amount,1)
```

Returns an *Amount* value from the previous record:

```
RECOFFSET(Amount, -1)
```

Advanced examples

Using RECOFFSET in a computed field

The computed field *Next_Amount* shows the value of the Amount field in the next record only if the next record has the same customer number.

To define this computed field in a script, use the following syntax:

```
DEFINE FIELD Next_Amount COMPUTED
RECOFFSET(Amount,1) IF RECOFFSET(Customer,1) = Customer
0
```

Next_Amount is the value of the next record's Amount field only if the customer number in the next record is the same as the customer number in the current record. Otherwise, *Next_Amount* is assigned a value of zero.

Remarks

The RECOFFSET( ) function returns a field value from a record that is a specified number of records above or below the current record.

When to use RECOFFSET( )

This function is commonly used for advanced comparison testing.

You can use this function to compare values in a field in the current record with a field in another record. For example, you can add a computed field that calculates the difference between an amount in the current record and an amount in the previous record.

The beginning or end of a table

If the beginning or end of the table is encountered, the function returns zero for numeric fields, a blank string for character fields, or 1900/01/01 for date fields. The function returns blank output in these instances because there is no further record to compare the current record to.

REGEXFIND( ) function

Returns a logical value indicating whether the pattern specified by a regular expression occurs in a string.

Syntax

```
REGEXFIND(string, pattern)
```

Parameters

Name	Type	Description
string	character	The field, expression, or literal value to test for a matching pattern.
pattern	character	The pattern string (regular expression) to search for. *pattern* can contain literal characters, metacharacters, or a combination of the two. Literal characters include all alphanumeric characters, some punctuation characters, and blanks. The search is case-sensitive, which means that uppercase and lowercase alpha characters must be explicitly specified.

Output

Logical. Returns **T** (true) if the specified *pattern* value is found, and **F** (false) otherwise.

Examples

Basic examples

Alpha character patterns

Returns T for all records that contain "Phoenix", "Austin", or "Los Angeles" in the **Vendor_City** field. Returns F otherwise:

```
REGEXFIND(Vendor_City, "Phoenix|Austin|Los Angeles")
```

Returns T for all last names that start with "John" or "Jon". For example: John, Jon, Johnson, Johnston, Jonson, Jonston, Jones, and so on. Returns F otherwise:

```
REGEXFIND(Last_Name,"^Joh?n")
```

Returns T for only those last names that are "John" or "Jon". Returns F otherwise:

```
REGEXFIND(Last_Name,"^Joh?n\b")
```

Numeric character patterns

Returns T for all records with invoice numbers that contain "98". Returns F otherwise:

```
REGEXFIND(Invoice_Number, "98")
```

Returns T for all records with invoice numbers that begin with "98". Returns F otherwise:

```
REGEXFIND(Invoice_Number, "\b98")
```

Returns T for all records with invoice numbers that end with "98". Returns F otherwise:

```
REGEXFIND(Invoice_Number, "98\b")
```

Returns T for all records with invoice numbers that contain "98" in the 5th and 6th positions. Returns F otherwise:

```
REGEXFIND(Invoice_Number, "\b\d\d\d\d98")
```

```
REGEXFIND(Invoice_Number, "\b\d{4}98")
```

Mixed character patterns

Returns T for all records with product codes that start with 3 numbers, followed by a hyphen and 6 letters. Returns F otherwise:

```
REGEXFIND(Product_Code, "\b\d{3}-[a-zA-Z]{6}\b")
```

Returns T for all records with product codes that start with 3 or more numbers, followed by a hyphen and 6 or more letters. Returns F otherwise:

```
REGEXFIND(Product_Code, "\b\d{3,}-[a-zA-Z]{6}")
```

Returns T for all records with alphanumeric invoice identifiers that contain "98" in the 5th and 6th positions. Returns F otherwise:

```
REGEXFIND(Invoice_Number, "\b\w{4}98")
```

Returns T for all records with invoice identifiers that contain both of the following, otherwise returns F:

- any character in the first four positions
- "98" in the 5th and 6th positions

```
REGEXFIND(Invoice_Number, "\b.{4}98")
```

Returns T for all records with invoice identifiers that contain "98" preceded by 1 to 4 initial characters. Returns F otherwise:

```
REGEXFIND(Invoice_Number, "\b.{1,4}98")
```

Returns 'T' for all records with invoice identifiers that contain all of the following, otherwise returns F:

- any character in the first three positions
- "5" or "6" in the 4th position
- "98" in the 5th and 6th positions

```
REGEXFIND(Invoice_Number, "\b.{3}[56]98")
```

Returns T for all records with invoice identifiers that contain all of the following, otherwise returns F:

- any character in the first two positions
- "55" or "56" in the 3rd and 4th positions
- "98" in the 5th and 6th positions

```
REGEXFIND(Invoice_Number, "\b.{2}(55|56)98")
```

Remarks

How it works

The REGEXFIND( ) function uses a regular expression to search data in Analytics.

Regular expressions are powerful and flexible search strings that combine literal characters and metacharacters, which are special characters that perform a wide variety of search operations.

For example:

```
REGEXFIND(Last_Name,"Sm(i|y)the{0,1}")
```

uses the group `( )`, alternation `|`, and quantifier `{ }` metacharacters to create a regular expression that finds "Smith", "Smyth", "Smithe", or "Smythe" in the **Last_Name** field.

Matching performed sequentially

Matching between *string* and *pattern* values is performed sequentially. In the example above:

- "S" is matched against the first position in the **Last_Name** field
- "m" is matched against the second position
- "i" and "y" are matched against the third position
- "t" is matched against the fourth position
- "h" is matched against the fifth position
- "e" is matched against the sixth position, if a sixth position exists in the source value

When to use REGEXFIND( )

Use REGEXFIND( ) to search data using simple or complex pattern matching.

Constructing regular expressions can be tricky, especially if you are new to the syntax. You may be able to achieve your search goals using simpler Analytics search functions such as FIND( ), MATCH ( ), or MAP( ).

If your search requirements exceed the capabilities of these simpler functions, regular expressions give you almost unlimited flexibility in constructing search strings.

How REGEXFIND( ) handles spaces

Spaces (blanks) are treated as characters in both *string* and *pattern*, so you should exercise care when dealing with spaces.

In *pattern*, you can indicate a space either literally, by typing a space, or by using the metacharacter `\s`. Using the metacharacter makes spaces easier to read in *pattern*, and therefore less likely to be overlooked, especially when you construct more complex patterns.

Concatenating fields

You can concatenate two or more fields in *string* if you want to search across multiple fields simultaneously.

For example:

```
REGEXFIND(Vendor_Name+Vendor_Street,"Hardware.*Main")
```

searches both the **Vendor_Name** and the **Vendor_Street** fields for the words "Hardware" and "Main" separated by zero or more characters.

A business with the word "Hardware" in its name, located on a street called "Main", matches the regular expression. So does a business called "Hardware on Main".

The concatenated fields are treated like a single field that includes leading and trailing spaces from the individual fields, unless you use the ALLTRIM( ) function to remove spaces.

Order of concatenated fields matters

Because REGEXFIND( ) searches for the characters in *pattern* in the order in which you specify them, the order in which you concatenate the fields has an effect. If you reversed **Vendor_Name** and **Vendor_Street** in the expression above, you would be less likely to get any results.

Regular expression metacharacters

The table below lists metacharacters you can use with REGEXFIND( ) and REGEXREPLACE( ) and describes the operation that each one performs.

Additional regular expression syntax exists, and is supported by Analytics, but it is more complex. A full explanation of additional syntax is beyond the scope of this guide. Numerous resources explaining regular expressions are available on the Internet.

Analytics uses the **ECMAScript** implementation of regular expressions. Most implementations of regular expressions use a common core syntax.

> **Note**
> The current implementation of regular expressions in Analytics does not fully support searching languages other than English.

Metacharacter	Description
.	Matches any character (except a new line character)
?	Matches 0 or 1 occurrences of the immediately preceding literal, metacharacter, or element
*	Matches 0 or more occurrences of the immediately preceding literal, metacharacter, or element
+	Matches 1 or more occurrences of the immediately preceding literal, metacharacter, or element
{}	Matches the specified number of occurrences of the immediately preceding literal, metacharacter, or element. You can specify an exact number, a range, or an open-ended range. For example: ◦ **a{3}** matches "aaa" ◦ **X{0,2}L** matches "L", "XL", and "XXL" ◦ **AB-\d{2,}-YZ** matches any alphanumeric identifier with the prefix "AB-", the suffix "-YZ", and two or more numbers in the body of the identifier
[]	Matches any single character inside the brackets For example: ◦ **[aeiou]** matches a, or e, or i, or o, or u ◦ **[^aeiou]** matches any character that is not a, or e, or i, or o, or u ◦ **[A-G]** matches any uppercase letter from A to G ◦ **[A-Ga-g]** matches any uppercase letter from A to G, or any lowercase letter from a to g ◦ **[5-9]** matches any number from 5 to 9
()	Creates a group that defines a sequence or block of characters, which can then be treated as a single unit. For example: ◦ **S(ch)?mid?th?** matches "Smith" or "Schmidt" ◦ **(56A.*){2}** matches any alphanumeric identifier in which the sequence "56A" occurs at least twice ◦ **(56A).*-.*\1** matches any alphanumeric identifier in which the sequence "56A" occurs at least twice, with a hyphen located between two of the occurrences
\	An escape character that specifies that the character immediately following is a literal. Use the escape character if you want to literally match metacharacters. For example, **\(** finds a left parenthesis, and **** finds a backslash. Use the escape character if you want to literally match any of the following characters: **^ $. * + ? = ! : | \ ( ) [ ] { }** Other punctuation characters such as the ampersand (&) or the 'at sign' (@) do not require the escape character.

Metacharacter	Description
int	Specifies that a group, previously defined with parentheses **( )**, recurs. *int* is an integer that identifies the sequential position of the previously defined group in relation to any other groups. This metacharacter can be used in the *pattern* parameter in both REGEXFIND( ) and REGEXREPLACE( ). For example: ◦ **(123).*\1** matches any identifier in which the group of digits "123" occurs at least twice ◦ **^(\d{3}).*\1** matches any identifier in which the first 3 digits recur ◦ **^(\d{3}).*\1.*\1** matches any identifier in which the first 3 digits recur at least twice ◦ **^(\D)(\d)-.*\2\1** matches any identifier in which the alphanumeric prefix recurs with the alpha and numeric characters reversed
$*int*	Specifies that a group found in a target string is used in a replacement string. *int* is an integer that identifies the sequential position of the group in the target string in relation to any other groups. This metacharacter can only be used in the *new_string* parameter in REGEXREPLACE( ). For example: ◦ If the pattern **(\d{3})[-]?(\d{3})[-]?(\d{4})** is used to match a variety of different telephone number formats, the *new_string***($1)-$2-$3** can be used to replace the numbers with themselves, and standardize the formatting. 999 123-4567 and 9991234567 both become (999)-123-4567.
|	Matches the character, block of characters, or expression before or after the pipe (**|**) For example: ◦ **a|b** matches a or b ◦ **abc|def** matches "abc" or "def" ◦ **Sm(i|y)th** matches Smith or Smyth ◦ **[a-c]|[Q-S]|[x-z]** matches any of the following letters: a, b, c, Q, R, S, x, y, z ◦ **\s|-** matches a space or a hyphen
\w	Matches any word character (a to z, A to Z, 0 to 9, and the underscore character _)
\W	Matches any non-word character (not a to z, A to Z, 0 to 9, or the underscore character _)
\d	Matches any number (any decimal digit)
\D	Matches any non-number (any character that is not a decimal digit)
\s	Matches a space (a blank)
\S	Matches any non-space (a non-blank character)
\b	Matches a word boundary (between \w and \W characters) Word boundaries consume no space themselves. For example: ◦ "United Equipment" contains 4 word boundaries - one either side of the space, and one

Metacharacter	Description
	at the start and the end of the string. "United Equipment" is matched by the regular expression **\b\w*\b\W\b\w*\b** **Tip** In addition to spaces, word boundaries can result from commas, periods, and other non-word characters. For example, the following expression evaluates to True: `REGEXFIND("jsmith@example.net", "\bexample\b")`
^	Matches the start of a string Inside brackets **[ ]**, **^** negates the contents
$	Matches the end of a string

Related functions

If you want to find and replace matching patterns, use the "REGEXREPLACE( ) function" on the facing page.

REGEXREPLACE( ) function

Replaces all instances of strings matching a regular expression with a new string.

Syntax

```
REGEXREPLACE(string, pattern, new_string)
```

Parameters

Name	Type	Description
string	character	The field, expression, or literal value to test for a matching pattern.
pattern	character	The pattern string (regular expression) to search for. *pattern* can contain literal characters, metacharacters, or a combination of the two. Literal characters include all alphanumeric characters, some punctuation characters, and blanks. The search is case-sensitive, which means that uppercase and lowercase alpha characters must be explicitly specified.
new_string	character	The string to use to replace all values matching *pattern*. The replacement string can contain literal characters, groups of characters from the original string (using the `$int` element), or a combination of the two.

Output

Character.

Examples

Basic examples

Working with spaces

Returns "AB CD EF", by replacing multiple spaces between text characters with a single space:

```
REGEXREPLACE("AB CD   EF", "\s+", " ")
```

Returns the character field data with the spacing between words standardized on a single space:

```
REGEXREPLACE(character_field, "\s+", " ")
```

Returns the character field data with the spacing between words standardized on a single space. Using the BLANKS( ) function in *new_string*, rather than a literal space, makes spaces easier to read and less likely to be overlooked:

```
REGEXREPLACE(character_field, "\s+", BLANKS(1))
```

Standardizing telephone numbers

Returns "(123) 456-7890". The formatting of the telephone number '1234567890' is standardized:

```
REGEXREPLACE(SUBSTR("1234567890",1,14), "(\d{3})[\s-]*(\d{3})[\s-]*(\d{4})","
($1) $2-$3")
```

Returns the numbers in the **Telephone_Number** field and standardizes their formatting:

```
REGEXREPLACE(Telephone_Number, ".*(\d{3})[\s-\.\)]*(\d{3})[\s-\.]*(\d{4})", "
($1) $2-$3")
```

Extracts "123-456-7890" from the surrounding text:

```
REGEXREPLACE("Tel num: 123-456-7890 (office)", "(.*)(\d{3}[\s-\)\.]*\d{3}[\s-
\.]*\d{4})(.*)", "$2")
```

Extracts telephone numbers from surrounding text in the **Comment** field and standardizes their formatting:

```
REGEXREPLACE(Comment, "(.*)(\d{3})[\s-\)\.]*(\d{3})[\s-\.]*(\d{4})(.*)","($2)
$3-$4")
```

Identifying generic formats

Returns "9XXX-999xx", which represents the generic format of the value specified by *string* ("1ABC-123aa"):

```
REGEXREPLACE(REGEXREPLACE(REGEXREPLACE("1ABC-123aa","\d","9"),"[a-z]","x"),"
[A-Z]", "X")
```

Returns the generic format of all identifiers in the **Invoice_Number** field:

```
REGEXREPLACE(REGEXREPLACE(REGEXREPLACE(Invoice_Number,"\d","9"),"[a-z]","x"),"
[A-Z]", "X")
```

Standardizing name format

Returns "John David Smith":

```
REGEXREPLACE("Smith, John David", "^(\w+),(\s\w+)(\s\w+)?(\s\w+)?","$2$3$4
$1")
```

Returns the names in the **Full_Name** field in their regular order: *First* (*Middle*) (*Middle*) *Last*:

```
REGEXREPLACE(Full_Name, "^(\w+),(\s\w+)(\s\w+)?(\s\w+)?","$2$3$4 $1")
```

> **Note**
> Name data can present various complications, such as apostrophes in names. Accounting for variations in name data typically requires more complex regular expressions than the one provided in the example above.

Removing HTML markup

Returns "https://www.flgov.com/wp-content/uploads/orders/2020/EO_20-166.pdf":

```
REGEXREPLACE("<a href='https://www.flgov.com/wp-con-
tent/uploads/orders/2020/EO_20-166.pdf' tar-
get='blank'>https://www.flgov.com/wp-content/uploads/orders/2020/EO_20-
166.pdf</a>", "<[^>]*>",' ')
```

Returns the hyperlinks in the **Source_URL_Link** field with the HTML markup removed:

```
REGEXREPLACE(Source_URL_Link, "<[^>]*>",' ')
```

Remarks

How it works

The REGEXREPLACE( ) function uses a regular expression to find matching patterns in data, and replaces any matching values with a new string.

For example:

```
REGEXREPLACE(character_field, "\s+", " ")
```

standardizes spacing in character data by replacing one or more spaces between text characters with a single space.

The search portion of REGEXREPLACE( ) is identical to the REGEXFIND( ) function. For detailed information about the search capability common to both functions, see "REGEXFIND( ) function" on page 2327.

When to use REGEXREPLACE( )

Use REGEXREPLACE( ) any time you want to find and replace data in Analytics using simple or complex pattern matching.

Replacing characters with themselves

You can use the `$int` element to replace characters with themselves, which allows you to preserve the meaningful parts of data, while standardizing or omitting surrounding or intermixed data.

Several examples using telephone numbers and names appear above.

To use the `$int` element you must first create groups by using parentheses `( )` in the *pattern* value. For more information, see "REGEXFIND( ) function" on page 2327.

Avoiding sequential character matching

You can avoid sequential character matching, and replace substrings regardless of their position in relation to one another, by nesting REGEXREPLACE( ) functions.

The problem in the two examples below is to derive a generic format from alphanumeric source data in which numbers and letters can appear in any order. Without knowing the order of numbers and letters, how do you construct the *pattern* string?

The solution is to first find and replace numbers using the inner REGEXREPLACE( ) function, and then find and replace letters using the outer REGEXREPLACE( ) function.

Returns "999XXX":

```
REGEXREPLACE(REGEXREPLACE("123ABC","\d","9"),"[A-Z]","X")
```

Returns "9X9X9X":

```
REGEXREPLACE(REGEXREPLACE("1A2B3C","\d","9"),"[A-Z]","X")
```

Replacement string length and truncation

When you use REGEXREPLACE( ) to create a computed field, the computed field length is identical to the original field length.

If the replacement string length exceeds the target string length, overall string length increases, which results in truncation if the computed field length cannot accommodate the increased string length.

Characters that trail the target string are truncated first, followed by trailing replacement string characters. The examples below illustrate truncation:

string	*pattern*	*new_string*	Field length	Result	Truncated characters
x123x	123	A	5	"xAx"	none
x123x	123	ABC	5	"xABCx"	none
x123x	123	ABCD	5	"xABCD"	"x"
x123x	123	ABCDE	5	"xABCD"	"x", "E"
x123x	123	ABCDE	6	"xABCDE"	"x"
x123x	123	ABCDE	7	"xABCDEx"	none

How to avoid truncation

To avoid truncation, use the SUBSTR( ) function to increase field length, as demonstrated by the second example below.

Returns "xABCD", which truncates the replacement character "E" and the existing character "x":

```
REGEXREPLACE("x123x","123","ABCDE")
```

Returns "xABCDEx", which includes all replacement characters and unreplaced existing characters:

```
REGEXREPLACE(SUBSTR("x123x",1,10),"123","ABCDE")
```

Regular expression metacharacters

The table below lists metacharacters you can use with REGEXFIND( ) and REGEXREPLACE( ) and describes the operation that each one performs.

Additional regular expression syntax exists, and is supported by Analytics, but it is more complex. A full explanation of additional syntax is beyond the scope of this guide. Numerous resources explaining regular expressions are available on the Internet.

Analytics uses the **ECMAScript** implementation of regular expressions. Most implementations of regular expressions use a common core syntax.

> **Note**
> The current implementation of regular expressions in Analytics does not fully support searching languages other than English.

Metacharacter	Description
.	Matches any character (except a new line character)
?	Matches 0 or 1 occurrences of the immediately preceding literal, metacharacter, or element
*	Matches 0 or more occurrences of the immediately preceding literal, metacharacter, or element
+	Matches 1 or more occurrences of the immediately preceding literal, metacharacter, or element
{}	Matches the specified number of occurrences of the immediately preceding literal, metacharacter, or element. You can specify an exact number, a range, or an open-ended range. For example:

Metacharacter	Description
	◦ **a{3}** matches "aaa" ◦ **X{0,2}L** matches "L", "XL", and "XXL" ◦ **AB-\d{2,}-YZ** matches any alphanumeric identifier with the prefix "AB-", the suffix "-YZ", and two or more numbers in the body of the identifier
[]	Matches any single character inside the brackets For example: ◦ **[aeiou]** matches a, or e, or i, or o, or u ◦ **[^aeiou]** matches any character that is not a, or e, or i, or o, or u ◦ **[A-G]** matches any uppercase letter from A to G ◦ **[A-Ga-g]** matches any uppercase letter from A to G, or any lowercase letter from a to g ◦ **[5-9]** matches any number from 5 to 9
()	Creates a group that defines a sequence or block of characters, which can then be treated as a single unit. For example: ◦ **S(ch)?mid?th?** matches "Smith" or "Schmidt" ◦ **(56A.*){2}** matches any alphanumeric identifier in which the sequence "56A" occurs at least twice ◦ **(56A).*-.*\1** matches any alphanumeric identifier in which the sequence "56A" occurs at least twice, with a hyphen located between two of the occurrences
****	An escape character that specifies that the character immediately following is a literal. Use the escape character if you want to literally match metacharacters. For example, **\(** finds a left parenthesis, and **** finds a backslash. Use the escape character if you want to literally match any of the following characters: **^ $. * + ? = ! : | \ ( ) [ ] { }** Other punctuation characters such as the ampersand (&) or the 'at sign' (@) do not require the escape character.
int	Specifies that a group, previously defined with parentheses **( )**, recurs. *int* is an integer that identifies the sequential position of the previously defined group in relation to any other groups. This metacharacter can be used in the *pattern* parameter in both REGEXFIND( ) and REGEXREPLACE( ). For example: ◦ **(123).*\1** matches any identifier in which the group of digits "123" occurs at least twice ◦ **^(\d{3}).*\1** matches any identifier in which the first 3 digits recur ◦ **^(\d{3}).*\1.*\1** matches any identifier in which the first 3 digits recur at least twice ◦ **^(\D)(\d)-.*\2\1** matches any identifier in which the alphanumeric prefix recurs with the alpha and numeric characters reversed
$*int*	Specifies that a group found in a target string is used in a replacement string. *int* is an integer that identifies the sequential position of the group in the target string in relation to

Metacharacter	Description
	any other groups. This metacharacter can only be used in the *new_string* parameter in REGEXREPLACE( ). For example: ◦ If the pattern **(\d{3})[-]?(\d{3})[-]?(\d{4})** is used to match a variety of different telephone number formats, the *new_string***($1)-$2-$3** can be used to replace the numbers with themselves, and standardize the formatting. 999 123-4567 and 9991234567 both become (999)-123-4567.
|	Matches the character, block of characters, or expression before or after the pipe (**|**) For example: ◦ **a|b** matches a or b ◦ **abc|def** matches "abc" or "def" ◦ **Sm(i|y)th** matches Smith or Smyth ◦ **[a-c]|[Q-S]|[x-z]** matches any of the following letters: a, b, c, Q, R, S, x, y, z ◦ **\s|-** matches a space or a hyphen
\w	Matches any word character (a to z, A to Z, 0 to 9, and the underscore character _)
\W	Matches any non-word character (not a to z, A to Z, 0 to 9, or the underscore character _)
\d	Matches any number (any decimal digit)
\D	Matches any non-number (any character that is not a decimal digit)
\s	Matches a space (a blank)
\S	Matches any non-space (a non-blank character)
\b	Matches a word boundary (between \w and \W characters) Word boundaries consume no space themselves. For example: ◦ "United Equipment" contains 4 word boundaries - one either side of the space, and one at the start and the end of the string. "United Equipment" is matched by the regular expression **\b\w*\b\W\b\w*\b** **Tip** In addition to spaces, word boundaries can result from commas, periods, and other non-word characters. For example, the following expression evaluates to True: `REGEXFIND("jsmith@example.net", "\bexample\b")`
^	Matches the start of a string Inside brackets **[ ]**, **^** negates the contents

Metacharacter	Description
$	Matches the end of a string

Related functions

If you want to find matching patterns without replacing them, use the "REGEXFIND( ) function" on page 2327.

REMOVE( ) function

Returns a string that includes only the specified characters.

Syntax

```
REMOVE(string, valid_characters)
```

Parameters

Name	Type	Description
string	character	The field, expression, or literal value to remove characters from.
valid_characters	character	The characters to retain in *string*. If you specify double quotation marks in *valid_characters*, you must enclose the list of characters in single quotation marks. For example: `'"-/'` **Note** If a character you specify does not appear in *string*, it is not included in the return value.

Output

Character.

Examples

Basic examples

Returns "ABC123 ":

```
REMOVE("ABC 123 XX4","ABC123")
```

Returns "ABC123XX ":

```
REMOVE("zABC 123 XX4","ABCX123")
```

Returns "1234 ":

```
REMOVE("ABC 123 XX4", "1234567890")
```

Returns all the values in the **Product_Number** field with all non-numeric characters removed:

```
REMOVE(Product_Number,"0123456789")
```

Remarks

> **Note**
> The REMOVE( ) function has been superseded by the INCLUDE( ) and EXCLUDE( ) functions.
> REMOVE( ) is still available in the current version of Analytics for backwards compatibility with earlier versions.

How it works

The REMOVE( ) function removes unwanted characters from character data and returns a fixed length string.

When to use REMOVE( )

Use REMOVE( ) to normalize data fields that do not have a consistent format, such as address fields. You can also use REMOVE( ) to remove punctuation or other invalid information from poorly edited fields.

You can also use the function to clean data in fields before using the SORT or JOIN commands, for duplicate matching, or for report output.

Case sensitivity

The REMOVE( ) function is case-sensitive. If you specify "ID" in *valid_characters*, these characters are not included in "id#94022". If there is a chance the case may be mixed, use the UPPER( ) function to convert *string* to uppercase.

For example:

```
REMOVE(UPPER("id#94022"), "ID0123456789")
```

Related functions

REMOVE( ) is similar to the INCLUDE( ) function, with the following difference:

- REMOVE( ) adds blanks to the end of the output to replace the characters that have been removed. The original length of *string* is retained.
- INCLUDE( ) does not add any blanks.

REPEAT( ) function

Returns a string that repeats a substring a specified number of times.

Syntax

```
REPEAT(string, count)
```

Parameters

Name	Type	Description
string	character	The field, expression, or literal value to repeat.
count	numeric	The number of times to repeat the value of *string*.

Output

Character.

Examples

Basic examples

Returns "ABCABCABC":

```
REPEAT("ABC",3)
```

Returns "000000000":

```
REPEAT("0",9)
```

Remarks

When to use REPEAT( )

Use the REPEAT( ) function to initialize a variable with constant values or blanks, or to set a default value for a computed field.

REPLACE( ) function

Replaces all instances of a specified character string with a new character string.

Syntax

```
REPLACE(string, old_text, new_text)
```

Parameters

Name	Type	Description
string	character	The value to replace characters in.
old_text	character	The character string to replace. The search is case-sensitive.
new_text	character	The text to replace the value in *old_text* with.

Output

Character.

Examples

Basic examples

Returns "a12345efg":

```
REPLACE("abcdefg","bcd","12345")
```

Returns "Rd.":

```
REPLACE("Road","Road","Rd.")
```

Returns "ac":

```
REPLACE("abc","b","")
```

Advanced examples

Removing specified characters

Use REPLACE( ) to remove a specified character string from a source string, by replacing it with an empty character string (`""`).

Returns "1234 Scott":

```
REPLACE("1234 Scott rd.", "rd.", "")
```

Field length adjustment

If *new_text* ("ABC") is longer than *old_text* ("X"), the field length of the resulting string is automatically increased to accommodate the first replacement:

Returns "9ABC9", with a field length increased to 5 characters from 3 characters:

```
REPLACE("9X9", "X", "ABC")
```

The field length is not automatically increased for subsequent replacements, and truncation can result if the field is not long enough to accommodate all the new characters.

Returns "9ABC9A":

```
REPLACE("9X9X", "X", "ABC")
```

To avoid truncation, you can increase the length of *string* using the BLANKS( ) function, or literal blank spaces.

Returns "9ABC9ABC":

```
REPLACE("9X9X" + BLANKS(2), "X", "ABC")
```

```
REPLACE("9X9X" + "  ", "X", "ABC")
```

If the resulting string is shorter than *string*, the resulting string is padded with blanks to maintain the same field length.

Returns "9X9 ":

```
REPLACE("9ABC9", "ABC", "X")
```

Remarks

How it works

The REPLACE( ) function replaces every instance of an existing string with a new string.

Returns "1234 Scott Road":

```
REPLACE("1234 Scott rd.", "rd.", "Road")
```

When to use REPLACE( )

Use REPLACE( ) for normalizing data fields with inconsistent formats, such as address fields, or for replacing invalid information in poorly edited fields. To be performed accurately, operations such as duplicates testing, or joining or relating tables, require data with a normalized or standardized format.

Case sensitivity

The REPLACE( ) function is case-sensitive. If you specify `"RD."` in *old_text* and the values in *string* are lowercase, the *new_text* value will not be substituted because no matches will be found.

If there is a chance the case in *string* may be mixed, first use the UPPER( ) function to convert all characters to uppercase.

Returns "1234 SCOTT ROAD":

```
REPLACE(UPPER("1234 Scott rd."), "RD.", "ROAD")
```

Protecting against inadvertent replacements

When building a REPLACE( ) expression you must be aware of every possible instance of *old_text* in *string* so that you do not get inadvertent replacements.

Returns "645 RichaRoad Road", because the last two letters of "Richard" are "rd":

```
REPLACE("645 Richard rd  ", "rd", "Road")
```

By adding both a leading space and a trailing space to the value in *old_text* (`" rd "`), you prevent the function from replacing instances where "rd" occurs in a name, because in these instances there are no leading spaces.

Returns "645 Richard Road":

```
REPLACE("645 Richard rd  ", " rd ", " Road")
```

REVERSE( ) function

Returns a string with the characters in reverse order.

Syntax

```
REVERSE(string)
```

Parameters

Name	Type	Description
string	character	The field, expression, or literal value to reverse the order of.

Output

Character.

Examples

Basic examples

Returns "E DCBA":

```
REVERSE("ABCD E")
```

RJUSTIFY( ) function

Returns a right-justified string the same length as a specified string, with any trailing spaces moved to the left of the string.

Syntax

```
RJUSTIFY(string)
```

Parameters

Name	Type	Description
string	character	The field, expression, or literal value to right-justify.

Output

Character.

Examples

Basic examples

Returns " ABC":

```
RJUSTIFY("ABC   ")
```

Remarks

When to use RJUSTIFY( )

Use the RJUSTIFY( ) function to right-justify a character field.

RLOGICAL( ) function

Returns a logical value calculated by an R function or script. Data processing in R is external to Analytics.

Syntax

```
RLOGICAL(rScript|rCode <,field|value <,...n>>)
```

Parameters

Name	Type	Description
rScript | *rCode*	character	The full or relative path to the R script, or a snippet of R code to run. If you enter R code directly rather than use an external file, you cannot use the enclosing quotation character in your code, even if you escape it: ◦ **valid** - `'var <- "\"test\"" '` ◦ **invalid** - `'var <- "\'test\'" '`
field | *value* <,...n> optional	character numeric datetime logical	The list of fields, expressions, or literal values to use as arguments for the R script or code snippet. The values are passed into the function you call in the order you specify them, and you reference them using `value1, value2 ... valueN` in the R code. You may include as many arguments as necessary to satisfy the function definition in the R code. **Note** Use the `ALLTRIM()` function to remove any leading or trailing spaces from character input: `ALLTRIM(str)`. For more information, see "ALLTRIM( ) function" on page 2041.

Output

Logical.

Examples

Basic examples

Returns T:

```
RLOGICAL("(value1>0.6) & (value2>0.7) & (value3>0.5)", 0.8, 0.9, 0.55)
```

Advanced examples

Using an external R script

Accepts an amount, and an upper and lower threshold value. The function returns a truth value based on a series of logical comparisons:

```
RLOGICAL("a<-'c:\\scripts\\sample.r');a[[1]]", expense_amt, threshold_
low, threshold_hi)
```

External R script (sample.r):

```
test_truth <- function(amt, low, hi) {
    return(((amt > low) & (amt < hi)) | ((amt==low) | (amt==hi)))
}
test_truth(value1, value2, value3)
```

Using R code stored in a variable

Performs a logical test of three fields using AND logic:

```
v_rcode = "(value1>0.6) & (value2>0.7) & (value3>0.5)"
RLOGICAL(v_rcode, PACKED, MICRO_LONG, ACCPAC)
```

Remarks

Returning data from R

When calling R scripts, use the `source` function and assign the return object to a variable. You can then access the value returned from your R function from the return object:

```
# 'a' holds the response object and a[[1]] access the data value
"a<-source('c:\\scripts\\r_scripts\\sample.r');a[[1]]"
```

R log file

Analytics logs R language messages to an `aclrlang.log` file in the project folder. Use this log file for debugging R errors.

> **Tip**
> The log file is available in the Results folder of Analytics Exchange analytic jobs.

Running external R scripts on AX Server

If you are writing an analysis app to run on AX Server and you want to work with external R scripts:

1. Upload the file as a related file with the analysis app.
2. Use the `FILE` analytic tag to identify the file(s).
3. Reference the file(s) using the relative path `./filename.r`.

> **Note**
> Using a related file ensures that the TomEE application server account has sufficient permissions to access the file when running R with Analytics Exchange.

RNUMERIC( ) function

Returns a numeric value calculated by an R function or script. Data processing in R is external to Analytics.

Syntax

```
RNUMERIC(rScript|rCode, decimals <,field|value <,...n>>)
```

Parameters

Name	Type	Description
rScript | *rCode*	character	The full or relative path to the R script, or a snippet of R code to run. If you enter R code directly rather than use an external file, you cannot use the enclosing quotation character in your code, even if you escape it: ◦ **valid** - `'var <- "\"test\"" '` ◦ **invalid** - `'var <- "\'test\'" '`
decimals	numeric	The number of decimal places to include in the return value. Must be a positive integer.
field | *value* <,...*n*> optional	character numeric datetime logical	The list of fields, expressions, or literal values to use as arguments for the R script or code snippet. The values are passed into the function you call in the order you specify them, and you reference them using `value1, value2 ... valueN` in the R code. You may include as many arguments as necessary to satisfy the function definition in the R code. **Note** Use the `ALLTRIM()` function to remove any leading or trailing spaces from character input: `ALLTRIM(str)`. For more information, see "ALLTRIM( ) function" on page 2041.

Output

Numeric.

Examples

Basic examples

Returns 100 with 10 decimals (100.0000000000):

```
RNUMERIC("print(value1)", 10, 100)
```

Advanced examples

Storing R code as a variable

Returns 100 with 10 decimals (100.0000000000):

```
ASSIGN v_rcode = "print(value1)"
RNUMERIC(v_rcode, 10, 100)
```

Writing to an external file

Performs a simple addition and writes the comment attached to the function to file using the `sink` function in R:

```
RNUMERIC("foo<-function(x,y){x+y};attr(foo, 'comment') <- 'foo performs
simple addition';sink('c:/temp/result.txt');attributes(foo);sink(NULL);-
foo(value1,value2)",0, amt, gross)
```

Remarks

Returning data from R

When calling R scripts, use the `source` function and assign the return object to a variable. You can then access the value returned from your R function from the return object:

```
# 'a' holds the response object and a[[1]] access the data value
"a<-source('c:\\scripts\\r_scripts\\sample.r');a[[1]]"
```

R log file

Analytics logs R language messages to an `aclrlang.log` file in the project folder. Use this log file for debugging R errors.

> **Tip**
> The log file is available in the Results folder of Analytics Exchange analytic jobs.

Running external R scripts on AX Server

If you are writing an analysis app to run on AX Server and you want to work with external R scripts:

1. Upload the file as a related file with the analysis app.
2. Use the `FILE` analytic tag to identify the file(s).
3. Reference the file(s) using the relative path `./filename.r`.

> **Note**
> Using a related file ensures that the TomEE application server account has sufficient permissions to access the file when running R with Analytics Exchange.

ROOT( ) function

Returns the square root of a numeric expression.

Syntax

```
ROOT(number, decimals)
```

Parameters

Name	Type	Description
number	numeric	The numeric expression to find the square root of. This function returns zero if *number* is a negative number.
decimals	numeric	The number of decimals to use in the output.

Output

Numeric.

Examples

Basic examples

Returns 10.00:

```
ROOT(100, 2)
```

Returns 31.6228:

```
ROOT(1000, 4)
```

Remarks

How it works

The ROOT( ) function returns the square root of the numeric expression or field value with the specified number of decimal places. The result is rounded appropriately.

When to use ROOT( )

Use LOG( ) to perform other root functions, such as cube root.

ROUND( ) function

Returns a rounded whole number for a numeric value.

Syntax

```
ROUND(number)
```

Parameters

Name	Type	Description
number	numeric	The value to round to the nearest integer.

Output

Numeric.

Examples

Basic examples

Returns 7:

```
ROUND(7.2)
```

Returns 8:

```
ROUND(7.5)
```

Returns -8:

```
ROUND(-7.5)
```

Advanced examples

Rounding monetary values

Creates a field that is equal to the balance rounded to the nearest dollar value:

```
DEFINE FIELD Nearest_dollar_value COMPUTED ROUND(Balance)
```

Remarks

How it works

ROUND( ) returns a number equal to the *number* value rounded to the nearest integer:

	Positive values	Negative values
Rounds up to the next integer	>= 0.5	< 0.5
Rounds down to the next integer	< 0.5	>= 0.5

Rounding to a particular number of decimal places

If you want to round a number to a particular number of decimal places, use the "DEC( ) function" on page 2111. The ROUND( ) function is the same as the DEC( ) function with zero decimal places specified.

```
ROUND(number)
```

is equivalent to:

```
DEC(number, 0)
```

RSTRING( ) function

Returns a string value calculated by an R function or script. Data processing in R is external to Analytics.

Syntax

```
RSTRING(rScript|rCode, length <,field|value <,...n>>)
```

Parameters

Name	Type	Description
rScript | *rCode*	character	The full or relative path to the R script, or a snippet of R code to run. If you enter R code directly rather than use an external file, you cannot use the enclosing quotation character in your code, even if you escape it: ◦ **valid** - `'var <- "\"test\"" '` ◦ **invalid** - `'var <- "\'test\'" '`
length	numeric	The length to allocate for the return string.
field | *value* <,...*n*> optional	character numeric datetime logical	The list of fields, expressions, or literal values to use as arguments for the R script or code snippet. The values are passed into the function you call in the order you specify them, and you reference them using `value1, value2 ... valueN` in the R code. You may include as many arguments as necessary to satisfy the function definition in the R code. **Note** Use the `ALLTRIM()` function to remove any leading or trailing spaces from character input: `ALLTRIM(str)`. For more information, see "ALLTRIM( ) function" on page 2041.

Output

Character.

Examples

Basic examples

Returns "abc123":

```
RSTRING("print(paste(value1,value2,sep=""))",6,"abc","123")
```

Advanced examples

Using an external R script

Concatenates x and y into a single string delimited by a space character:

```
RSTRING("a<-source('./sample.r');a[[1]]",50, FirstName, LastName)
```

External R script (sample.r):

```
conc <- function(x, y) {
    paste(x, y, sep=" ")
}
print(conc(value1, value2))
```

Using R code stored in a variable

Concatenates x and y into a single string delimited by a space character:

```
ASSIGN v_script = "conc <- function(x, y){paste(x, y, sep=' ')};conc
(value1, value2)"
RSTRING(v_script, 50, FirstName, LastName)
```

Using R to generate a UUID for a table

You are preparing a table of exceptions to upload to Results and you require a guaranteed unique identifier for each record. To generate this field, you use the **uuid** package in R to create a unique primary key value for each record:

```
EXTRACT RSTRING("uuid::UUIDgenerate()", 36) AS "id", first_name, last_
name, birthdate TO export_table
```

> **Tip**
> To install the uuid package, open R.exe and execute the following command:
>
> ```
> install.packages("uuid")
> ```

Remarks

Returning data from R

When calling R scripts, use the `source` function and assign the return object to a variable. You can then access the value returned from your R function from the return object:

```
# 'a' holds the response object and a[[1]] access the data value
"a<-source('c:\\scripts\\r_scripts\\sample.r');a[[1]]"
```

R log file

Analytics logs R language messages to an `aclrlang.log` file in the project folder. Use this log file for debugging R errors.

> **Tip**
> The log file is available in the Results folder of Analytics Exchange analytic jobs.

Running external R scripts on AX Server

If you are writing an analysis app to run on AX Server and you want to work with external R scripts:

1. Upload the file as a related file with the analysis app.
2. Use the `FILE` analytic tag to identify the file(s).
3. Reference the file(s) using the relative path `./filename.r`.

> **Note**
> Using a related file ensures that the TomEE application server account has sufficient permissions to access the file when running R with Analytics Exchange.

RTIME( ) function

Returns a time value calculated by an R function or script. Data processing in R is external to Analytics.

Syntax

```
RTIME(rScript|rCode <,field|value <,...n>>)
```

Parameters

Name	Type	Description
rScript | *rCode*	character	The full or relative path to the R script, or a snippet of R code to run. If you enter R code directly rather than use an external file, you cannot use the enclosing quotation character in your code, even if you escape it: ◦ **valid** - `'var <- "\"test\"" '` ◦ **invalid** - `'var <- "\'test\'" '`
field | *value <,...n>* optional	character numeric datetime logical	The list of fields, expressions, or literal values to use as arguments for the R script or code snippet. The values are passed into the function you call in the order you specify them, and you reference them using `value1, value2 ... valueN` in the R code. You may include as many arguments as necessary to satisfy the function definition in the R code. **Note** Use the `ALLTRIM()` function to remove any leading or trailing spaces from character input: `ALLTRIM(str)`. For more information, see "ALLTRIM( ) function" on page 2041.

Output

Datetime.

Examples

Basic examples

Returns `t0545`:

```
RTIME("value1+2700",`t0500`)
```

Advanced examples

Using an external R script

Adds 45 minutes to a time field by passing a field and a literal value to an external R function:

```
RTIME("a<-source('c:\\scripts\\sample.r');a[[1]]", end_time, 2700)
```

External R script (sample.r):

```
add_time <- function(start, sec) {
    return(start + sec)
}
add_time(value1, value2)
```

Remarks

Returning data from R

When calling R scripts, use the `source` function and assign the return object to a variable. You can then access the value returned from your R function from the return object:

```
# 'a' holds the response object and a[[1]] access the data value
"a<-source('c:\\scripts\\r_scripts\\sample.r');a[[1]]"
```

R log file

Analytics logs R language messages to an `aclrlang.log` file in the project folder. Use this log file for debugging R errors.

> **Tip**
> The log file is available in the Results folder of Analytics Exchange analytic jobs.

Running external R scripts on AX Server

If you are writing an analysis app to run on AX Server and you want to work with external R scripts:

1. Upload the file as a related file with the analysis app.
2. Use the `FILE` analytic tag to identify the file(s).
3. Reference the file(s) using the relative path `./filename.r`.

> **Note**
> Using a related file ensures that the TomEE application server account has sufficient permissions to access the file when running R with Analytics Exchange.

System time zone

Greenwich Mean Time (GMT) is the default current time zone in the R environment used by Analytics.

SECOND( ) function

Extracts the seconds from a specified time or datetime and returns it as a numeric value.

Syntax

```
SECOND(time/datetime)
```

Parameters

Name	Type	Description
time/datetime	datetime	The field, expression, or literal value to extract the seconds from.

Output

Numeric.

Examples

Basic examples

Returns 30:

```
SECOND(`t235930`)
```

```
SECOND(`20141231 235930`)
```

Returns the seconds for each value in the **Call_start_time** field:

```
SECOND(Call_start_time)
```

Remarks

Parameter details

A field specified for *time/datetime* can use any time or datetime format, as long as the field definition correctly defines the format.

Specifying a literal time or datetime value

When specifying a literal time or datetime value for *time/datetime*, you are restricted to the formats in the table below, and you must enclose the value in backquotes – for example, `` `20141231 235959` ``.

Do not use any separators such as slashes (/) or colons (:) between the individual components of dates or times.

- **Time values** – you can use any of the time formats listed in the table below. You must use a separator before a standalone time value for the function to operate correctly. Valid separators are the letter 't', or the letter 'T'. You must specify times using the 24-hour clock. Offsets from Coordinated Universal Time (UTC) must be prefaced by a plus sign (+) or a minus sign (-).
- **Datetime values** – you can use any combination of the date, separator, and time formats listed in the table below. The date must precede the time, and you must use a separator between the two. Valid separators are a single blank space, the letter 't', or the letter 'T'.

Example formats	Example literal values
thhmmss	`t235959`
Thhmm	`T2359`
YYYYMMDD hhmmss	`20141231 235959`
YYMMDDthhmm	`141231t2359`
YYYYMMDDThh	`20141231T23`
YYYYMMDD hhmmss+/-hhmm (UTC offset)	`20141231 235959-0500`
YYMMDD hhmm+/-hh (UTC offset)	`141231 2359+01`
Note Do not use hh alone in the main time format with data that has a UTC offset. For example, avoid: hh+hhmm. Results can be unreliable.	

SHIFT( ) function

Returns a single character string with the bits in the first character of the input value shifted to the left or right.

Syntax

```
SHIFT(character, number_of_bits_to_left)
```

Parameters

Name	Type	Description
character	character	The value to shift bits for.
number_of_bits_to_left	numeric	Specifies the number of bits to shift the *character* value. ◦ **If the value is positive** - *character* is shifted to the left ◦ **If the value is negative** - *character* is shifted to the right If the specified value is greater than 15 or less than -15 the result is binary zero, CHR(0).

Output

Character.

Examples

Basic examples

Returns the letter "X", or CHR(88) (00010110 becomes 01011000):

```
SHIFT(CHR(22), 2)
```

Returns the backspace character, or CHR(8) (00010000 becomes 00001000):

```
SHIFT(CHR(16), -1)
```

Returns the grave accent character, or CHR(96) (10011011 becomes 01100000):

```
SHIFT(CHR(155), 5)
```

Remarks

When to use SHIFT( )

Use the SHIFT( ) function in conjunction with the BYTE( ), CHR( ) and MASK( ) functions to isolate and move individual bits in a record.

SIN( ) function

Returns the sine of an angle expressed in radians, with a precision of 15 decimal places.

Syntax

```
SIN(radians)
```

Parameters

Name	Type	Description
radians	numeric	The measurement of the angle in radians.

Output

Numeric.

Examples

Basic examples

Returns 0.500000000000000 (the sine of the specified number of *radians*, equivalent to 30 degrees):

```
SIN(0.523598775598299)
```

Returns 0.500000000000000 (the sine value of 30 degrees):

```
SIN(30 * PI( )/180)
```

Advanced examples

Using degrees as input

Returns 0.500 (the sine of 30 degrees, rounded to 3 decimal places):

```
DEC(SIN(30 * PI( )/180),3)
```

Remarks

Performing the Mantissa Arc Test

The three trigonometric functions in Analytics – SIN( ), COS( ), and TAN( ) – support performing the Mantissa Arc Test associated with Benford's Law.

Converting degrees to radians

If your input is in degrees you can use the PI( ) function to convert the input to radians: (*degrees* * PI ( )/180) = *radians*. If required, you can round or truncate the return value using the DEC( ) function.

SORTWORDS( ) function

Returns a string with individual words sorted in sequential order.

Syntax

```
SORTWORDS(string)
```

Parameters

Name	Type	Description
string	character	The string, or the expression, containing words to be sorted. A "word" is any character or group of characters separated by spaces: ◦ an actual word or name ◦ a number that uses the character data type ◦ groups of letters, numbers, or special characters such as abbreviations or identifying prefixes Multiple spaces between words are automatically converted to a single space. Leading or trailing spaces are automatically trimmed.

Output

Character.

Examples

Basic examples

Literal character input

Returns "1 2 A Z a z" (non-Unicode Analytics):

```
SORTWORDS("Z a 2 z A 1")
```

Returns "1 2 a A z Z" (Unicode Analytics):

```
SORTWORDS("Z a 2 z A 1")
```

Returns "1 2 A A Z Z":

```
SORTWORDS(UPPER("Z a 2 z A 1"))
```

Returns "CA, FL NY, TX,":

```
SORTWORDS("CA, TX, NY, FL")
```

Returns "CA FL NY TX":

```
SORTWORDS(OMIT("CA, TX, NY, FL", ","))
```

Field input

Returns all the values in the **Vendor_Address** field with address elements sorted into sequential order:

```
SORTWORDS(Vendor_Address)
```

Advanced examples

Sort address elements to improve fuzzy duplicate matching

You can use SORTWORDS( ) as a helper function when performing various kinds of fuzzy matching in Analytics.

Effect of SORTWORDS( ) on Levenshtein distance

First, let's look at the Levenshtein distance between two occurrences of the same address, differently formatted.

Without the SORTWORDS( ) function, the Levenshtein distance returned is 22. A Levenshtein distance that large suggests the two strings are not the same address:

```
LEVDIST("125 SW 39TH ST, Suite 100", "Suite 100, 125 SW 39TH ST")
```

Now, let's add the SORTWORDS( ) function. The Levenshtein distance returned is 2 - dramatically lower - which suggests the two strings are the same address.

```
LEVDIST(SORTWORDS("125 SW 39TH ST, Suite 100"), SORTWORDS("Suite 100,
125 SW 39TH ST"))
```

Isolating fuzzy duplicates for "125 SW 39TH ST, Suite 100"

You create a filter that isolates all values in the **Vendor_Address** field that are within a specified Levenshtein distance of "125 SW 39TH ST, Suite 100":

```
SET FILTER TO LEVDIST(SORTWORDS(Vendor_Address), SORTWORDS("125 SW 39TH
ST, Suite 100"), F) < 3
```

```
SET FILTER TO ISFUZZYDUP(SORTWORDS(Vendor_Address), SORTWORDS("125 SW
39TH ST, Suite 100"), 3, 99)
```

Increasing or decreasing the Levenshtein distance in the expressions (`3`) allows you to adjust the degree of difference in the filtered values.

For more information about Levenshtein distance, see "LEVDIST( ) function" on page 2214.

Remarks

Overview video

For a video providing an overview of the function, see Fuzzy Matching Using SORTWORDS() (English only).

The sort sequence used by SORTWORDS( )

The SORTWORDS( ) function uses whatever sort sequence is specified in the **Sort Order** option (**Tools > Options > Table**). The default sort sequences are shown below.

For detailed information, see "The Sort Order option and sort sequences " on page 1123.

Analytics Edition	Sort Order default	Associated sort sequence
non-Unicode	System Default (ASCII)	Numbers, then uppercase, then lowercase: `0, 1, 2... A, B, C... a, b, c...` For example, “Z” sorts before “a”.
Unicode	Mix Languages (UCA) (Unicode collation algorithm)	Numbers, then lowercase and uppercase intermixed: `0, 1, 2... a, A, b, B, c, C...` For example, “a” sorts before “Z”.

Case sensitivity

SORTWORDS( ) is case sensitive. Depending on which edition of Analytics you are using (non-Unicode or Unicode), casing in strings may affect sorting.

You can use the UPPER( ) function in conjunction with SORTWORDS( ) if you do not want case to affect sorting:

```
SORTWORDS(UPPER("string"))
```

SORTWORDS( ) can improve fuzzy matching

SORTWORDS( ) can improve the effectiveness of fuzzy matching commands, and filters or conditions that use fuzzy matching functions:

- "FUZZYJOIN command" on page 1729
- "FUZZYDUP command" on page 1723
- "ISFUZZYDUP( ) function" on page 2198
- "LEVDIST( ) function" on page 2214
- "DICECOEFFICIENT( ) function" on page 2116

> **Caution**
> If you use SORTWORDS( ) in conjunction with any of the fuzzy matching commands or functions you must apply SORTWORDS( ) to both strings or both fields being compared. Applying the function to only one of the two strings or fields can seriously degrade the results of fuzzy matching.

Levenshtein distance algorithm

SORTWORDS( ) is especially helpful when used with commands and functions based on the Levenshtein distance algorithm, which returns greater difference values when identical or similar elements in two strings are in different positions. By sorting the elements first, you can significantly reduce the difference values returned by the Levenshtein algorithm. Result sets have fewer false positives and more true positives.

Dice coefficient algorithm

SORTWORDS( ) may be helpful when used with commands and functions based on the Dice coefficient algorithm. However, an improvement in effectiveness is not always the case. By design, the Dice coefficient algorithm minimizes the importance of the position of elements, so sorting elements has less impact than it does with the Levenshtein distance algorithm.

Depending on the nature of the data, SORTWORDS( ) may actually degrade effectiveness by causing the Dice coefficient algorithm to return lower scores. Test a set of sample data before deciding whether to use SORTWORDS( ) in conjunction with the Dice coefficient algorithm in a production setting.

A second consideration is that the benefit of using SORTWORDS( ) is more modest when the Dice coefficient *n*-gram length is shorter. As you reduce the *n*-gram length, the Dice coefficient algorithm increasingly minimizes the importance of the position of elements.

SOUNDEX( ) function

Returns the soundex code for the specified string, which can be used for phonetic comparisons with other strings.

Syntax

```
SOUNDEX(name)
```

Parameters

Name	Type	Description
name	character	The character expression to evaluate.

Output

Character. Returns a four-character soundex code.

Examples

Basic examples

Words that sound the same but are spelled differently

The two examples below return the same soundex code because they sound the same even though they are spelled differently.

Returns F634:

```
SOUNDEX("Fairdale")
```

Returns F634:

```
SOUNDEX("Faredale")
```

Words that sound similar

The two examples below return soundex codes that are different, but close to one another, because the two words sound similar.

Returns J525:

```
SOUNDEX("Jonson")
```

Returns J523:

```
SOUNDEX("Jonston")
```

Words that sound different

The two examples below return soundex codes that are quite different, because the two words sound nothing alike.

Returns S530:

```
SOUNDEX("Smith")
```

Returns M235:

```
SOUNDEX("MacDonald")
```

Field input

Returns the soundex code for each value in the **Last_Name** field:

```
SOUNDEX(Last_Name)
```

Advanced examples

Identifying matching soundex codes

Create the computed field **Soundex_Code** to display the soundex code for each value in the **Last_Name** field:

```
DEFINE FIELD Soundex_Code COMPUTED SOUNDEX(Last_Name)
```

Add the computed field **Soundex_Code** to the view, and then perform a duplicates test on the computed field to identify any matching soundex codes:

```
DUPLICATES ON Soundex_Code OTHER Last_Name PRESORT OPEN TO "Possible_
Dupes.fil"
```

Matching soundex codes indicate that the associated character values in the **Last_Name** field are possible duplicates.

Remarks

When to use SOUNDEX( )

Use the SOUNDEX( ) function to find values that sound similar. Phonetic similarity is one way of locating possible duplicate values, or inconsistent spelling in manually entered data.

How it works

SOUNDEX( ) returns the American Soundex code for the evaluated string. All codes are one letter followed by three numbers. For example: "F634".

How the soundex code is derived

- The first character in the code represents the first letter of the evaluated string.
- Each number in the code represents one of the six American Soundex groups. The groups are composed of phonetically similar consonants.

  Based on these groups, the soundex process encodes the first three consonants in the evaluated string after the first letter.

What the soundex process ignores

The soundex process ignores:

- capitalization
- vowels
- the consonants "H", "W", and "Y"
- any consonants that appear after the three encoded consonants

One or more trailing zeros (0) in the returned code indicate an evaluated string with fewer than three consonants after the first letter.

Limitations of the soundex process

Both the SOUNDEX( ) and SOUNDSLIKE( ) functions have certain limitations:

- The soundex algorithm is designed to work with words pronounced in English, and has varying degrees of effectiveness when used with other languages.
- Although the soundex process performs a phonetic match, matching words must all begin with the same letter, which means that some words that sound the same are not matched.

  For example, a word that begins with "F", and a word that begins with a "Ph", could sound the same but they will never be matched.

Related functions

- **SOUNDSLIKE( )** – an alternate method for phonetically comparing strings.
- **ISFUZZYDUP( ) and LEVDIST** – compare strings based on an orthographic comparison (spelling) rather than on a phonetic comparison (sound).
- **DICECOEFFICIENT( )** – de-emphasizes or completely ignores the relative position of characters or character blocks when comparing strings.

SOUNDSLIKE( ) function

Returns a logical value indicating whether a string phonetically matches a comparison string.

Syntax

```
SOUNDSLIKE(name, sounds_like_name)
```

Parameters

Name	Type	Description
name	character	The first string in the comparison.
sounds_like_name	character	The second string in the comparison.

Output

Logical. Returns **T** (true) if the values being compared phonetically match, and **F** (false) otherwise.

Examples

Basic examples

Returns T, because "Fairdale" and "Faredale" both have a soundex code of F634:

```
SOUNDSLIKE("Fairdale","Faredale")
```

Returns F, because "Jonson" has a soundex code of J525, and "Jonston" has a soundex code of J523:

```
SOUNDSLIKE("Jonson","Jonston")
```

Returns a logical value (T or F) indicating whether the soundex code for each value in the **Last_Name** field matches the soundex code for the string "Smith":

```
SOUNDSLIKE(Last_Name,"Smith")
```

Advanced examples

Isolating values that sound like "Smith"

Create a filter that isolates all values in the **Last_Name** field that sound like "Smith":

```
SET FILTER TO SOUNDSLIKE(Last_Name,"Smith")
```

Remarks

When to use SOUNDSLIKE( )

Use the SOUNDSLIKE( ) function to find values that sound similar. Phonetic similarity is one way of locating possible duplicate values, or inconsistent spelling in manually entered data.

How it works

SOUNDSLIKE( ) converts the comparison strings to four-character American Soundex codes, which are based on the first letter, and the first three consonants after the first letter, in each string.

The function then compares each string's code and returns a logical value indicating whether they match.

For more information about soundex codes, see "SOUNDEX( ) function" on page 2383.

Case sensitivity

The function is not case-sensitive, so "SMITH" is equivalent to "smith."

Limitations of the soundex process

Both the SOUNDSLIKE( ) and SOUNDEX( ) functions have certain limitations:

- The soundex algorithm is designed to work with words pronounced in English, and has varying degrees of effectiveness when used with other languages.
- Although the soundex process performs a phonetic match, matching words must all begin with the same letter, which means that some words that sound the same are not matched.

  For example, a word that begins with "F", and a word that begins with a "Ph", could sound the same but they will never be matched.

Related functions

- **SOUNDEX( )** – an alternate method for phonetically comparing strings.
- **ISFUZZYDUP( ) and LEVDIST** – compare strings based on an orthographic comparison (spelling) rather than on a phonetic comparison (sound).
- **DICECOEFFICIENT( )** – de-emphasizes or completely ignores the relative position of characters or character blocks when comparing strings.

SPLIT( ) function

Returns a specified segment from a string.

Syntax

```
SPLIT(string, separator, segment <,text_qualifier>)
```

Parameters

Name	Type	Description
string	character	The field, expression, or literal value to extract the segment from.
separator	character	The character or characters that delimit segments. For more information, see "How the separator character works" on page 2392.
segment	numeric	The segment to extract. Use a number to specify which segment to extract. For example, to extract the third segment, specify `3`.
text_qualifier optional	character	The character or characters that indicate the start and end of segments of text. If the *separator* character occurs inside a paired set of text qualifiers, it is read as text and not as a separator. You must enclose the *text qualifier* in quotation marks. A single quotation mark *text qualifier* must be enclosed in double quotation marks, and a double quotation mark *text qualifier* must be enclosed in single quotation marks. **Tip** This optional parameter can be useful when working with imported source data that retains separators and text qualifiers.

Output

Character.

Examples

Basic examples

Comma-delimited segments

Returns "seg1":

```
SPLIT("seg1,seg2,seg3", ",", 1)
```

Returns "seg3":

```
SPLIT("seg1,seg2,seg3", ",", 3)
```

Returns "" (the third segment is empty):

```
SPLIT("seg1,seg2,,seg4", ",", 3)
```

Multi-character and space delimiters

Returns "seg3":

```
SPLIT("seg1/*seg2/*seg3", "/*", 3)
```

Returns "Doe":

```
SPLIT("Jane Doe", " ", 2)
```

Escaping delimiters with a text qualifier

Returns "Doe, Jane", which includes a comma that is read as text rather than as a separator:

```
SPLIT('"Doe, Jane","Smith, John"', ",", 1, '"')
```

Advanced examples

Extracting digits from a credit card number

Use the SPLIT( ) command to remove dashes from a credit card number.

Variables are used to capture each segment of the credit card number, and then the segments are concatenated together in an additional variable.

```
ASSIGN seg1 = SPLIT("4150-2222-3333-4444", "-", 1)
ASSIGN seg2 = SPLIT("4150-2222-3333-4444", "-", 2)
ASSIGN seg3 = SPLIT("4150-2222-3333-4444", "-", 3)
ASSIGN seg4 = SPLIT("4150-2222-3333-4444", "-", 4)
ASSIGN ccNum = seg1 + seg2 + seg3 + seg4
```

The value of ccNum is "4150222233334444".

The example illustrates the SPLIT( ) function, but note that the dashes can be removed more efficiently using the EXCLUDE( ) function.

Remarks

How it works

The SPLIT( ) function breaks character data into segments based on separators such as spaces or commas and returns a specified segment.

When to use SPLIT( )

Use the SPLIT( ) function to extract a particular segment of data from a record or field. The segment must appear in the same position in each record or field.

How the separator character works

The separator character delimits, or indicates, the segments of data in a source string.

In a string with a number of segments, most of the segments appear between two separators. However, the first segment may not have a separator character preceding it, and the last segment may not have a separator character following it.

If the source string does not begin with a separator, the segment preceding the first separator is treated as segment 1.

Returns "seg1":

```
SPLIT("seg1,seg2,seg3", ",", 1)
```

If the source string begins with a separator, segment 1 is consider to be null. The segment that follows the separator is treated as segment 2.

Returns "seg1":

```
SPLIT(",seg1,seg2,seg3", ",", 2)
```

Case sensitivity

If *separator* or *text_qualifier* specify characters that have both an uppercase and a lowercase version, the case used must match the case of the separator or text qualifier in the data.

Related functions

SPLIT( ) and SUBSTR( ) both return a segment of data from a longer source string.

- SPLIT( ) identifies the segment based on a separator character.
- SUBSTR( ) identifies the segment based on a numeric character position.

STOD( ) function

Converts a serial date – that is, a date expressed as an integer – to a date value. Abbreviation for "Serial to Date".

Syntax

```
STOD(serial_date <,start_date>)
```

Parameters

Name	Type	Description
serial_date	numeric	The field, expression, or literal value to convert. *serial_date* can be a serial date or a serial datetime. Only the date portion of serial datetimes is considered. The time portion is ignored.
start_date optional	datetime	The start date from which serial dates are calculated. If omitted, the default start date of 01 January 1900 is used.

Output

Datetime. The date value is output using the current Analytics date display format.

Examples

Basic examples

Returns `20141231` displayed as 31 Dec 2014 assuming a current Analytics date display format of DD MMM YYYY:

```
STOD(42003)
```

Returns `20181231` displayed as 31 Dec 2018 assuming a current Analytics date display format of DD MMM YYYY:

```
STOD(42003, `19040101`)
```

Returns the equivalent date for each serial date value in the **Invoice_Date** field:

```
STOD(Invoice_Date)
```

Advanced examples

Adjusting for a start date before 1900-01-01

Use date arithmetic to adjust the start date to a value that is earlier than the Analytics minimum date of January 1, 1900:

1. Convert the serial date using the default start date.
2. Subtract the number of days before 1900-01-01 that the actual start date falls.

To use 1899-01-01 as the start date (evaluates to `20131231`):

```
STOD(42003) - 365
```

Remarks

How it works

The STOD( ) function allows you to convert serial dates to regular dates. Analytics serial dates represent the number of days that have elapsed since 01 January 1900.

Serial date	Regular date equivalent
1	02 January 1900
365	31 December 1900
42003	31 December 2014
0	not valid

For more information about serial dates, see "Serial datetimes" on page 827.

Analytics serial dates compared to Excel serial dates

Analytics serial dates are similar to Microsoft Excel serial dates. You should be aware of one key point of similarity and one key point of difference. The two points are unrelated.

Point of similarity

Both Analytics and Excel treat the year 1900 as a leap year, with 366 days. Although 1900 was not in fact a leap year, Excel treated it as one in order to maintain compatibility with Lotus 1-2-3.

Point of difference

Analytics serial dates are offset from Excel serial dates by one day. In Excel, 01 January 1900 has a serial date of '1'. In Analytics, 01 January 1900 is not counted, and 02 January 1900 has a serial date of '1'.

The *start_date*

Some source data files may use a start date other than 01 January 1900. The *start_date* allows you to match the start date in a source data file. The start date is the date from which serial dates are calculated.

Start date in source data file	Specify:	Details
01 January 1900	`STOD(date_field)`	You do not need to specify a *start_date*, because 01 January 1900 is the default start date.
01 January 1901	`STOD(date_field, `19010101`)`	You specify a *start_date* of `` `19010101` `` to match the start date of 01 January 1901 used in the source data file.
01 January 1899	`STOD(date_field) - 365`	You cannot specify a *start_date* earlier than 01 January 1900. If a source data file uses a start date earlier than 01 January 1900, you can create a datetime expression that subtracts an appropriate number of days from the output results of the STOD( ) function.

Other datetime conversion functions

Serial to Datetime conversion

Function	Description
STODT( )	Converts a serial datetime - that is, a datetime expressed as an integer, and a fractional portion of 24 hours - to a datetime value. Abbreviation for "Serial to Datetime".
STOT( )	Converts a serial time - that is, a time expressed as a fractional portion of 24 hours, with 24 hours equaling 1 - to a time value. Abbreviation for "Serial to Time".

Character or Numeric to Datetime conversion

Function	Description
CTOD( )	Converts a character or numeric date value to a date. Can also extract the date from a character or numeric datetime value and return it as a date. Abbreviation for "Character to Date".
CTODT( )	Converts a character or numeric datetime value to a datetime. Abbreviation for "Character to Datetime".
CTOT( )	Converts a character or numeric time value to a time. Can also extract the time from a character or numeric datetime value and return it as a time. Abbreviation for "Character to Time".

Datetime to Character conversion

Function	Description
DATE( )	Extracts the date from a specified date or datetime and returns it as a character string. Can also return the current operating system date.
DATETIME( )	Converts a datetime to a character string. Can also return the current operating system datetime.
TIME( )	Extracts the time from a specified time or datetime and returns it as a character string. Can also return the current operating system time.

STODT( ) function

Converts a serial datetime – that is, a datetime expressed as an integer, and a fractional portion of 24 hours – to a datetime value. Abbreviation for "Serial to Datetime".

Syntax

```
STODT(serial_datetime <,start_date>)
```

Parameters

Name	Type	Description
serial_datetime	numeric	The field, expression, or literal value to convert. Serial datetime values with the date and time portions separated by a decimal point are required. For example, 42003.75000
start_date optional	datetime	The start date from which serial dates are calculated. If omitted, the default start date of 01 January 1900 is used.

Output

Datetime. The datetime value is output using the current Analytics date and time display formats.

Examples

Basic examples

Unadjusted start dates

Returns `20141231t060000` displayed as 31 Dec 2014 06:00:00 AM assuming current Analytics date and time display formats of DD MMM YYYY and hh:mm:ss PM:

```
STODT(42003.25000)
```

Returns `20141231t191530` displayed as 31 Dec 2014 07:15:30 PM assuming current Analytics date and time display formats of DD MMM YYYY and hh:mm:ss PM:

```
STODT(42003.802431)
```

Adjusted start dates

Returns `20181231t120000` displayed as 31 Dec 2018 12:00:00 PM assuming current Analytics date and time display formats of DD MMM YYYY and hh:mm:ss PM:

```
STODT(42003.50000, `19040101`)
```

Fields as input

Returns the equivalent datetime for each serial datetime value in the **Receipt_datetime** field:

```
STODT(Receipt_datetime)
```

Advanced examples

Adjusting for a start date before 1900-01-01

Use date arithmetic to adjust the start date to a value that is earlier than the Analytics minimum date of January 1, 1900:

1. Convert the serial datetime using the default start date.
2. Subtract the number of days before 1900-01-01 that the actual start date falls.

To use 1899-01-01 as the start date (evaluates to `20131231t180000`):

```
STODT(42003.75000) - 365
```

Remarks

How it works

The STODT( ) function allows you to convert serial datetimes to regular datetimes. Analytics serial datetimes represent the number of days that have elapsed since 01 January 1900, and following the decimal point, represent a fractional portion of 24 hours, with 24 hours equaling 1.

Serial datetime	Regular datetime equivalent
1.25	02 January 1900 06:00:00 AM
365.75000	31 December 1900 06:00:00 PM
42003.79167	31 December 2014 07:00:00 PM
42003.802431	31 December 2014 07:15:30 PM
42003.00000	31 December 2014 12:00:00 AM
42003.50000	31 December 2014 12:00:00 PM
0.0	not valid

For more information about serial datetimes, see "Serial datetimes" on page 827.

Analytics serial dates compared to Excel serial dates

Analytics serial dates are similar to Microsoft Excel serial dates. You should be aware of one key point of similarity and one key point of difference. The two points are unrelated.

Point of similarity

Both Analytics and Excel treat the year 1900 as a leap year, with 366 days. Although 1900 was not in fact a leap year, Excel treated it as one in order to maintain compatibility with Lotus 1-2-3.

Point of difference

Analytics serial dates are offset from Excel serial dates by one day. In Excel, 01 January 1900 has a serial date of '1'. In Analytics, 01 January 1900 is not counted, and 02 January 1900 has a serial date of '1'.

The *start_date*

Some source data files may use a start date other than 01 January 1900. The *start_date* allows you to match the start date in a source data file. The start date is the date from which serial datetimes are calculated.

Start date in source data file	Specify:	Details
01 January 1900	`STODT(datetime_field)`	You do not need to specify a *start_date*, because 01 January 1900 is the default start date.
01 January 1901	`STODT(datetime_field, `19010101`)`	You specify a *start_date* of `` `19010101` `` to match the start date of 01 January 1901 used in the source data file.
01 January 1899	`STODT(datetime_field) - 365`	You cannot specify a *start_date* earlier than 01 January 1900. If a source data file uses a start date earlier than 01 January 1900, you can create a datetime expression that subtracts an appropriate number of days from the output results of the STODT( ) function.

Other datetime conversion functions

Serial to Datetime conversion

Function	Description
STOD( )	Converts a serial date - that is, a date expressed as an integer - to a date value. Abbreviation for "Serial to Date".
STOT( )	Converts a serial time - that is, a time expressed as a fractional portion of 24 hours, with 24 hours equaling 1 - to a time value. Abbreviation for "Serial to Time".

Character or Numeric to Datetime conversion

Function	Description
CTOD( )	Converts a character or numeric date value to a date. Can also extract the date from a character or numeric datetime value and return it as a date. Abbreviation for "Character to Date".
CTODT( )	Converts a character or numeric datetime value to a datetime. Abbreviation for "Character to Datetime".

Function	Description
CTOT( )	Converts a character or numeric time value to a time. Can also extract the time from a character or numeric datetime value and return it as a time. Abbreviation for "Character to Time".

Datetime to Character conversion

Function	Description
DATE( )	Extracts the date from a specified date or datetime and returns it as a character string. Can also return the current operating system date.
DATETIME( )	Converts a datetime to a character string. Can also return the current operating system datetime.
TIME( )	Extracts the time from a specified time or datetime and returns it as a character string. Can also return the current operating system time.

STOT( ) function

Converts a serial time – that is, a time expressed as a fractional portion of 24 hours, with 24 hours equaling 1 – to a time value. Abbreviation for "Serial to Time".

Syntax

```
STOT(serial_time)
```

Parameters

Name	Type	Description
serial_time	numeric	The field, expression, or literal value to convert. *serial_time* can be a serial time or a serial datetime. Only the time portion of serial datetimes is considered. The date portion is ignored.

Output

Datetime. The time value is output using the current Analytics time display format.

Examples

Basic examples

Returns `t060000` displayed as 06:00:00 AM assuming a current Analytics time display format of hh:mm:ss PM:

```
STOT(0.25000)
```

Returns `t191530` displayed as 07:15:30 PM assuming a current Analytics time display format of hh:mm:ss PM:

```
STOT(0.802431)
```

Returns the equivalent regular time for each serial time value in the **Login_time** field:

```
STOT(Login_time)
```

Remarks

When to use STOT( )

Use the STOT( ) function to convert serial times to regular times.

What are serial times?

Analytics serial times represent a fractional portion of 24 hours, with 24 hours equaling 1.

For example:

- the serial time equivalent of 1 hour is 1/24, or 0.04167
- the serial time equivalent of 1 minute is 1/1440, or 0.0006945

Serial times can be prefaced with a '0' (zero) and a decimal point, or just a decimal point.

1.000000 is not a valid serial time

Although 24 hours equals 1 for the purposes of calculating serial times, 1.000000 is not a valid serial time. Valid serial times are all decimal fractions less than 1. For example: 0.75000 (06:00:00 PM).

Analytics treats the serial number 1.000000 as the serial datetime equivalent to 02 Jan 1900 12:00:00 AM. Because STOT( ) ignores the date portion of datetimes, `STOT(1.000000)` is equivalent to `STOT(0.000000)` and both are equivalent to the regular time 12:00:00 AM.

Serial times and regular time equivalents

Serial time	Regular time equivalent
0.00	12:00:00 AM
0.0006945	12:01:00 AM
0.04167	01:00:00 AM

Serial time	Regular time equivalent
0.0423645	01:01:00 AM
0.042998	01:01:55 AM
0.25	06:00:00 AM
0.50	12:00:00 PM
0.75	06:00:00 PM
0.79167	07:00:00 PM
0.802431	07:15:30 PM
1.00	12:00:00 AM

Other datetime conversion functions

Serial to Datetime conversion

Function	Description
STOD( )	Converts a serial date - that is, a date expressed as an integer - to a date value. Abbreviation for "Serial to Date".
STODT( )	Converts a serial datetime - that is, a datetime expressed as an integer, and a fractional portion of 24 hours - to a datetime value. Abbreviation for "Serial to Datetime".

Character or Numeric to Datetime conversion

Function	Description
CTOD( )	Converts a character or numeric date value to a date. Can also extract the date from a character or numeric datetime value and return it as a date. Abbreviation for "Character to Date".
CTODT( )	Converts a character or numeric datetime value to a datetime. Abbreviation for "Character to Datetime".
CTOT( )	Converts a character or numeric time value to a time. Can also extract the time from a character or numeric datetime value and return it as a time. Abbreviation for "Character to Time".

Datetime to Character conversion

Function	Description
DATE( )	Extracts the date from a specified date or datetime and returns it as a character string. Can also return the current operating system date.
DATETIME( )	Converts a datetime to a character string. Can also return the current operating system datetime.
TIME( )	Extracts the time from a specified time or datetime and returns it as a character string. Can also return the current operating system time.

STRING( ) function

Converts a numeric value to a character string.

Syntax

```
STRING(number, length <,format>)
```

Parameters

Name	Type	Description
number	numeric	The numeric value to convert to a string.
length	numeric	The length of the output string in characters. ◦ If *length* is longer than *number*, leading spaces are added to the output string ◦ If *length* is shorter than *number*, the output string is truncated from the left side Ensure that the length you specify provides enough space for the longest numeric value in a field, including any non-numeric format characters if you specify the optional *format* parameter.
format optional	character	The format to apply to the output string. *format* must be enclosed in double quotation marks. For example, `"(9,999.99)"` Use the optional *format* parameter to add formatting to the output string that is not present in the source data. You can add a dollar sign, a percent sign, one or more decimal placeholders, a thousands separator, parentheses, and so on. **Note** Non-numeric format characters that you specify increase the length of *number*.

Output

Character.

Examples

Basic examples, output not formatted

Numeric value `125.2`

Returns " 125.2":

```
STRING(125.2, 6)
```

The output string is padded with one leading space because the *length* value is `6`, which is one character longer than the number of digits and format characters in *number*.

Numeric value `-125.2`

Returns "25.2":

```
STRING(-125.2, 4)
```

The output string is truncated because the *length* value is `4`, which is two characters shorter than the number of digits and format characters in *number*.

Returns " -125.2":

```
STRING(-125.2, 7)
```

The output string is padded with one leading space because the *length* value is `7`, which is one character longer than the number of digits and format characters in *number*.

Basic examples, output formatted

Numeric value `125.2`

Returns "25.20":

```
STRING(125.2, 6, "(9,999.99)")
```

The output string is truncated because the *length* value is `6`, which is one character shorter than the *number* value after the specified format is applied.

Returns "125.20":

```
STRING(125.2, 7, "(9,999.99)")
```

> **Note**
> Starting from the right, the characters you specify for *format* are included in the calculation of the length of *number* even if a format character is not required for a specific instance of *number*. In the examples above, the right-hand parenthesis is counted as a character even though it is not required for a positive value in *number*.

Numeric value `-125.2`

Returns " (125.20)":

```
STRING(-125.2, 10, "(9,999.99)")
```

The output string is padded with two leading spaces because the *length* value is `10`, which is two characters longer than the *number* value after the specified format is applied.

Basic example, field input

Returns numeric values in the **Employee_number** field as character strings with a length of 10 characters. If required, the output string is padded or truncated:

```
STRING(Employee_number, 10)
```

Remarks

Formatting the output string

You can format the output string to display formatting that might be missing in the source data.

Placeholder digits in the format

In the format that you specify, the digit `9` acts as a placeholder for digits. Ensure that you specify enough placeholder digits to account for the longest numeric value in a field. For example, if a field contains amounts up to $5,000,000, with two decimal places, you need to specify nine placeholder digits: `"$9,999,999.99"`

How the format affects the minimum required output string length

The value you specify for *length* must, at a minimum, be long enough to contain all the digits in the longest value in a field, as well as any format characters that you specify.

If you want to add a dollar sign, and thousands separators, to the values in the field containing amounts up to $5,000,000, you need to specify at least `13` for *length*: 9 digits + 4 non-numeric format characters.

Returns numeric values in the **Amount** field as character strings with the specified format displayed.

```
STRING(Amount, 13, "$9,999,999.99")
```

Returns $4,789,123.50, as a character string:

```
STRING(4789123.50, 13, "$9,999,999.99")
```

Related functions

The STRING( ) function is the opposite of the VALUE( ) function, which converts character data to numeric data.

SUBSTR( ) function

Returns a specified substring from a string.

Syntax

```
SUBSTR(string, start, length)
```

Parameters

Name	Type	Description
string	character	The field, expression, or literal value to extract the substring from.
start	numeric	The starting character position of the substring. The numeric positions of the characters in *string* start at `1`. To extract a substring beginning with `C` from the string `ABCDEF`, you would specify a *start* value of `3`.
length	numeric	The number of characters in the substring. If *length* is 0, the output is blank.

Output

Character.

Examples

Basic examples

Literal character input

Returns "BCD":

```
SUBSTR("ABCDEF", 2, 3)
```

Returns "EF":

```
SUBSTR("ABCDEF", 5, 10)
```

Parsing structured character data

Returns "189543":

```
SUBSTR("***189543***", 4, 6)
```

Returns the four-digit year out of a character field containing dates formatted as "MM/DD/YYYY":

```
SUBSTR(DATE, 7, 4)
```

Advanced examples

Increasing field length

Use SUBSTR( ) to increase the length of a character field. Increasing the length of a field is a common harmonization task that you may need to perform before joining or appending two fields.

The example below pads the **Product_Description** field with blank spaces to create the computed field **Product_Description_Long** with a length of 50 characters.

```
DEFINE FIELD Product_Description_Long COMPUTED SUBSTR(Product_Descrip-
tion, 1, 50)
```

Remarks

How it works

The SUBSTR( ) function returns characters from the *string* value starting at the character position specified by *start*. The number of characters returned is specified by *length*.

How SUBSTR( ) handles spaces

Leading, trailing, or internal spaces in the *string* value are treated like characters. Spaces captured by *start* and *length* are included in the output string.

How padding works

If the *length* value exceeds the number of characters, including trailing spaces, from the *start* position to the end of *string*, the output string may or may not be padded on the right with spaces.

Padded with spaces

If you use SUBSTR( ) within a command that creates a field, the output is padded with spaces.

Padding when creating a computed field

Creates the computed field **Product_Description_Long**, with a length of 50 characters, based on the physical field **Product_Description**, with a length of 24 characters:

```
DEFINE FIELD Product_Description_Long COMPUTED SUBSTR(Product_Descrip-
tion, 1, 50)
```

Padding when extracting a physical field

Extracts the field **Product_Description_Long**, with a length of 50 characters, to a new table, based on the physical field **Product_Description**, with a length of 24 characters:

```
EXTRACT FIELDS SUBSTR(Product_Description, 1, 50) AS "Product_Descrip-
tion_Long" TO New_Table
```

Not padded with spaces

If you use SUBSTR( ) in a variable definition or an expression, the output is not padded with spaces.

No padding when defining a variable

Creates the variable *v_prod_desc*, with a length of 24 characters, based on the field length of **Product_Description**:

```
ASSIGN v_prod_desc = SUBSTR(Product_Description, 1, 50)
```

> **Note**
> Even though SUBSTR( ) specifies a *length* of 50 characters, the output is limited to the field length of **Product_Description**.

Related functions

SUBSTR( ) and SPLIT( ) both return a segment of data from a longer source string.

- SUBSTR( ) identifies the segment based on a numeric character position.
- SPLIT( ) identifies the segment based on a separator character.

TAN( ) function

Returns the tangent of an angle expressed in radians, with a precision of 15 decimal places.

Syntax

```
TAN(radians)
```

Parameters

Name	Type	Description
radians	numeric	The measurement of the angle in radians.

Output

Numeric.

Examples

Basic examples

Returns 0.999999999999999 (the tangent of the specified number of *radians*, equivalent to 45 degrees):

```
TAN(0.785398163397448)
```

Returns 0.999999999999999 (the tangent of 45 degrees):

```
TAN(45 * PI( )/180)
```

Advanced examples

Using degrees as input

Returns 1.000 (the tangent of 45 degrees, rounded to 3 decimal places):

```
DEC(TAN(45 * PI( )/180),3)
```

Remarks

Performing the Mantissa Arc Test

The three trigonometric functions in Analytics – SIN( ), COS( ), and TAN( ) – support performing the Mantissa Arc Test associated with Benford's Law.

Converting degrees to radians

If your input is in degrees you can use the PI( ) function to convert the input to radians: (*degrees* * PI ( )/180) = *radians*. If required, you can round or truncate the return value using the DEC( ) function.

TEST( ) function

Returns a logical value indicating whether a specified string occurs at a specific byte position in a record.

Syntax

```
TEST(byte_position, string)
```

Parameters

Name	Type	Description
byte_position	numeric	The sequential number from the left in the table layout that identifies the location of the first character of *string*. The function evaluates to F if the start of *string* is not identified at this position, even if *string* appears at another position in the record.
string	character	The character string to search for. The search is case-sensitive. If there is a chance the case may be mixed, use the UPPER( ) function to convert all characters to uppercase.

Output

Logical. Returns **T** (true) if the specified string starts at the specified byte location within a record, and **F** (false) otherwise.

Examples

Basic examples

Given a record containing:

```
    Department: Marketing
....|....|....|....|....|
```

Returns T:

```
TEST(5, "Department")
```

Returns F, because in the record, "Department" starts at the fifth byte position, not the sixth:

```
TEST(6, "Department")
```

Returns F, because the function is case-sensitive:

```
TEST(5, "DEPARTMENT")
```

Advanced examples

Isolating records that are page headings

Use TEST( ) to create a filter that isolates all records that start with "Page:":

```
SET FILTER TO TEST(1, "Page:")
```

TIME( ) function

Extracts the time from a specified time or datetime and returns it as a character string. Can also return the current operating system time.

Syntax

```
TIME(<time/datetime> <,format>)
```

Parameters

Name	Type	Description
time/datetime optional	datetime	The field, expression, or literal value to extract the time from. If omitted, the current operating system time is returned in the format hh:mm:ss.
format optional	character	The format to apply to the output string, for example `"hh:mm:ss"`. If omitted, the current Analytics time display format is used. You cannot specify *format* if you have omitted *time/datetime*.

Output

Character.

Examples

Basic examples

Literal input values

Returns "23:59:59" assuming an Analytics time display format of hh:mm:ss:

```
TIME(`20141231 235959`)
```

Returns "11:59 P":

```
TIME(`20141231 235959`, "hh:mm A")
```

Returns the current operating system time returned as a character string in hh:mm:ss format (24-hour clock):

```
TIME()
```

Field as input values

Returns a character string for each value in the **Receipt_timestamp** field, using the current Analytics time display format:

```
TIME(Receipt_timestamp)
```

Returns a character string for each value in the **Receipt_timestamp** field, using the specified time display format:

```
TIME(Receipt_timestamp,  "hh:mm:ss")
```

Advanced examples

Calculating the elapsed time for a command or a script to execute

Use the TIME( ) function to help calculate the amount of time a particular Analytics command, or an entire script, takes to execute.

Immediately before the command you want to time, or at the start of the script, specify this line to create a variable that stores the current operating system time:

```
ASSIGN Time_started = TIME()
```

Immediately after the command, or at the end of the script, specify the two lines below.

The first line creates a variable that stores the operating system time after the command or script completes. The second line calculates the difference between the finish and start times, and converts the result to an easily readable format.

> **Tip**
> You can double-click the CALCULATE log entry to see the elapsed time for the command or the script.

```
ASSIGN Time_finished = TIME()
CALCULATE STOT(CTOT(Time_finished) - CTOT(Time_started))
```

If the command or script will run over the boundary of midnight, use this second line instead:

```
CALCULATE `T000000` - (CTOT(Time_started) - CTOT(Time_finished))
```

Remarks

Output string length

The length of the output string is always 14 characters. If the specified output *format*, or the Analytics time display format, is less than 14 characters, the output string is padded with trailing blank spaces.

Parameter details

A field specified for *time/datetime* can use any time or datetime format, as long as the field definition correctly defines the format.

If you use *format* to control how the output string is displayed, you are restricted to the formats in the table below. You can use any combination of time and AM/PM formats. The AM/PM format is optional, and is placed last.

Specify *format* using single or double quotation marks. For example: `"hh:mm:ss AM"`.

Time formats	AM/PM formats	Examples
hh:mm:ss	none 24-hour clock	"hh:mm:ss"
hhmmss	AM, or PM 12-hour clock	"hhmmss PM"

Time formats	AM/PM formats	Examples
hh:mm	A, or P 12-hour clock	"hh:mm A"
hhmm		
hh		

Specifying a literal time or datetime value

When specifying a literal time or datetime value for *time/datetime*, you are restricted to the formats in the table below, and you must enclose the value in backquotes – for example, `` `20141231 235959` ``.

Do not use any separators such as slashes (/) or colons (:) between the individual components of dates or times.

- **Time values** – you can use any of the time formats listed in the table below. You must use a separator before a standalone time value for the function to operate correctly. Valid separators are the letter 't', or the letter 'T'. You must specify times using the 24-hour clock. Offsets from Coordinated Universal Time (UTC) must be prefaced by a plus sign (+) or a minus sign (-).
- **Datetime values** – you can use any combination of the date, separator, and time formats listed in the table below. The date must precede the time, and you must use a separator between the two. Valid separators are a single blank space, the letter 't', or the letter 'T'.

Example formats	Example literal values
thhmmss	`t235959`
Thhmm	`T2359`
YYYYMMDD hhmmss	`20141231 235959`
YYMMDDthhmm	`141231t2359`
YYYYMMDDThh	`20141231T23`
YYYYMMDD hhmmss+/-hhmm (UTC offset)	`20141231 235959-0500`
YYMMDD hhmm+/-hh (UTC offset)	`141231 2359+01`
Note Do not use hh alone in the main time format with data that has a UTC offset. For example, avoid: hh+hhmm. Results can be unreliable.	

Related functions

If you need to return the current operating system time as a datetime value, use NOW( ) instead of TIME( ).

Other datetime conversion functions

Datetime to Character conversion

Function	Description
DATE( )	Extracts the date from a specified date or datetime and returns it as a character string. Can also return the current operating system date.
DATETIME( )	Converts a datetime to a character string. Can also return the current operating system datetime.

Character or Numeric to Datetime conversion

Function	Description
CTOD( )	Converts a character or numeric date value to a date. Can also extract the date from a character or numeric datetime value and return it as a date. Abbreviation for "Character to Date".
CTODT( )	Converts a character or numeric datetime value to a datetime. Abbreviation for "Character to Datetime".
CTOT( )	Converts a character or numeric time value to a time. Can also extract the time from a character or numeric datetime value and return it as a time. Abbreviation for "Character to Time".

Serial to Datetime conversion

Function	Description
STOD( )	Converts a serial date - that is, a date expressed as an integer - to a date value. Abbreviation for "Serial to Date".
STODT( )	Converts a serial datetime - that is, a datetime expressed as an integer, and a fractional portion of 24 hours - to a datetime value. Abbreviation for "Serial to Datetime".

Function	Description
STOT( )	Converts a serial time - that is, a time expressed as a fractional portion of 24 hours, with 24 hours equaling 1 - to a time value. Abbreviation for "Serial to Time".

TODAY( ) function

Returns the current operating system date as a Datetime data type.

Syntax

```
TODAY()
```

Parameters

This function does not have any parameters.

Output

Datetime.

Examples

Basic examples

Returns the current operating system date as a datetime value, for example `20141231`, displayed using the current Analytics date display format:

```
TODAY()
```

Remarks

Related functions

If you need to return the current operating system date as a character string, use DATE( ) instead of TODAY( ).

TRANSFORM( ) function

Reverses the display order of bi-directional text within a specified string.

Syntax

```
TRANSFORM(original_string)
```

Parameters

Name	Type	Description
original_string	character	The field, expression, or literal value containing bi-directional text.

Output

Character.

Examples

Basic examples

In the input string, the characters "XZQB" represent Hebrew/bidirectional characters in an input string that otherwise contains regular characters.

In the output string, the direction of "XZQB" is reversed, and returns "BQZX". The other characters are unmodified.

Returns "ABC BQZX 123":

```
TRANSFORM("ABC XZQB 123")
```

Remarks

How it works

The TRANSFORMS( ) function identifies bi-directional data and displays it correctly in the view, from right to left.

All other characters processed by the function are unmodified and continue to display from left to right.

When to use TRANSFORMS( )

Use TRANSFORMS( ) to adjust the display order of Arabic or Hebrew characters, so they display correctly.

TRIM( ) function

Returns a string with trailing spaces removed from the input string.

Syntax

```
TRIM(string)
```

Parameters

Name	Type	Description
string	character	The field, expression, or literal value to remove trailing spaces from.

Output

Character.

Examples

Basic examples

Note that in both examples the leading spaces and spaces between words are not removed by the TRIM( ) function.

Returns " Vancouver":

```
TRIM("   Vancouver   ")
```

Returns " New York":

```
TRIM("   New York")
```

Advanced examples

Removing non-breaking spaces

Non-breaking spaces are not removed by the TRIM( ) function.

If you need to remove trailing non-breaking spaces, create a computed field using the following expression:

```
DEFINE FIELD Description_cleaned COMPUTED TRIM(REPLACE(Description, CHR
(160), CHR(32)))
```

The REPLACE( ) function replaces any non-breaking spaces with regular spaces, and then TRIM( ) removes any trailing regular spaces.

Remarks

How it works

The TRIM( ) function removes trailing spaces only. Spaces inside the string, and leading spaces, are not removed.

Related functions

TRIM( ) is related to the LTRIM( ) function, which removes any leading spaces from a string, and to the ALLTRIM( ) function, which removes both leading and trailing spaces.

UNSIGNED( ) function

Returns numeric data converted to the Unsigned data type.

Syntax

```
UNSIGNED(number, length_of_result)
```

Parameters

Name	Type	Description
number	numeric	The value to convert.
length_of_result	numeric	The number of bytes to use in the output string.

Output

Numeric.

Examples

Basic examples

Returns 000075:

```
UNSIGNED(75, 3)
```

```
UNSIGNED(-75, 3)
```

```
UNSIGNED(7.5, 3)
```

Returns 2456 (1 is truncated because only 4 digits can be stored when the *length_of_result* is 2):

```
UNSIGNED(12456, 2)
```

Returns 000000012456:

```
UNSIGNED(-12.456, 6)
```

Remarks

What is Unsigned data?

The Unsigned data type is used by mainframe operating systems to store numeric values in a format that uses minimal space, storing two digits in each byte. The Unsigned data type is the same as the Packed data type, but it does not use the last byte to specify whether the value is positive or negative.

When to use UNSIGNED( )

Use the UNSIGNED( ) function to convert numeric data to the Unsigned format for export to mainframe systems.

Truncated return values

If the *length_of_result* value is shorter than the length of the *number* value, the additional digits are truncated.

UPPER( ) function

Returns a string with alphabetic characters converted to uppercase.

Syntax

```
UPPER(string)
```

Parameters

Name	Type	Description
string	character	The field, expression, or literal value to convert to uppercase.

Output

Character.

Examples

Basic examples

Returns "ABC":

```
UPPER("abc")
```

Returns "ABC 123 DEF":

```
UPPER("abc 123 DEF")
```

Returns "ABCD 12":

```
UPPER("AbCd 12")
```

Returns all the values in the **Last_Name** field converted to uppercase:

```
UPPER(Last_Name)
```

Remarks

How it works

The UPPER( ) function converts all alphabetic characters in *string* to uppercase. All non-alphabetic characters are left unchanged.

When to use UPPER( )

Use UPPER( ) when you need to ensure that all characters in a field, variable, or expression have consistent casing. Consistent casing is particularly important when you are comparing values.

UPPER( ) can also be used for formatting values in reports as uppercase text.

UTOD( ) function

Converts a Unicode string containing a formatted date to an Analytics date value. Abbreviation for "Unicode to Date".

> **Note**
> This function is specific to the Unicode edition of Analytics. It is not a supported function in the non-Unicode edition.
>
> Use this function when working with dates in languages and formats that are different from your default installation. If the string you want to convert is in your default language, use CTOD( ) instead.

Syntax

```
UTOD(string <,locale> <,style>)
```

Parameters

Name	Type	Description
string	character	The Unicode string to convert to a date. The Unicode string can contain a datetime value, but the time portion of the value is ignored. Standalone time values are not supported. *string* must match the input format required by the *style* value for the date's locale.
locale optional	character	The code that specifies the language and locale of the output string, and optionally the version of the language associated with a particular country or region. For example, `"zh"` specifies Chinese, and `"pt_BR"` specifies Brazilian Portuguese. If omitted, the default locale for your computer is used. If a language is specified, but no country is specified, the default country for the language is used. You cannot specify *locale* if you have not specified *date*. For more information about locale codes, see www.unicode.org.
style optional	numeric	The date format style to use for the Unicode string. The format style matches the standard for the locale you specify:

Name	Type	Description
		◦ `0` - full specification format, such as "Sunday, September 18, 2016" ◦ `1` - long format, such as "September 18, 2016" ◦ `2` - medium format, such as "Sep 18, 2016" ◦ `3` - short, numeric format such as "9/18/16" If omitted, the default value of **2** is used. You cannot specify *style* if you have not specified *locale*. **Tip** For help determining the expected format of your input string, do one of the following: • Use the DTOU( ) function to generate an example value using the style and locale. In the command line, use the DISPLAY command to print the value: `DISPLAY DTOU(`20160909`, "es_MX", 3)` • Consult an authoritative source on the standard date format for the style in the specific locale.

Output

Datetime. The date value is output using the current Analytics date display format.

Examples

Basic examples

Note

All examples assume a current Analytics date display format of DD MMM YYYY.

In the examples below, the locale code for Chinese (`"zh"`) and Simplified Chinese (`"zh_CN"`) match different input strings and are not interchangeable.

You must also specify the correct *style*. A long Unicode date string (that is, *style* is `1`) does not return an Analytics date if you specify a *style* of `2`.

Literal input values

Returns `20141231` displayed as 31 Dec 2014:

```
UTOD("31 de dezembro de 2014", "pt_BR", 1)
```

Returns `20141231` displayed as 31 Dec 2014:

```
UTOD("31 grudnia 2014", "pl", 1)
```

Field input values

Returns the date equivalent for each Unicode string in the **Invoice_date** field:

```
UTOD(Invoice_date, "zh", 1)
```

Input uses full date style

Returns `20141231` displayed as 31 Dec 2014 (no region identifier specified):

```
UTOD("星期三, 2014 十二月 31", "zh", 0)
```

Returns `20141231` displayed as 31 Dec 2014 (region identifier specified):

```
UTOD("2014年12月31日星期三", "zh_CN", 0)
```

Input uses long date style

Returns `20141231` displayed as 31 Dec 2014 (no region identifier specified):

```
UTOD("2014 十二月 31", "zh", 1)
```

Returns `20141231` displayed as 31 Dec 2014 (region identifier specified):

```
UTOD("2014年12月31日", "zh_CN", 1)
```

Remarks

Converting Unicode strings successfully

To successfully convert Unicode strings containing dates into Analytics dates you must specify *locale* and *style* parameters that match the language country/region (if applicable), and style of the date in the Unicode string.

Related functions

UTOD( ) is the inverse of DTOU( ), which converts a date to a Unicode string. If you are uncertain which country/region and style to specify for UTOD( ), you can use DTOU( ) and experiment with different parameters to produce an output Unicode string that matches the form of the input Unicode strings you want to convert with UTOD( ).

VALUE( ) function

Converts a character string to a numeric value.

Syntax

```
VALUE(string, decimals)
```

Parameters

Name	Type	Description
string	character	The field, literal, or expression to convert.
decimals	numeric	The number of decimal places to include in the output.

Output

Numeric.

Examples

Basic examples

Returns -123.400:

```
VALUE("123.4-", 3)
```

Returns 123456.00:

```
VALUE("$123,456", 2)
```

Returns -77.45:

```
VALUE("77.45CR", 2)
```

Returns -123457:

```
VALUE(" (123,456.78)", 0)
```

Field input

Returns character values in the **Salary** field as numbers without any decimal places:

```
VALUE(Salary, 0)
```

Remarks

How it works

This function converts character data to numeric data. You can use the VALUE( ) function if you need to convert character expressions or field values to numeric values for use in Analytics commands.

Numeric input formatting

VALUE( ) accepts numbers in any format. You can use any numeric formatting accepted by the Print data type such as punctuation, leading or trailing signs, and parentheses as input.

Negative values

The VALUE( ) function can interpret different indicators of negative values such as parentheses and the minus sign. It can also interpret CR (credit) and DR (debit). For example:

Returns -1000.00:

```
VALUE("(1000)", 2)
```

```
VALUE("1000CR", 2)
```

Decimal vs integer values

If the *string* value does not include decimals, Analytics treats the number as an integer. For example:

Returns 123.00:

```
VALUE("123", 2)
```

If the number of decimals specified by *decimals* is less than the number in the field or expression, the result is rounded. For example:

Returns "10.6":

```
VALUE("10.56", 1)
```

Related functions

The VALUE( ) function is the opposite of the STRING( ) function, which converts numeric data to character data.

VERIFY( ) function

Returns a logical value indicating whether the data in a physical data field is valid.

Syntax

```
VERIFY(field)
```

Parameters

Name	Type	Description
field	character numeric datetime	Must be a physical data field.

Output

Logical. Returns **T** (true) if data in the field is valid, and **F** (false) otherwise.

Examples

Basic examples

Extracts any records where the VERIFY( ) function evaluates to false to a new Analytics table:

```
EXTRACT RECORD IF NOT VERIFY(Address) TO "InvalidEntries.fil"
```

Remarks

The VERIFY( ) function determines whether the data in a field is consistent with the specified data type for the field.

When to use VERIFY( )

The function provides more precise control over the fields you want to verify than either the VERIFY command or the **Verify Data** option in the **Numeric** tab in the **Options** dialog box (**Tools > Options**). You can use the function to detect errors in individual fields and proceed in a situation-specific manner.

When the function evaluates to true

For the function to evaluate to true:

- character fields must contain only valid, printable characters, such as letters, numbers, and symbols
- numeric fields must contain only valid numeric characters, such as numbers, decimals, and currency symbols
- datetime fields must contain only valid dates, datetimes, or times

Computed fields and expressions

Computed fields and expressions always evaluate to **T** (true), so this function cannot be used with computed fields or expressions unless they are first converted to physical fields. Using the **Fields** option in the **Extract** dialog box to extract computed fields or expressions converts them to physical fields.

WORKDAY( ) function

Returns the number of workdays between two dates.

Syntax

```
WORKDAY(start_date, end_date <,nonworkdays>)
```

Parameters

Name	Type	Description
start_date	datetime	The start date of the period for which workdays are calculated. The start date is included in the period.
end_date	datetime	The end date of the period for which workdays are calculated. The end date is included in the period.
nonworkdays optional	character	The days of the week that are weekend days, or non workdays, and excluded from the calculation. If *nonworkdays* is omitted, Saturday and Sunday are used as the default non workdays. Enter *nonworkdays* using the following abbreviations, separated by a space or a comma: ◦ Mon ◦ Tue ◦ Wed ◦ Thu ◦ Fri ◦ Sat ◦ Sun *nonworkdays* is not case-sensitive. The abbreviations must be entered in English even if you are using a non-English version of Analytics: `"Fri, Sat, Sun"`

> **Note**
>
> You can specify a datetime value for *start_date* or *end_date* but the time portion of the value is ignored.
>
> If *start_date* is more recent than *end_date*, a negative value is returned.

Output

Numeric. The number of workdays in the period for which workdays are calculated.

Examples

Basic examples

Literal input values

Returns 5 (the number of workdays between Monday, March 02, 2015 and Sunday, March 08, 2015 inclusive):

```
WORKDAY(`20150302`, `20150308`)
```

Returns 6 (the number of workdays between Monday, March 02, 2015 and Sunday, March 08, 2015 inclusive, when Sunday is the only non workday):

```
WORKDAY(`20150302`, `20150308`, "Sun")
```

Returns 5 (the number of workdays between Sunday, March 01, 2015 and Saturday, March 07, 2015 inclusive, when Friday and Saturday are the non workdays):

```
WORKDAY(`20150301`, `20150307`, "Fri, Sat")
```

You can also use the function to calculate the number of weekend days in a date range.

Returns 2 (the number of weekend days between Monday, March 02, 2015 and Sunday, March 08, 2015 inclusive):

```
WORKDAY(`20150302`, `20150308`, "Mon, Tue, Wed, Thu, Fri")
```

Field input values

Returns the number of workdays between each date in the **Start_date** field and December 31, 2015 inclusive:

```
WORKDAY(Start_date, `20151231`)
```

Returns the number of workdays between each date in the **Start_date** field and a corresponding date in the **End_date** field inclusive:

- Statutory holidays are included in the workdays total and may need to be factored out using a separate calculation
- A negative return value indicates a start date that is more recent than an end date

```
WORKDAY(Start_date, End_date)
```

Remarks

Date formats

A field specified for *start_date* or *end_date* can use any date format, as long as the field definition correctly defines the format.

When specifying a literal date value for *start_date* or *end_date*, you are restricted to the formats `YYYYMMDD` and `YYMMDD`, and you must enclose the value in backquotes – for example, `` `20141231` ``.

Non workdays other than Saturday and Sunday

The ability to specify non workdays other than Saturday and Sunday allows you to use the WORKDAY( ) function with data that is not based on a Monday-to-Friday work week, or on a five-day work week.

For example, if you specify `"Sun"` by itself as a non workday, you create a six-day work week from Monday to Saturday.

Accounting for statutory holidays

The WORKDAY( ) function does not take into account statutory holidays, which means that the return value may not reflect the actual number of workdays in a period if the period contains one or more statutory holidays.

"Calculate Working Days" script in ScriptHub

If you need to account for statutory holidays, one option is to use the Calculate Working Days script in ScriptHub, which accepts a list of user-defined holidays.

For data that covers longer time periods and includes a number of holidays, using the script is probably the easier approach. For more information, see "Importing from ScriptHub" on page 1503.

For shorter time periods with only three or four holidays, such as a quarter, you may find creating the conditional computed field described below is not too laborious.

Conditional computed field for deducting statutory holidays

If required, you can create a conditional computed field to deduct statutory holidays from the value returned by the WORKDAY( ) function.

For example, for first-quarter data for 2015, you could decrement the WORKDAY( ) return value by 1 for each of these holidays that falls into a specified period:

- 01 January 2015
- 19 January 2015
- 16 February 2015

The example that follows accommodates periods that have any start date and end date during the quarter.

You first create a computed field, for example **Workdays**, that calculates the workdays for a specified period during the quarter:

```
DEFINE FIELD Workdays COMPUTED WORKDAY(Start_date, End_date)
```

You then create a conditional computed field, for example **Workdays_no_holidays**, that adjusts the value returned by the first computed field (**Workdays**):

```
DEFINE FIELD Workdays_no_holidays COMPUTED

Workdays-1 IF Start_date = `20150101` AND End_date < `20150119`
Workdays-2 IF Start_date = `20150101` AND End_date < `20150216`
Workdays-3 IF Start_date = `20150101` AND End_date <= `20150331`
Workdays   IF Start_date < `20150119` AND End_date < `20150119`
Workdays-1 IF Start_date < `20150119` AND End_date < `20150216`
Workdays-2 IF Start_date < `20150119` AND End_date <= `20150331`
Workdays-1 IF Start_date = `20150119` AND End_date < `20150216`
Workdays-2 IF Start_date = `20150119` AND End_date <= `20150331`
Workdays   IF Start_date < `20150216` AND End_date < `20150216`
Workdays-1 IF Start_date < `20150216` AND End_date <= `20150331`
Workdays-1 IF Start_date = `20150216` AND End_date <= `20150331`
Workdays   IF Start_date < `20150331` AND End_date <= `20150331`
Workdays
```

> **Note**
>
> The order of the conditions in the conditional computed field is important.
>
> Analytics evaluates multiple conditions starting at the top. The first condition that evaluates to true for a record assigns the value of the conditional computed field for that record. A subsequent condition that evaluates to true does not change the assigned value.

YEAR( ) function

Extracts the year from a specified date or datetime and returns it as a numeric value using the YYYY format.

Syntax

```
YEAR(date/datetime)
```

Parameters

Name	Type	Description
date/datetime	datetime	The field, expression, or literal value to extract the year from.

Output

Numeric.

Examples

Basic examples

Returns 2014:

```
YEAR(`20141231`)
```

```
YEAR(`141231 235959`)
```

Returns the year for each value in the **Invoice_date** field:

```
YEAR(Invoice_date)
```

Remarks

Parameter details

A field specified for *date/datetime* can use any date or datetime format, as long as the field definition correctly defines the format.

Specifying a literal date or datetime value

When specifying a literal date or datetime value for *date/datetime*, you are restricted to the formats in the table below, and you must enclose the value in backquotes – for example, `` `20141231` ``.

Do not use any separators such as slashes (/) or colons (:) between the individual components of dates or times.

- **Datetime values** – you can use any combination of the date, separator, and time formats listed in the table below. The date must precede the time, and you must use a separator between the two. Valid separators are a single blank space, the letter 't', or the letter 'T'.
- **Time values** – you must specify times using the 24-hour clock. Offsets from Coordinated Universal Time (UTC) must be prefaced by a plus sign (+) or a minus sign (-).

Example formats	Example literal values
YYYYMMDD	`20141231`
YYMMDD	`141231`
YYYYMMDD hhmmss	`20141231 235959`
YYMMDDthhmm	`141231t2359`
YYYYMMDDThh	`20141231T23`
YYYYMMDD hhmmss+/-hhmm (UTC offset)	`20141231 235959-0500`
YYMMDD hhmm+/-hh (UTC offset)	`141231 2359+01`
Note Do not use hh alone in the main time format with data that has a UTC offset. For example, avoid: hh+hhmm. Results can be unreliable.	

ZONED( ) function

Converts numeric data to character data and adds leading zeros to the output.

> **Note**
> You can also use the LEADINGZEROS( ) function to add leading zeros, and in many cases it is easier to use than ZONED( ). See "LEADINGZEROS( ) function" on page 2208.

Syntax

```
ZONED(number, length)
```

Parameters

Name	Type	Description
number	numeric	The numeric value to convert to a string. **Note** If you want to add leading zeros to a character value that contains a numeric string, you must use the VALUE( ) function to convert the character to the numeric data type before using the value as input for ZONED( ). For more information, see "VALUE( ) function" on page 2438.
length	numeric	The length of the output string.

Output

Character.

Examples

Basic examples

Integer input

Returns "235":

```
ZONED(235, 3)
```

Returns "00235", because *length* is greater than the number of digits in *number* so two leading zeros are added to the result:

```
ZONED(235, 5)
```

Returns "35", because *length* is less than the number of digits in *number* so the leftmost digit is truncated from the result:

```
ZONED(235, 2)
```

Decimal input

Returns "23585", because the zoned data format does not support a decimal point:

```
ZONED(235.85, 5)
```

Negative input

Returns "64489M", because the number is negative and "M" represents the final digit 4:

```
ZONED(-6448.94, 6)
```

Returns "489J", because *length* is less than the number of digits in *number* so the two leftmost digits are truncated from the result, and the number is negative and "J" represents the final digit 1:

```
ZONED(-6448.91, 4)
```

Advanced examples

Adding leading zeros to a character field containing numbers

The `Employee_Number` field contains the value `"254879"`. You need to convert the value to a 10-digit string with leading zeros.

> **Tip**
> You must use the VALUE( ) function to convert the character to numeric data before using the numeric as input for ZONED( ).

```
COMMENT returns "0000254879"
ASSIGN v_str_length = 10
ASSIGN v_num_decimals = 0
ZONED(VALUE(Employee_Number, v_num_decimals), v_str_length)
```

Harmonizing a key field when joining tables

You have two tables, **Ar** and **Customer**, and you need to join them on the **CustNo** field for further analysis. The two tables each have a **CustNo** field, but the data format is different:

- **Ar** - numeric field (for example, 235)
- **Customer** - 5 character field that pads numbers with leading zeros (for example, "00235")

To harmonize the fields when joining so that the data types and lengths are equal, you use the ZONED( ) function to convert the **Ar** key field **CustNo** to a character field of length 5 so that it matches the format of the key field in **Customer**:

```
OPEN Ar PRIMARY
OPEN Customer SECONDARY
JOIN PKEY ZONED(CustNo,5) FIELDS CustNo Due Amount SKEY CustNo UNMATCHED
TO Ar_Cust OPEN PRESORT SECSORT
```

Remarks

How it works

This function converts numeric data to character data and adds leading zeros to the output. The function is commonly used to harmonize fields that require leading zeros, for example, check number, purchase order number, and invoice number fields.

When to use ZONED( )

Use the function to convert a positive numeric value to a character value containing leading zeros. This is useful for normalizing data in fields to be used as key fields.

For example, if one table contains invoice numbers in the form 100 in a numeric field, and another table contains invoice numbers in the form "00100" in a character field, you can use ZONED( ) to convert the numeric value 100 to the character value "00100". This allows you to compare like invoice numbers.

String length and return values

Leading zeros are added to the output value when the *length* value is more than the number of digits in *number*. When *length* is less than the number of digits in *number*, the output is truncated from the left side. If the *number* value is the same length as *length*, then no zeros are added.

Decimal numbers

The zoned data format does not include an explicit decimal point.

Negative numbers

If the input *number* is negative, the rightmost digit is displayed as a character in the result:

- "}" for 0
- a letter between "J" and "R" for the digits 1 to 9

ZONED( ) and the Unicode edition of Analytics

If you are working with the Unicode edition of Analytics, you need to use the BINTOSTR( ) function to correctly display the return value of the ZONED( ) function. You also need to use the BINTOSTR( ) function if you want to use the return value of the ZONED( ) function as a parameter in another function.

ZSTAT( ) function

Returns the standard Z-statistic.

Syntax

```
ZSTAT(actual, expected, population)
```

Parameters

Name	Type	Description
actual	numeric	◦ **When specifying parameters as numbers** - represents the actual count, such as a leading digit or a leading digit combination. ◦ **When specifying parameters as proportions** - represents the expected proportion of the value being tested and must be between 0 and 1 inclusive (i.e., greater than or equal to 0 and less than or equal to 1).
expected	numeric	◦ **When specifying parameters as numbers** - represents the expected count, such as a leading digit or a leading digit combination. ◦ **When specifying parameters as proportions** - represents the expected proportion of the value being tested and must be between 0 and 1 exclusive (i.e., greater than 0 and less than 1).
population	numeric	The total number of items being tested. This parameter must be a positive whole number greater than 0.

Output

Numeric.

Examples

Advanced examples

Parameters expressed as numbers

Based on 10 years of previous data, you know that the distribution of worker disability claims per month is normally highly uniform. In April, May, and June of this year, claims were higher by about 10 percent, averaging 220 per month instead of 200. Claims in July and August were slightly low, at 193 and 197. The total claims for the year were 2,450. To test whether these high and low results were significant, use the Z-statistic.

The actual number of claims for April to June is higher than expected at 660. The expected number of claims for this period is 25 percent of the 2,450 annual claims, or 612.5. The Z-statistic for these counts is calculated as 2.193:

```
ZSTAT(660, 612.5, 2450)
```

A Z-statistic of 1.96 has a significance of 0.05, and 2.57 has a significance of 0.01. Thus, the probability that the higher rates of claims are due to chance is between 1:20 and 1:100.

The actual number of claims for July and August is lower than expected at 390. The expected number of claims for this period is one sixth of the 2,450 annual claims, or 408.33. The Z-statistic for these proportions is calculated as 0.967:

```
ZSTAT(390, 408.33, 2450)
```

This is not a very significant result. Z-statistics of 1.000 and less are very common and can typically be ignored.

Parameters expressed as proportions

Based on 10 years of previous data, you know that the distribution of worker disability claims per month is normally highly uniform. In April, May, and June of this year, claims were higher by about 10 percent, averaging 220 per month instead of 200. Claims in July and August were slightly low, at 193 and 197. The total claims for the year were 2,450. To test whether these high and low results were significant, use the Z-statistic.

The actual number of claims for April to June is represented by the proportion 660/2450, which is higher than expected. The expected number of claims for this period should be 25 percent of the 2,450 annual claims. The Z-statistic for these proportions is 2.193:

```
ZSTAT((1.00000000 * 660 / 2450), 0.25, 2450)
```

A Z-statistic of 1.96 has a significance of 0.05, and 2.57 has a significance of 0.01. Thus, the probability that the higher rates of claims are due to chance is between 1:20 and 1:100.

The actual number of claims for July and August is low at 390. The expected number of claims for this period should be one sixth, or 16.6667 percent of the 2,450 annual claims. The Z-statistic for these proportions is 0.967:

```
ZSTAT((1.00000000 * 390 / 2450), 0.16667, 2450)
```

This is not a very significant result. Z-statistics of 1.000 and less are very common and can typically be ignored.

Remarks

How it works

The ZSTAT( ) function calculates the standard Z-statistic for use in many problem-solving tasks, including digital analysis. It outputs the result with a precision of three decimal places.

Using ZSTAT( )

Use ZSTAT( ) to evaluate the likely frequency of occurrence of a given result in a specified period or category. The larger the resulting Z-statistic, the more unlikely the occurrence.

For example, a Z-statistic of 1.96 has a significance of 0.05, representing the likelihood of a one time in 20 occurrence, whereas a Z-statistic of 2.57 has a significance of 0.01, representing the likelihood of a one time in 100 occurrence. For information on the Z-statistic, consult a statistics textbook.

Specifying input for ZSTAT( )

You can specify the parameters for ZSTAT( ) as either numbers or proportions:

- When you specify both input values as numbers, the function computes the Z-statistic using floating-point arithmetic
- When you specify both input values as proportions, the function computes the Z-statistic using fixed-point arithmetic, and you need to use a decimal multiplier to control rounding

- When using an expression within an expression to calculate the *actual* or *expected* value, you must specify the level of precision you want in the result by using a decimal multiplier. Analytics has a precision of 8 digits, therefore a multiplier of 1.00000000 will return the greatest precision attainable

This page intentionally left blank

Analytic scripts overview

Scripts are not limited to running in Analytics only. By converting regular scripts to **analytic scripts**, you can schedule and run scripts in the Robots app on the HighBond platform, or in Analytics Exchange. You can also run analytic scripts in the Analysis App window, a freestanding component of Analytics.

What are analytic scripts?

An analytic script, or "an analytic", is a regular script **with an analytic header**. The analytic header is a series of declarative tags that allow the script to run in Robots, on AX Server, or in the Analysis App window. The analytic header includes input parameters that a user populates in advance, which allows the analytic script to run unattended, either immediately, or at a scheduled time.

> **Tip**
> Analytic scripts are almost exclusively developed and tested in Analytics, which supports easier development. Use AX Client to make simple updates to existing analytic scripts that are stored on AX Server.

What are analysis apps?

An **analysis app** is an Analytics project that is packaged for use in Analytics Exchange or the Analysis App window. Analysis apps contain one or more analytic scripts, and can also contain data tables and interpretations.

> **Note**
> Analysis apps are typically created or developed by an organization's in-house scripting experts, or by arrangement with Galvanize consultants.

Turning regular scripts into analytic scripts

Analytic scripts begin as regular scripts. To run a regular script in Robots, on AX Server, or in the Analysis App window, you must convert the regular script into an analytic script:

1. Create and test a script in Analytics.
2. Add the appropriate analytic header tags to make the script an analytic script.
3. Package the analytic script to run on AX Server or in the Analysis App window. You do not package analytic scripts run in Robots.

For more information, see "Developing analytic scripts" on page 2464.

Adding analytic headers

Analytic headers are defined in a comment block that starts on the first line of the script. At a minimum, an analytic header declares the script is an analytic script:

```
COMMENT
//ANALYTIC Identify missing checks
This analytic script identifies missing check numbers
END
```

For more information, see "Working with analytic headers" on page 2473.

What are auxiliary scripts?

An auxiliary script is a regular script **without an analytic header** that is designed to work in conjunction with an analytic script. In a typical design, an analytic script uses the DO SCRIPT command to call one or more auxiliary scripts. Once an auxiliary script completes, processing returns to the analytic script, which continues to execute.

Auxiliary scripts can also be referred to as subscripts, secondary scripts, utility scripts, or helper scripts. You do not have to use auxiliary scripts. They offer an option for compartmentalizing blocks of script logic that might be conditional, reusable, or simply unwieldy to include in the parent analytic script.

Auxiliary script restrictions

Auxiliary scripts can be used in a variety of different ways, but because no analytic header exists, two restrictions apply:

- **No input or output tags** – you cannot specify input or output analytic tags, which means you cannot create input or output parameters in the auxiliary script itself. Any required parameters must be created in the analytic header in the parent analytic script.
- **Cannot be run directly** – users cannot schedule or run auxiliary scripts directly. They can only be called from an analytic script – either directly, or indirectly through another auxiliary script.

Distributing and running analytic scripts

There are several options for distributing and running analytic scripts, depending on which Galvanize products and components your organization uses.

App/product/component	Method for distributing and running an analytic script
Robots	◦ commit one or more analytic scripts, and any auxiliary scripts, as a **script version** to development mode in Robots, and schedule and run an activated script version in production mode
AX Server	Either of these methods: ◦ import the Analytics project (`.acl` file) directly into AX Server, and schedule and run an analytic script using AX Client ◦ package the project into a compressed analysis app file (`.aclapp` file), import it into AX Server, and run an analytic script using AX Web Client For more information, see "Packaging analysis apps for import to AX Server" on page 2487.
Analysis App window	◦ package the project into a compressed analysis app file (`.aclapp` file), open the project as an analysis app (`.aclx` file), and run the analytic script in the Analysis App window For more information, see "Packaging analysis apps for use in the Analysis App window" on page 2649.

Determining the environment where an analytic script is running

If you want to create an analytic script that can run in Analytics, Analytics Exchange, or the Analysis App window, you can determine the runtime environment during script execution. You can use this information to make decisions about which commands to run based on where the script is running.

Use the FTYPE( ) function to determine where the script is running:

```
FTYPE("ax_main") = "b"
```

If the script is running in either Analytics Exchange or the Analysis App window, the expression evaluates to true (T). For scripts running in Analytics, the expression evaluates to false (F). For more information, see "FTYPE( ) function" on page 2157.

Identifying the user running the script on AX Server

For analytic scripts run on AX Server, you can use the system variable *AXRunByUser* to identify the name of the user who is currently running the script, in the format *domain**username*:

```
EXTRACT FIELDS TIME() AS "Time", DATE() AS "Date", AXRunByUser AS "Current
User" TO R_RunRecord APPEND
```

> **Note**
> *AXRunByUser* is only available when running analytic scripts on AX Server. The variable is unrecognized when running scripts in Analytics.

Developing analytic scripts

The recommended method for developing an analytic script is to first create and test a regular script in Analytics. Once the script is working correctly, add the analytic header to convert the script to an analytic script. Analytic scripts can run in Robots, on AX Server, or in the Analysis App window.

For information about creating regular scripts, see "Get started with scripting" on page 1378.

Identify script inputs and outputs

In the analytic header, you use **analytic tags** to declare script inputs, and any script outputs that you want to make available to end users, or use as input for subsequent scripts. The different types of inputs and outputs are described below, with the associated analytic tags in parentheses.

> **Tip**
> Identifying the required inputs and outputs before you begin will make development go more smoothly.

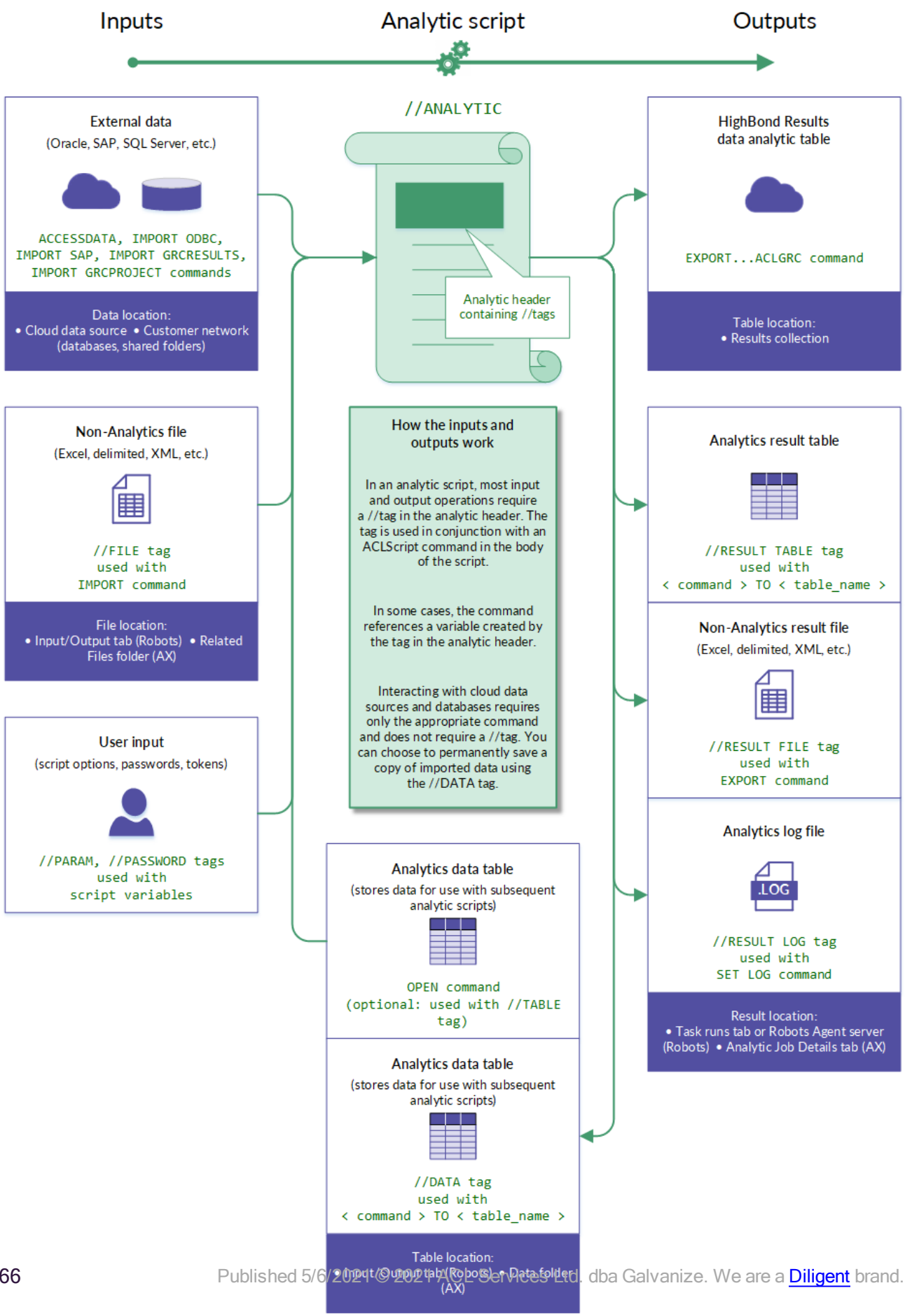
Inputs	Outputs
◦ non-Analytics files such as Excel or delimited ("FILE tag" on page 2514) ◦ input parameters such as cutoff amount, date, or ID codes ("PARAM tag" on page 2517) ◦ passwords ("PASSWORD tag" on page 2530) ◦ Analytics tables and fields ("TABLE tag" on page 2533, "FIELD tag" on page 2535)	◦ Analytics and non-Analytics results tables ("RESULT tag" on page 2538) ◦ log files for successful scripts ("RESULT tag" on page 2538) ◦ Analytics output tables that will be used as input for subsequent scripts ("DATA tag" on page 2545)

Script inputs and outputs – the big picture

The diagram below illustrates all the possible inputs to, and outputs from, an analytic script. Each input or output shows:

- the type of data or user input involved
- the associated analytic tag (if applicable)
- the associated ACLScript command or commands
- the location of the input or output data, tables, or files

Depending on its design and purpose, a single analytic script might have multiple different inputs and outputs.



Accessing source data

There are two basic approaches to accessing the source data required by an analytic script:

- Automated connectivity
- Manual upload

You are free to use both approaches in the same analytic script, if required.

Automated connectivity

The advantage of this approach is that data imports to Robots or AX Server can be fully automated including being run on a schedule.

In the body of the analytic script, use one of the ACLScript commands for connecting to an external data source, importing data, and creating an Analytics table with a copy of the data:

- ACCESSDATA
- IMPORT ODBC
- IMPORT GRCRESULTS
- IMPORT GRCPROJECT
- DEFINE TABLE DB

> **Note**
>
> These commands do not require any corresponding analytic tag in the analytic header.
>
> Use ACCESSDATA unless you have a reason for using one of the other commands. DEFINE TABLE DB is an older command that is maintained for backward compatibility with older scripts.

Manual upload

Manual upload provides a simple way to import data to Robots or AX Server, and it may be appropriate when users have source data files stored locally.

Robots

You can manually upload non-Analytics files such as Excel or delimited to Robots. You need to use a different method to make Analytics tables available.

- **non-Analytics files** – You can manually upload non-Analytics files such as Excel or delimited to the **Input/Output** tab in a robot. To access the uploaded data in an analytic script, use a FILE tag in the analytic header, and an appropriate import command, such as IMPORT EXCEL, in the body of the script.

- **Analytics tables** – You cannot manually upload Analytics tables to the **Input/Output** tab. Instead, use a DATA tag in the analytic header to save an Analytics output table to the **Input/Output** tab. To access the Analytics table in a subsequent script, use the OPEN command in the body of the script.

AX Server

You can manually upload non-Analytics files such as Excel or delimited, and Analytics tables, to AX Server:

- **non-Analytics files** – You can import files such as Excel and delimited to the **Related Files** subfolder. To access the imported data in an analytic script, use a FILE tag in the analytic header, and an appropriate import command, such as IMPORT EXCEL, in the body of the script.
- **Analytics tables** – When you import an Analytics project into AX Server, the project's tables are imported to the **Data** subfolder. To access an imported table in an analytic script, use the OPEN command in the body of the script.

Workflow for creating and testing an analytic script

> **Note**
> The following workflow is a suggested approach for developing analytic scripts, however you are free to develop analytic scripts in whatever manner best suits you.

Create the Analytics script

Create a script in Analytics without using any custom dialog boxes for user input, or any other features that require user interaction during the running of the script. Analytic scripts allow user input prior to running the script, but unlike regular scripts do not support user interaction during script execution.

To store test input values in the Analytics script, temporarily create variables at the top of the script. For example:

```
ASSIGN v_AnalysisTable = "Trans_May"
```

Test and debug the script until it executes without errors.

Add the analytic header and tags

Add an analytic header to the script. Copy the variable names from the top of the script to the corresponding tags in the Analytic Header Designer.

An example of a resulting tag in the analytic header:

```
//TABLE v_AnalysisTable "Table to classify"
```

For more information, see "Working with analytic headers" on page 2473.

Include the log in the analytic script results

The log is a crucial tool for diagnosing the cause of analytic script failures. It can also be important when analytic scripts succeed but give unexpected results. The log is automatically output when an analytic script fails, but it is only output when an analytic script succeeds if you specify the `RESULT` analytic tag.

In the Analytic Header Designer, turn on **Keep log file** to ensure that a log is available every time the analytic script is run. The corresponding tag is added to the analytic header:

```
//RESULT LOG
```

Validate the analytic header

Validate the analytic header. You can validate the analytic header as frequently as you want.

For more information, see "Validating analytic headers" on page 2479.

Assign temporary test values to the analytic tags

Using the special assignment operator (`:=`), assign temporary test values to all analytic tags that require user input. You can copy the test values from the temporary variable assignments at the top of the script. For example:

```
//TABLE v_AnalysisTable "Table to classify" := "Trans_May"
```

To assign temporary test values using the Analytic Header Designer, enter the value in the **Test value** field for all analytic tags that require user input.

For more information about assigning temporary test values, see "Specifying test input values in Analytics" on page 2506.

Delete the temporary variables

Delete the temporary variables from the top of the script, or comment them out if you think you might want to use them again.

Step through the analytic script

Step through the analytic script by clicking **Step**  , or by pressing **F10** repeatedly. Review the contents of the **Variables** tab in the **Navigator** to ensure that all variables in the analytic header are being created correctly, with the correct assignment of test values.

Test and debug the analytic script until it executes without errors.

> **Note**
> If you want to exit the analytic script before it completes, press **Esc** and click **Yes** in the confirmation prompt.

> **Tip**
> You can delete all stored variables and variable assignments from the Analytics project by entering `DELETE ALL OK` in the command line. Clearing the **Variables** tab prior to stepping through an analytic script gives you a clean start.

Delete the temporary test values

When you are finished testing, you can delete the temporary test values and the special assignment operator from all analytic tags. Or you can choose to keep them if you anticipate that additional testing might be required. Test values are ignored in deployment environments.

Deploy the analytic script

To deploy the analytic script to the target environment, commit the script to Robots, or import the Analytics project into AX Server.

To deploy the analytic script, commit the script to Robots.

Workflow for testing an analysis app

For analytic scripts that will be run in AX Web Client, or the Analysis App window, you also need to test the analysis app.

Delete redundant table layouts

Once all analytic scripts and any subscripts in the analysis app are tested, debugged, and executing correctly, delete any table layouts from the Analytics project that you are not going to include in the analysis app.

Redundant table layouts will clutter the analysis app in AX Client, AX Web Client, and the Analysis App window, and could be confusing to end users.

Open the analysis app in the Analysis App window

Open the completed analysis app in the Analysis App window by right-clicking the Analytics project entry in the **Overview** tab and selecting **Open as Analysis App**.

> **Note**
> If the analysis app fails to open and you get an error message stating that analytic scripts have identical names, check the *name* value in the `ANALYTIC` tag for each analytic script specified in the error message. The *name* values of analytic scripts must be unique in an Analytics project.

Run the analytic scripts

Run all the analytic scripts in the analysis app to confirm that they are functioning correctly.

Observe the correct order for running the analytic scripts if you are using the `TYPE` option with the `ANALYTIC` tag and creating import, preparation, and analysis analytic scripts.

Check the log

If an analytic script fails, open and review the log file (*analytic_name*.log). The log should include an entry, marked with a red X, that indicates why the analytic script failed:

- for incorrectly entered input values, immediately re-run the analytic script with a correctly entered input value
- for syntax or logical errors in the script body, correct the error in Analytics, and then reopen the analysis app in the Analysis App window

An analytic script may succeed but the result table may not contain the results you were expecting. In this situation, review the log entries in sequence, and check the input values that have been passed to the analytic script, to ensure that the analytic script is functioning in the manner you intended.

Packaging and validating an analysis app

Package or import the analysis app

Once you are satisfied the analysis app is working as intended, package it for distribution and use in the Analysis App window, or import it to AX Server for use in AX Client or AX Web Client. For more information, see either:

- "Packaging analysis apps for use in the Analysis App window" on page 2649
- "Packaging analysis apps for import to AX Server" on page 2487

Run AX Server analysis apps

If you are developing analytic scripts for use in AX Server, run all analytic scripts using both AX Client and AX Web Client to ensure they work as intended.

Working with analytic headers

An **analytic header** is a series of analytic tags enclosed in a comment block at the beginning of a script. The tags specify either script inputs or script outputs.

For a visual overview of analytic script inputs and outputs, and associated analytic tags, see "Script inputs and outputs – the big picture" on page 2464.

Tags that specify user-facing input parameters let a user specify script input values in advance, which means an analytic script can run unattended, either immediately, or at a scheduled time.

After you develop scripts in an Analytics project, you must add an analytic header to at least one script before you can commit the scripts to Robots, or use the project as an analysis app on AX Server or in the Analysis App window.

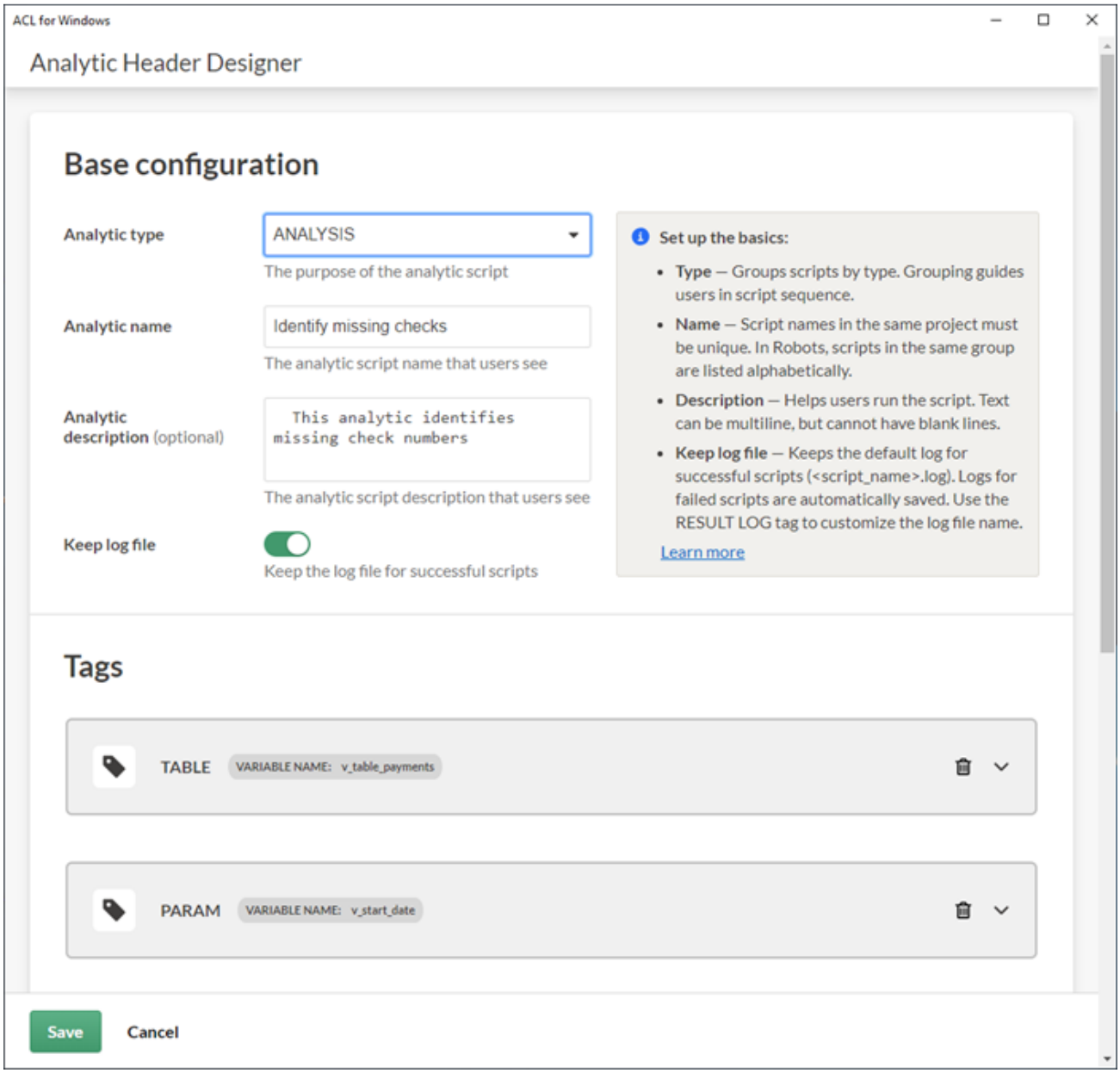
Using the Analytic Header Designer is the easiest way to add or modify an analytic header. You can also add or modify an analytic header manually.

The Analytic Header Designer

The Analytic Header Designer has an intuitive interface for progressively adding the analytic tags that make up an analytic header. You are free to add, modify, or delete tags as you build an analytic header.

Automated error checking and embedded guidance in the Designer help ensure that the header you build is valid and works correctly.

When you click **Save** in the Designer, the tags that you have configured are automatically translated into an analytic header at the top of the script. You can manually edit the analytic header if you want to, but the recommended approach is to reopen the Designer to perform edits.



Sample analytic header

The analytic header shown below, for an analytic script that identifies missing checks, was created using the tags shown in the Analytic Header Designer above. To save space, the screen capture of the Designer is sized to show only a subset of the tags in the analytic header.

```
COMMENT
//ANALYTIC TYPE ANALYSIS Identify missing checks
  This analytic script identifies missing check numbers
```

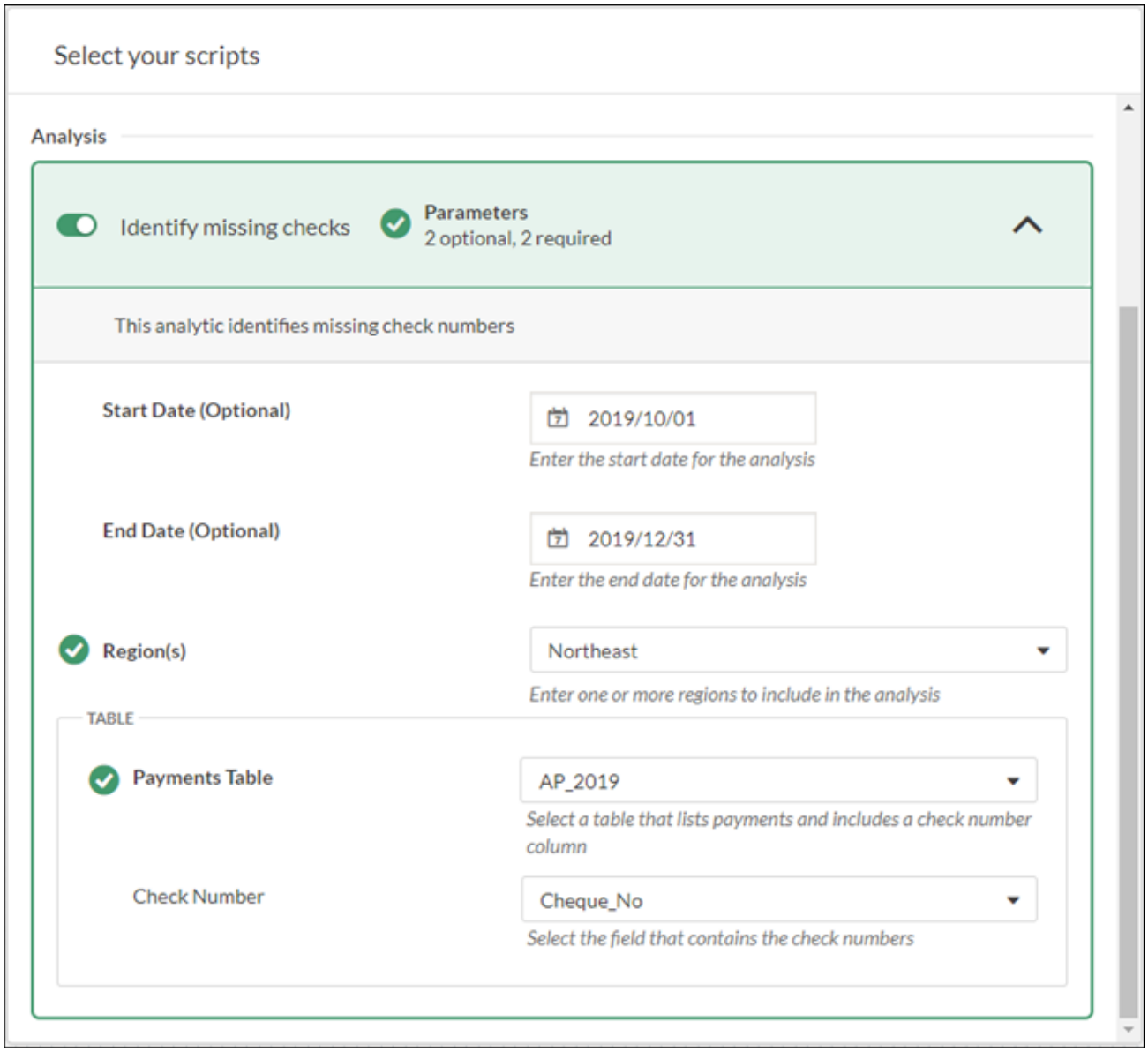
```
//TABLE v_table_payments Payments Table
  Select a table that lists payments and includes a check number column
//FIELD v_check_num CN Check Number
  Select the field that contains the check numbers
//PARAM v_start_date D OPTIONAL Start Date (Optional)
  Enter the start date for the analysis
//PARAM v_end_date D OPTIONAL End Date (Optional)
  Enter the end date for the analysis
//PARAM v_region C MULTI SEPARATOR , QUALIFIER ' VALUES |North-
east|Southeast|Central|Western|West Coast| Region(s)
  Enter one or more regions to include in the analysis
//RESULT TABLE Missing_Checks
//RESULT FILE Missing_Checks.xls
//RESULT LOG
END

COMMENT The body of the script starts here.
SET SAFETY OFF
OPEN %v_table_payments%
.
.
.
SET SAFETY ON
```

What the script inputs look like in a client application

The input tags in the sample analytic header above create input parameters that a user must populate when scheduling or running the analytic script in a client application.

The way the input parameters are displayed in Robots is shown below.



What each tag does

Each analytic tag in the sample analytic header above performs a specific task when a user schedules or runs the associated analytic script in a client application.

Analytic header syntax	Description
`COMMENT . . . END`	Encloses the block of analytic tags. Every analytic header must be enclosed by a `COMMENT` command that starts on the first line of the script.

Analytic header syntax	Description
`//ANALYTIC`	Creates the base configuration of the analytic header, including the type and name of the analytic script. Every analytic header must start with an `//ANALYTIC` tag.
`//TABLE v_table_payments`	Creates an input parameter that allows a user to select a payments table. Because table names vary, the name of the table selected by the user is stored in the `v_table_payments` variable.
`//FIELD v_check_num`	Creates an input parameter that allows a user to select a check number field from the payments table. Because field names vary, the name of the field selected by the user is stored in the `v_check_num` variable.
`//PARAM v_start_date`	Creates an input parameter that allows a user to specify a start date for the range of records being analyzed. Because users will specify different start dates, the actual date specified by the user is stored in the `v_start_date` variable.
`//PARAM v_end_date`	Creates an input parameter that allows a user to specify an end date for the range of records being analyzed. Because users will specify different end dates, the actual date specified by the user is stored in the `v_end_date` variable.
`//PARAM v_region`	Creates an input parameter that allows a user to specify which region or regions are included in the analysis. Because users will specify different regions, the actual regions specified by the user are stored in the `v_region` variable.
`//RESULT TABLE Missing_Checks`	Creates an output parameter that specifies the Missing_Checks results table is made available to users in client applications. Output results from scripts, even if they exist, are not automatically made available. Availability must be specified in the analytic header.
`//RESULT FILE Missing_Checks.xls`	Creates an output parameter that specifies the Missing_Checks.xls results file is made available to users in client applications.

Analytic header syntax	Description
	Output results from scripts, even if they exist, are not automatically made available. Availability must be specified in the analytic header.
`//RESULT LOG`	Specifies that a log file is output for scripts that run successfully. A log file is automatically output if a script fails.

Build an analytic header

To build an analytic header you must know in advance what script inputs and outputs you need. For more information, see "Identify script inputs and outputs" on page 2464.

Set up the base configuration of the analytic header

1. Open a new or existing script in the Script Editor.
2. Click **Edit Analytic Header**  .

   The Analytic Header Designer opens.
3. Select an **Analytic type**.

   Analytic scripts are grouped by type in Robots, AX Web Client, and the Analysis App window. Grouping guides users in script sequence.

   - `IMPORT` - a script that retrieves data from a data source.
   - `PREPARE` - a script that transforms raw data in whatever way is necessary to make it suitable for analysis.
   - `ANALYSIS` - a script that performs analysis on data.
4. Specify an **Analytic name**.

   > **Note**
   >
   > Names of analytic scripts in the same Analytics project must be unique.
   >
   > The name identifies the analytic script in client applications. The analytic script name is not the same as the script name that you specify in Analytics when you initially create the script.

5. Choose whether to keep a log file for successful scripts:
   - **Keep log file on** - a log file is automatically output when the script runs successfully
   - **Keep log file off** - a log file is not output when the script runs successfully

   Regardless of the **Keep log file** setting, a log file is automatically output if the script fails.

> **Tip**
> If you want to customize the name of the log file for successful scripts, use the `RESULT LOG` tag.

Add additional analytic tags

After setting up the base configuration of the analytic header, you can add as many additional analytic tags as you need.

You can add tags in any order.

1. In the Analytic Header Designer, click **Add tag**.
2. Select a **Tag type**.
3. To configure the tag, complete all the required fields in the tag configuration section, and any optional fields that you need.

   Guidance for configuring tags is embedded in the configuration section for each tag.

   For detailed information about analytic header syntax, and a full list of analytic tags, see "Analytic headers and tags" on page 2505.
4. Repeat the process for each additional tag that you need in the analytic header.
5. Click **Save** when you are finished.

Validating analytic headers

After you add an analytic header to one or more scripts, use tools in Analytics to validate the header syntax to ensure that it is correct. Perform the validation before committing scripts to Robots, or packaging analysis apps, so that the analytic scripts do not fail when they run.

One tool validates individual analytic headers at the script level. The other tool validates all the analytic headers in a project at once. The two types of validation focus on different things.

Validate an individual analytic header

Script-level validation of an analytic header focuses on the syntax of individual analytic tags and reports errors with accompanying line numbers.

1. Open the script containing the analytic header.
2. On the Script Editor toolbar, click **Validate Analytic Header**  .

   A message appears telling you that the analytic header is valid, or specifying an error and the line number where the error occurs.
3. If the analytic header contains an error, correct the error and click **Validate Analytic Header**  again to ensure that there are no further errors.

> **Tip**
> If the nature of the error is not apparent based on the error message, review the Help topic for the associated analytic tag. Carefully compare the syntax in the topic with the syntax in the line of the analytic header. Errors can be caused by minor discrepancies in the analytic header syntax.

Validate all the analytic headers in a project

Project-level validation of the analytic headers checks two things:

- at least one analytic header exists in the project
- names of multiple analytic scripts are unique

> **Note**
> The name of the analytic script is the name specified in the `ANALYTIC` tag, not the script name in the **Overview** tab in the **Navigator**.

Project-level validation is performed automatically when you commit scripts to Robots. You can also perform the validation manually if you add the **Check Scripts**  button to the Analytics toolbar.

1. If necessary, add the **Check Scripts** button to the Analytics toolbar:
   a. Double-click an empty spot on the toolbar to open the **Customize Toolbar** dialog box.
   b. In the **Available toolbar buttons** list, select the **Check Scripts** button and click **Add**.
   c. In the **Current toolbar buttons** list, select the **Check Scripts** button and click **Move Up** or **Move Down** to change the location of the button.

      The order of the buttons from top to bottom corresponds to their location from left to right on the toolbar.

   d. Click **Close** to save your changes.
2. On the toolbar, click **Check Scripts**  .

   A message appears telling you that the analytic headers in the project are valid, or specifying one or more errors.

3. If the analytic headers contain an error, correct the error and click **Check Scripts**  again to ensure there are no further errors.

Analytic development best practices

Analytic scripts support most of the commands that you can use in a regular script. However, you must ensure that analytic scripts run without user interaction, and that they do not include commands not supported by the engine that processes the analytic scripts in the deployment environment or in HighBond's Robots app.

Analytic scripts support all ACLScript functions.

General best practices

Use one Analytics project per robot or analysis app

Create a new Analytics project for each robot or analysis app. The project must contain all the analytic scripts that make up the robot or the analysis app, and any required subscripts.

For an analysis app, the project must also contain any Analytics tables required by any of the analytic scripts.

Test locally

Test all analytic scripts locally before deploying them to the target environment or the Robots app. Ensure that analytic scripts run as expected, and that they do not require user interaction.

For more information, see "Developing analytic scripts" on page 2464.

Use consistent data connections for testing

To test an analytic script locally if it uses an ODBC data source, you must configure an ODBC connection on your local computer that is identical to the connection in the environment where the analytic script will run.

For analytic scripts distributed for use in the Analysis App window, end users must configure an identical ODBC connection on their computers.

Avoid absolute file paths

Avoid using absolute file paths in analytic scripts (for example, `C:\results`) unless you are certain that identical file paths exist in the environment where the analytic script will run.

Using relative file paths such as `\results` allows you to develop and test analytic scripts locally and then deploy them in another environment without requiring that the other environment has an identical directory structure.

Use SET for preference settings

Use the SET command to specify any preference settings required by the analytic script. If you do not specify preferences in the analytic script, the default Analytics preferences are used. Position the SET command after the analytic header but before any of the analytic script logic.

Do not use computed fields in results or data output tables

Do not use computed fields in any output tables that you intend to keep beyond the session in which the analytic script runs.

Results and data tables that are kept for use in interpretations or as input for subsequent scripts may display unexpected values if they contain computed fields. Computed values are dependent on settings defined in the preference file (.prf), or by the SET command, and therefore different environments may produce different values.

If you need to retain the values in a computed field, use the EXTRACT command with the FIELDS or ALL option to convert the field to a physical field in a result or data table. For more information, see "EXTRACT command" on page 1712.

Encrypt data connection passwords

To avoid having a data source password in plain text in an analytic script, use the PASSWORD analytic tag. This tag prompts the user for a password before running the analytic script, and encrypts the entered value.

Use a password when importing from or exporting to HighBond

The PASSWORD parameter is required in any command that imports from or exports to HighBond:

- IMPORT GRCRESULTS
- IMPORT GRCPROJECT
- EXPORT... ACLGRC

Without the PASSWORD parameter, the commands fail in Robots, Analytics Exchange, or the Analysis App window.

When you use the PASSWORD parameter in a command, you must also specify an associated password tag in the analytic header. For more information, see "PASSWORD tag" on page 2530.

> **Note**
> The `PASSWORD` parameter is not required when running the import and export commands in Analytics because the current user's HighBond access token is automatically used.

Avoiding user interaction

Analytic scripts must be able to run without user interaction. If a command in an analytic script tries to create a dialog box, the engine in the deployment environment stops processing the analytic script, and an error is entered in the log.

Replace user interaction commands with analytic tags

Do not use Analytics commands that require user interaction. Replace them with equivalent analytic tags in the analytic header. Analytic tags allow users to provide input values before the analytic script runs.

Do not use	Replace with
`DIALOG`	`//TABLE`, `//FIELD`, `//PARAM`
`ACCEPT`	`//TABLE`, `//FIELD`, `//PARAM`
`PASSWORD`	`//PASSWORD`
`PAUSE`	no equivalent

Guidelines

- **Interactive commands** – To prevent analytic script processing failures, remove all interactive commands.
- **SET SAFETY** – To ensure files can be overwritten as necessary without displaying a confirmation dialog box, add the `SET SAFETY OFF` command at the beginning of an analytic script.

  Add the `SET SAFETY ON` command at the end of the analytic script to restore the default behavior.
- **OK parameter** – To prevent confirmation dialogs from crashing the analytic script, add the `OK` parameter after any commands that normally display a confirmation dialog box:
  - `RENAME`
  - `DELETE`

Checking script syntax

Analytics provides a tool for detecting script syntax that causes analytic scripts to fail, or that requires alignment between your local environment and the environment where the analytic scripts are deployed. The tool provides a warning only, and you are still free to commit or import analytic scripts that have warnings.

What the tool checks

The tool checks all scripts in a project for the following items:

- any command that requires user interaction
- any absolute file path
- any call of an external script

When the check is performed

Script syntax checking is performed automatically when you commit scripts to Robots.

Automatic syntax checking is enabled by default. If you want to turn it off, select **Disable Script Syntax Check Before Commit Scripts** in the **Options** dialog box (**Tools > Options > Interface**).

Perform checking manually

You can perform script syntax checking manually. You may need to first add the **Check Scripts**  button to the Analytics toolbar.

1. If necessary, add the **Check Scripts** button to the Analytics toolbar:
   a. Double-click an empty spot on the toolbar to open the **Customize Toolbar** dialog box.
   b. In the **Available toolbar buttons** list, select the **Check Scripts** button and click **Add**.
   c. In the **Current toolbar buttons** list, select the **Check Scripts** button and click **Move Up** or **Move Down** to change the location of the button.

      The order of the buttons from top to bottom corresponds to their location from left to right on the toolbar.

   d. Click **Close** to save your changes.
2. On the toolbar, click **Check Scripts**  .

   A message appears telling you that the script syntax in the project is valid, or specifying one or more warnings.
3. Do one of the following:
   - Correct any script syntax that generates a warning, and click **Check Scripts**  again to ensure that the warnings no longer appear.

- Ensure that the deployment environment contains a directory structure, or external scripts, that align with the paths or external scripts specified in the analytic script.

Best practices for analytic scripts run on AX Server

Develop in Analytics

Develop analytic scripts and their supporting scripts primarily in Analytics before importing them to AX Server.

As a convenience feature, the AX Client script editor does allow you to add new analytic scripts or subscripts, or edit existing analytic scripts or subscripts. This feature is useful for fine-tuning the behavior of an analytic script without having to export it to Analytics and then reimport it to AX Server. However, analytic script development work beyond minor adjustments is easier to accomplish in Analytics.

Store related files with the Analytics project

Related files such as database profile files should be stored in the same folder as the Analytics project, but must be imported to AX Server separately.

Avoid commands not supported on AX Server

- direct database server tables linked to Analytics Server Edition for z/OS
- the `NOTIFY` command supports only SMTP messaging. The MAPI and VIM mail protocols are not supported
- to use the `PRINT` or `TO PRINT` command, a default printer must be configured on the server
- the `SAVE GRAPH` and `PRINT GRAPH` commands are not supported
- do not use the `SET LEARN` command in analytic scripts

Minimize AX Server table transactions

Optimize the performance of analytic scripts by minimizing the number of times tables on AX Server are accessed:

1. Use the `FILTER` command to select the records you need.
2. Use the `EXTRACT` command to extract only the required fields.

The reduced data set will be processed locally on the server where the analytic script is being run by the AX Engine.

Optimizing analytic scripts in this way is important when the data files are not located on the same server as AX Server or the AX Engine Node processing the analytic script, and the **Copy analytic data** option is not selected in the AX Server Configuration web application.

Inefficient analytic script example

```
OPEN LargeTable
SET FILTER TO trans_date >= `20091201` AND trans_date < `20100101`
COUNT
TOTAL amount
CLASSIFY ON account ACCUMULATE amount TO TransClassAccount
```

Efficient analytic script example

```
OPEN LargeTable
SET FILTER TO trans_date >= `20091201` AND trans_date < `20100101`
EXTRACT FIELDS trans_date desc account type amount TO AnalysisTable
OPEN AnalysisTable
COUNT
TOTAL amount
CLASSIFY ON account ACCUMULATE amount TO TransClassAccount
```

Access SAP data in background mode

Use Background mode to access data from SAP ERP systems using Direct Link.

Packaging analysis apps for import to AX Server

To run analytic scripts on AX Server, package an Analytics project into an analysis app (an `.aclapp` file).

> **Note**
>
> If at least one script in an Analytics project contains an analytic header, the project can be packaged as an analysis app.
>
> Before packaging an analysis app make sure you validate the analytic header of each analytic script in the Analytics project.

Importing to AX Server

Use a packaged analysis app (`.aclapp`) to prepare an Analytics project for import to AX Server. Along with the analytic scripts in the project, you can choose which tables and data files to import.

Import existing interpretations

You can also use a packaged analysis app (`.aclapp`) to import existing interpretations. To include the interpretations from an existing analysis app (`.aclx`), you merge the Analytics project with the existing `.aclx` file during the creation of the packaged analysis app (`.aclapp`).

> **Note**
>
> When you use an existing analysis app (`.aclx` file), the contents of the Analytics project take precedence. If there are scripts or tables in the `.aclx` file that no longer exist in the `.acl` file, they are not included in the resulting packaged analysis app (`.aclapp` file).

File size limitation

To use a packaged analysis app successfully, you must ensure that the sum of all file sizes included in the package does not exceed 800 MB before packaging the analysis app. If the prepackaged files exceed this combined size, data files may be corrupted when unpacking the analysis app.

Package a new analysis app

1. In Analytics, right-click the project entry in the **Overview** tab of the **Navigator** and select **Package Analysis App**.

   The Analytics project is the top-level folder in the tree view.

2. In the **Select Tables** dialog box, do the following:
   a. If you want to include one or more of the project tables and associated data files in the analysis app, select the table(s) and the data file(s) to include.

      > **Note**
      > Generally you should include only static tables and data files that are required by one or more of the analytic scripts in the analysis app, such as a master vendor table, or a list of merchant category codes.

   b. Click **To** and navigate to the location where you want to save the packaged analysis app.
   c. In the **Save As** dialog box, enter a **File name** with the `.aclapp` file extension and click **Save**.
   d. Click **OK**.

Result – the packaged analysis app is saved to the location you specified. You can now import the file to AX Server.

Package an analysis app with existing interpretations

1. Ensure that the following files are in the same folder on your computer, and that they have the same name:
   - the Analytics project file (`.acl`)
   - the analysis app file (`.aclx`) containing the existing interpretations that you want to import

2. In Analytics, right-click the project entry in the **Overview** tab of the **Navigator** and select **Package Analysis App**.

   The Analytics project is the top-level folder in the tree view.

3. In the **Select Tables** dialog box, do the following:
   a. If you want to include one or more of the project tables and associated data files in the analysis app, select the table(s) and the data file(s) to include.

> **Note**
> Generally you should include only static tables and data files that are required by one or more of the analytic scripts in the analysis app, such as a master vendor table, or a list of merchant category codes.

b. Optional. To include the interpretations from the existing analysis app, select **Include Interpretations**.

   Interpretations that are associated with tables or scripts that do not exist in the new package are not included.

c. Click **To** and navigate to the location where you want to save the packaged analysis app.
d. In the **Save As** dialog box, enter a **File name** with the `.aclapp` file extension and click **Save**.
e. Click **OK**.

Result – the packaged analysis app is saved to the location you specified. You can now import the file to AX Server.

Sample analytic scripts (analysis app)

The sample analytic scripts contain one import script (two versions), one preparation script, and one analysis script. The analytic scripts can be run in any of the following environments or client applications:

- Robots
- AX Server :
  - AX Client
  - AX Web Client
- the Analysis App window

Sequence of the analytic scripts

The three analytic scripts are designed to work in conjunction, and need to be run in the following sequence:

Sequence	`ANALYTIC TYPE`	Analytic script name
1	`IMPORT`	`Sample Import analytic Robots_AX` or `Sample Import analytic Web_AA_Window`
2	`PREPARE`	`Sample Preparation analytic`
3	`ANALYSIS`	`Sample Analysis analytic`

Sample import analytic script

Imports data from the sample Excel file `Trans_May.xls` and saves it to the new Analytics table `Trans_May_raw` (the raw data table).

Two versions of this analytic script are provided.

Analytic script name	Use in	Import file requirement
Sample Import analytic Robots_AX	◦ Robots ◦ AX Client	◦ **Robots** -`Trans_May.xls` must be located in the **Input/Output** tab in the same robot as the analytic script ◦ **AX Client** -`Trans_May.xls` must be located in the **Related Files** subfolder in the AX folder where the

Analytic script name	Use in	Import file requirement
		analytic script is located
Sample Import analytic Web_AA_Window	◦ AX Web Client ◦ Analysis App window	

Sample import analytic script for use in Robots or AX Client

```
COMMENT
//ANALYTIC TYPE IMPORT Sample Import analytic Robots_AX
  This analytic script imports data from the sample Excel file Trans_May.xls
and saves it to the new Analytics table "Trans_May_raw" (the raw data table).
//FILE Trans_May.xls
//DATA Trans_May_raw
//RESULT LOG
END

SET SAFETY OFF
IMPORT EXCEL TO Trans_May_raw Trans_May_raw.fil FROM "Trans_May.xls" TABLE
"Trans2_May$" KEEPTITLE FIELD "CARDNUM" C WID 22 AS "" FIELD "CODES" C WID 4
AS "" FIELD "DATE" D WID 10 PIC "YYYY-MM-DD" AS "" FIELD "CUSTNO" C WID 6 AS
"" FIELD "DESCRIPTION" C WID 95 AS "" FIELD "AMOUNT" N WID 9 DEC 2 AS ""
SET SAFETY ON
```

Sample import analytic script for use in AX Web Client or the Analysis App window

```
COMMENT
//ANALYTIC TYPE IMPORT Sample Import analytic Web_AA_Window
  This analytic script imports data from the sample Excel file Trans_May.xls
and saves it to the new Analytics table "Trans_May_raw" (the raw data table).
//PARAM v_input_file F Input File
  Select an input file
//DATA Trans_May_raw
//RESULT LOG
END

SET SAFETY OFF
IMPORT EXCEL TO Trans_May_raw Trans_May_raw.fil FROM "%v_input_file%" TABLE
"Trans2_May$" KEEPTITLE FIELD "CARDNUM" C WID 22 AS "" FIELD "CODES" C WID 4
```

```
AS "" FIELD "DATE" D WID 10 PIC "YYYY-MM-DD" AS "" FIELD "CUSTNO" C WID 6 AS
"" FIELD "DESCRIPTION" C WID 95 AS "" FIELD "AMOUNT" N WID 9 DEC 2 AS ""
SET SAFETY ON
```

Sample preparation analytic script

Prepares the raw data table for analysis and saves it to the new Analytics table `Trans_May_prepared` (the analysis table). The analytic script defines a shorter version of the "Description" field because classifying only supports field lengths up to 64 characters.

```
COMMENT
//ANALYTIC TYPE PREPARE Sample Preparation analytic
  This analytic script prepares the raw data table for analysis and saves it
to the new Analytics table "Trans_May_prepared" (the analysis table). The ana-
lytic script defines a shorter version of the "Description" field because clas-
sifying only supports field lengths up to 64 characters.
//TABLE v_RawTable Table to prepare
  Select the raw data table you want to prepare
//RESULT TABLE Trans_*_prepared
//DATA Trans_*_prepared
//RESULT LOG
END

SET SAFETY OFF
OPEN %v_RawTable%
DEFINE FIELD DESC_SHORT     ASCII      43  64
EXTRACT RECORD TO "Trans_May_prepared"
SET SAFETY ON
```

Sample analysis analytic script

Classifies the analysis table and outputs the results to the new Analytics table `Classified_Trans_May_prepared` (the results table). Users can specify which field to use for classifying the table, and can specify merchant category codes, customer numbers, and date and transaction amount ranges to restrict which records are processed.

```
COMMENT
//ANALYTIC TYPE ANALYSIS Sample Analysis analytic
  This analytic script classifies the analysis table and outputs the results
to the new Analytics table "Classified_Trans_May_prepared" (the results
table). You can specify merchant category codes, customer numbers, and date
```

```
and transaction amount ranges, to restrict which records are processed.
//TABLE v_AnalysisTable Table to classify
  Select the analysis table you want to classify
  //FIELD v_FieldA C Field to classify on
  Select the field you want to classify on
//PARAM v_codes C MULTI SEPARATOR , QUALIFIER ' VALUES |4112 Passenger Rail-
ways|4121 Taxis/Limousines|4131 Bus travel|4215 Courier Services - Air or
Ground|4411 Cruise Lines|4457 Boat Leases and Boat Rentals|4722 Travel Agen-
cies and Tour Operations|4814 Local/long-distance calls|5812 Restaurants|5813
Drinking Places (Alcoholic Beverages)|5814 Fast food restaurants|5921 Package
Stores, Beer, Wine, Liquor|5993 Cigar Stores & Stands|5994 Newsstand|7216 Dry
cleaners| MC code(s) to include
  Specify one or more merchant category codes to include
//PARAM v_cust_no C OPTIONAL MULTI SEPARATOR , QUALIFIER ' Customer Number(s)
to exclude (optional)
  Specify one or more customer numbers to exclude. Press "Enter" after each
number, so that each number is on a separate line. Do not enclose numbers in
quotation marks.
//PARAM v_start_date D VALUES
|05/01/2003|05/02/2003|05/03/2003|05/04/2003|05/05/2003|05/06/2003|05/07/2003|-
05/08/2003|05/09/2003|05/10/2003|05/11/2003|05/12/2003|05/13/2003|05/14/2003|0-
5/15/2003|05/16/2003|05/17/2003|05/18/2003|05/19/2003|05/20/2003|05/21/2003|05-
/22/2003|05/23/2003|05/24/2003|05/25/2003|05/26/2003|05/27/2003|05/28/2003|05/-
29/2003|05/30/2003|05/31/2003|Start Date
  Select a start date
//PARAM v_end_date D End Date
  Enter an end date or pick one from the calendar
//PARAM v_min_amount N Minimum Amount
  Enter a minimum amount
//PARAM v_max_amount N Maximum Amount
  Enter a maximum amount
//RESULT TABLE Classified_*
//RESULT LOG
END

SET SAFETY OFF
OPEN %v_AnalysisTable%
IF NOT ISDEFINED("v_cust_no") v_cust_no = ""
GROUP IF v_cust_no = ""
  CLASSIFY ON %v_FieldA% IF MATCH(CODES, %v_codes%) AND BETWEEN(DATE, v_start_
date, v_end_date) AND BETWEEN(AMOUNT, v_min_amount, v_max_amount) SUBTOTAL
AMOUNT TO "Classified_%v_AnalysisTable%.FIL" OPEN
ELSE
  CLASSIFY ON %v_FieldA% IF MATCH(CODES, %v_codes%) AND NOT MATCH(CUSTNO, %v_
cust_no%) AND BETWEEN(DATE, v_start_date, v_end_date) AND BETWEEN(AMOUNT, v_
min_amount, v_max_amount) SUBTOTAL AMOUNT TO "Classified_%v_Ana-
lysisTable%.FIL" OPEN
```

```
END
SET SAFETY ON
```

Running Python scripts on AX Server

Have an Analytics Exchange administrator upload external Python scripts to the `PYTHONPATH` directory of AX Server and then call the Python scripts from your analytic scripts to leverage the object-oriented features of the Python programming language on the server. To prepare the AX Server environment to run Python scripts, you must first install Python, and then set the `PYTHONPATH` environment variable.

Prerequisites

To run Python scripts on AX Server, you must:

1. Install a supported version of the Python scripting language on your AX Server computer.
2. Set the `PYTHONPATH` environment variable on AX Server.
3. In Analytics, create a project to work with and import into AX Server.

> **Note**
>
> For help completing these prerequisites, contact your Analytics Exchange administrator and see:
>
> - AX Server requirements
> - Configuring Python for use with AX Server

Create a Python script

After you create your Analytics project in Analytics, create a Python script that you can call from an analytic script.

Then, give your Analytics Exchange administrator this script file to upload to the `PYTHONPATH` directory of the machine hosting AX Server before you call the Python script from an analytic script. When the analytic script runs on AX Server, the Python executable looks for the script in the `PYTHONPATH` directory, so it must be present.

Example Python file

The following example Python file contains a trivial script that uses a lambda expression to raise a number to the power of itself. This example is intended to show how Python scripts run on AX Server, not how to analyze data with Python.

Filename: lambda_example.py

```
# myFunc squares value1 and returns the value
myFunc = lambda value1: value1**2
```

Create an Analytics script

In your Analytics project, create a new script to use as the analytic script you run on AX Server. This script does the following:

1. Opens a simple table called `py` with one record.

   You must open a table to execute the `GROUP` command in Analytics, here the `py` table is used only for this purpose.

2. Loops 10 times and in each loop, executes the Python script by passing the incrementing counter as an argument and extracting the output to a results table.

Add the analytic header

Add the appropriate analytic header tags at the beginning of the script so that the Analytics script can run on AX Server after you import your analysis app:

```
COMMENT
//ANALYTIC Python integration test
 verify Python integration on AX Server
//DATA py
//DATA results
//RESULT TABLE results
END
```

Add the script logic

```
SET SAFETY OFF
DELETE ALL OK
CLOSE

OPEN py

GROUP
```

```
  ASSIGN v_max = 11
  ASSIGN v_counter = 1
  LOOP WHILE v_counter < v_max
    EXTRACT PYNUMERIC("lambda_example,myFunc",0,v_counter) AS "Results value"
TO "results.fil"
    v_counter = v_counter + 1
  END
END
CLOSE py
```

The complete analytic script

The complete analytic script that you run on AX Server looks as follows:

```
COMMENT
//ANALYTIC Python integration test
 verify Python integration on AX Server
//DATA py
//DATA results
//RESULT TABLE results
END

SET SAFETY OFF
DELETE ALL OK
CLOSE

OPEN py

GROUP
  ASSIGN v_max = 11
  ASSIGN v_counter = 1
  LOOP WHILE v_counter < v_max
    EXTRACT PYNUMERIC("lambda_example,myFunc",0,v_counter) AS "Results value"
TO "results.fil"
    v_counter = v_counter + 1
  END
END
CLOSE py
```

Import the Analytics project

Once you have authored the analytic script:

1. In AX Client, create a collection and folder to house the Analytics project.
2. To import the project:
   a. Right-click the folder you created and select **Import**.
   b. Navigate to your Analytics project on your local computer, select the `.acl` project file, and then click **Open**.

> **Note**
> Make sure you import the source data files to import the `py` table with your Analytics project.

Server explorer after import

- *collectionName*
  - *folderName*
    - **Analysis Apps**
      - *ACLProjectName*
        - *analyticScriptName*
    - **Data**
      - py
    - **Related Files**

Run the analytic script

From the **Server Explorer** of AX Client, right-click the analytic script and select **Run**. The Python script is executed as part of the analytic script and you can access the `results` results table from AX Web Client.

> **Note**
> When the script runs, the Python executable looks for the script file in the `PYTHONPATH` directory of the machine hosting AX Server. If your Analytics Exchange administrator has not uploaded the file to this directory, the analytic script fails.

Results

Server explorer after running the analytic script

- *collectionName*
  - *folderName*
    - **Analysis Apps**
      - *ACLProjectName*
        - *analyticScriptName*
    - **Data**
      - py
      - results
    - **Related Files**

Results table

- **Results value**
- 1
- 4
- 9
- 16
- 25
- 36
- 49
- 64
- 81
- 100

Running R scripts on AX Server

Import external R scripts as related files along with an analysis app and then call the R scripts from your analytic scripts to leverage the statistical analysis capabilities of the R scripting language on the server. To prepare the AX Server environment to run R scripts, you must first install R and then add the `.r` extension to the file extension allowlist.

Prerequisites

To run R scripts on AX Server, you must:

1. Install a supported version of the R scripting language on your AX Server computer.
2. Add the `.r` extension to the file extension allowlist on AX Server.
3. In Analytics, create a project to work with and import into AX Server.

> **Note**
>
> For help completing these prerequisites, contact your Analytics Exchange administrator and see:
>
> - AX Server requirements
> - Allowlisting file extensions

Add R scripts to the Analytics project directory

After you create your Analytics project in Analytics, copy the R scripts you intend to use and paste them into the project folder so that you can test your script locally in Analytics before importing it to Analytics Exchange.

Example R files

The following example R files contain trivial scripts that concatenate two strings and return a single string joined by a space character. These examples are intended to show how R scripts run on AX Server, not how to analyze data with R.

analysis_a.r

```
conc<-function(x, y) {
  paste(x, y, sep=" ")
}
print(conc(value1, value2))
```

analysis_b.r

```
conc<-function(x, y) {
  paste(y, x, sep=" ")
}
print(conc(value1, value2))
```

Create an Analytics script

In your Analytics project, create a new script to use as the analytic script you run on AX Server. This script does the following:

1. Opens a temporary table called `t_tmp` with one record.

   You must open a table to execute the `EXTRACT` command in Analytics, here the `t_tmp` table is used only for this purpose.
2. Uses the `EXTRACT` command to run each R script and writes the results to a table.

Add the analytic header

Add the appropriate analytic header tags at the beginning of the script so that the Analytics script can run on AX Server after you import your analysis app. You must add a `FILE` tag for any R script you intend to run from the analytic script:

```
COMMENT
//ANALYTIC R integration test
  verify R integration on AX Server
//DATA t_tmp
//FILE analysis_a.r
//FILE analysis_b.r
//RESULT TABLE results
END
```

Add the script logic

```
SET SAFETY OFF
DELETE ALL OK
CLOSE PRIMARY SECONDARY

OPEN t_tmp

COM **** execute R scripts and write results to table
EXTRACT FIELDS RSTRING("a<-source('./analysis_a.r');a[[1]]",50,"test","value")
AS "value" TO "results.fil"
EXTRACT FIELDS RSTRING("a<-source('./analysis_b.r');a[[1]]",50,"test","value")
AS "value" TO "results.fil" APPEND

CLOSE t_tmp
```

The complete analytic script

The complete analytic script that you run on AX Server looks as follows:

```
COMMENT
//ANALYTIC R integration test
 verify R integration on AX Server
//DATA t_tmp
//FILE analysis_a.r
//FILE analysis_b.r
//RESULT TABLE results
END

SET SAFETY OFF
DELETE ALL OK
CLOSE PRIMARY SECONDARY

OPEN t_tmp

COM **** execute R scripts and write results to table
EXTRACT FIELDS RSTRING("a<-source('./analysis_a.r');a[[1]]",50,"test","value")
AS "value" TO "results.fil"
EXTRACT FIELDS RSTRING("a<-source('./analysis_b.r');a[[1]]",50,"test","value")
AS "value" TO "results.fil" APPEND

CLOSE t_tmp
```

Import the Analytics project and related R files

Once you have authored the analytic script:

1. In AX Client, create a collection and folder to house the Analytics project.
2. To import the project and R files:
   a. Right-click the folder you created and select **Import**.
   b. Navigate to your Analytics project on your local computer and select the `.acl` project file and the `.r` R scripts.

> **Note**
> Make sure you select the R files in the project folder as well as the Analytics project using **Ctrl+click** so that they are imported into AX Server. You must also import the source data files for the `t_tmp` table.

   c. Click **Open**.

Server explorer after import

- *collectionName*
  - *folderName*
    - **Analysis Apps**
      - *ACLProjectName*
        - *analyticScriptName*
    - **Data**
      - t_tmp
    - **Related Files**
      - analysis_a.r
      - analysis_b.r

Run the analytic script

From the **Server Explorer** of AX Client, right-click the analytic script and select **Run**. The R scripts are executed as part of the analytic script and you can access the `results` results table from AX Web Client.

Results

Server explorer after running the analytic script

- *collectionName*
  - *folderName*
    - **Analysis Apps**
      - *ACLProjectName*
        - *analyticScriptName*
    - **Data**
      - results
    - **Related Files**
      - analysis_a.r
      - analysis_b.r

Results table

- **value**
- test value
- value test

Analytic headers and tags

An analytic header is a series of tags enclosed in a comment block at the start of an analytic script. The analytic tags specify input parameters that a user populates in advance of scheduling or running an analytic script, and output parameters.

An analytic header is required for any analytic script that you intend to run in Robots, on AX Server, or in the Analysis App window.

For a visual overview of analytic script inputs and outputs, and associated analytic tags, see "Script inputs and outputs – the big picture" on page 2464.

Adding or modifying an analytic header

The easiest way to add or modify an analytic header is to use the Analytic Header Designer. The Designer provides automated error checking and embedded guidance to help ensure that the header you build is valid and works correctly. For more information, see "Working with analytic headers" on page 2473.

You can also build an analytic header by manually entering the required analytic tags. Or you can use a combination of manual entry and the Designer.

Basic analytic header requirements

Analytic headers must be defined in a comment block that starts on the first line of the script. Tags can be in any order in the analytic header, except for:

- the `ANALYTIC` tag, which must be first
- `FIELD` tags, which must immediately follow their associated `TABLE` tag

Example

This analytic header identifies a table and field to use in the script, as well as a start date parameter:

```
COMMENT
//ANALYTIC Identify missing checks
  This analytic script identifies missing check numbers
//TABLE v_table_payments Payments Table
  Select a table that lists payments and includes a check number column
```

```
//FIELD v_check_num CN Check Number
  Select the field that contains the check numbers
//PARAM v_start_date D OPTIONAL Start Date (Optional)
  Enter the start date for the analysis
END
```

Tag format

Each tag in the header uses the following format:

```
//tag_name attributes
  optional_descriptive_text
```

The `//` tag indicator must be the first non-whitespace character on the script line. The tag name must immediately follow the tag indicator, without any space or characters in between.

The optional descriptive text must be entered on the next line after the tag. The text can be multiline, but it cannot skip lines. Line breaks are not preserved when the descriptive text is displayed in client applications.

Tag conventions

Component	Convention
Tag names	Tag names are not case-sensitive. Unlike Analytics command and function names, tag names cannot be abbreviated.
Tag attributes	When specifying attribute values for a tag, you may include spaces and optionally enclose the value in quotation marks.
Tag descriptions	Descriptions are optional. If a description is specified it can be multi-line, but line breaks are not preserved in client applications.

Specifying test input values in Analytics

You can use a special assignment operator (`:=`) to specify test input values for any analytic tag that requires definition:

- `FILE`
- `PARAM`

- `TABLE`
- `FIELD`

Use this syntax to test analytic scripts in Analytics:

```
//TABLE v_AnalysisTable "Table to classify" := "Trans_May"
```

When the script runs in Analytics, the parameter takes the value specified in the assignment. When the analytic script runs in a client application, the test value is ignored and the user-defined input parameters are used.

You must leave a space between the assignment operator and the tag syntax preceding it. Assignment values must use the correct qualifier for the data type as required throughout Analytics. For more information, see "Data types" on page 1404.

Full list of available analytic tags

Tag	Description
"ANALYTIC tag" on page 2509	Designates a script as an analytic script that can run in Robots, on AX Server, or in the Analysis App window.
Input tags	
"FILE tag" on page 2514	Specifies a non-Analytics file, such as an Excel file, or a delimited file, that provides input for an analytic script running in Robots, or on AX Server. ◦ **Robots** - the file must be located in the **Input/Output** tab in the same robot as the analytic script ◦ **AX Server** - the file must be located in the **Related Files** subfolder in the folder where the analytic script is located
"PARAM tag" on page 2517	Creates an input parameter for an analytic script, and defines the requirements for the input value. An input parameter is a placeholder that allows the user to specify the actual value when scheduling or running an analytic script.
"PASSWORD tag" on page 2530	Creates a password input parameter for an analytic script. The parameter provides encrypted storage of a password for subsequent use in an ACLScript command. The user is prompted to specify the required password value when they schedule or start the analytic script so that user intervention is not required as the analytic script is running.
"TABLE tag" on page 2533	Defines an Analytics table that the user selects as input for an analytic script. The `TABLE` tag can be followed by zero or more `FIELD` tags entered on sequential lines.
"FIELD tag" on page 2535	Defines a field that the user selects as input for an analytic script. The field must be part of the table defined in the preceding `TABLE` tag. The first `FIELD` tag must immediately follow a `TABLE` tag, and can be followed by additional `FIELD` tags

Tag	Description
	entered on sequential lines.
Output tags	
"RESULT tag" on page 2538	Specifies that the results output by an analytic script are available in client applications to end users. Output results, even if they exist, are not automatically made available.
"DATA tag" on page 2545	Specifies that an Analytics table output by an analytic script is copied to a data subfolder (a storage location) in the deployment environment. Typically, you store Analytics tables so that they can be used as input tables for subsequent analytic scripts.
"PUBLISH tag" on page 2550	Specifies a file that contains metadata defining which Analytics tables to publish to AX Exception when an analytic script is finished processing.

ANALYTIC tag

Designates a script as an analytic script that can run in Robots, on AX Server, or in the Analysis App window.

Syntax

```
//ANALYTIC <TYPE IMPORT|PREPARE|ANALYSIS> name
  <description>
```

> **Note**
> An ACLScript `COMMENT` command must be entered on the first line in an analytic script, followed by the `ANALYTIC` tag on the second line. If the `ANALYTIC` tag is used in any other location it is ignored.
>
> One or more scripts in an Analytics project can include an `ANALYTIC` tag.

Parameters

Name	Description
TYPE optional	Identifies an analytic script as one of the following three types: ◦ `IMPORT` - retrieves data from a data source. The output of an import analytic script is a raw data table. ◦ `PREPARE` - transforms raw data in whatever way is necessary to make it suitable for analysis. The output of a preparation analytic script is an analysis table. ◦ `ANALYSIS` - performs tests on data in analysis tables. The output of an analysis analytic script is one or more results tables. The specified type does not create any restrictions on the actual content of an analytic script. As the script writer, you control how the specified type and the script content align. Analytic scripts with a specified type are grouped in corresponding areas in Robots, AX Web Client, or the Analysis App window. The areas are ordered as follows: ◦ **Import** ◦ **Preparation** ◦ **Analysis** If you omit `TYPE`, the analytic script appears in the **Analysis** area. For more information, see "Using analytic script type and name to sequentially order a series of scripts" on page 2512.
name	The name of the analytic script.

Name	Description
	The name identifies the analytic script in client applications. The analytic script name is not the same as the script name that you specify in Analytics when you initially create the script. **Note** Analytic script names in the same project or analysis app must be unique. If *name* has an identical value in two or more analytic scripts, an error occurs when you try to commit the analytic scripts, or import or open the analysis app. ## Supported characters in the analytic script name To ensure that an analytic script name does not cause a processing problem, a best practice is to use only these characters in *name*: `A-Z`, `a-z`, `0-9`, underscore (`_`), or dash (`-`) ## Unsupported characters in the analytic script name Do not use the following characters in *name*. They are not supported: `< > : " / \ | ? *` Do not use the value `TYPE` as the name. ## Using a sequential prefix with the analytic script name In client applications, analytic script names are listed in alphanumeric order. To guide users in the correct sequence for running multiple analytic scripts, you can add a prefix to order the analytic script names. For example: `A_01_analyze_POs`, `A_02_analyze_invoices`, and so on. For more information, see "Using analytic script type and name to sequentially order a series of scripts" on page 2512.
description optional	A description of the analytic script or other information that the user might need to run the analytic script successfully. The description appears with the analytic script in client applications. The description can be multiline, but it cannot skip lines. The description must be entered on the line below the associated `ANALYTIC` tag.

Examples

Basic analytic header

The following analytic header contains a name and a description of the analytic script:

```
COMMENT
//ANALYTIC Identify missing checks
  This analytic script identifies missing check numbers.
END
```

Analytic header with type

The following analytic header specifies a preparation analytic script with a description of what the script does:

```
COMMENT
//ANALYTIC TYPE PREPARE Standardize address data
  This analytic script cleans and standardizes the address field in pre-
paration for duplicates analysis.
END
```

Analytic header with additional analytic tags

The following analytic header contains additional analytic tags:

```
COMMENT
//ANALYTIC TYPE IMPORT Import transactional data
  This analytic script imports data from the Excel file Trans_May.xls
and saves it to the new Analytics table "Trans_May_raw" (the raw data
table).
//FILE Trans_May.xls
//DATA Trans_May_raw
```

```
//RESULT LOG
END
```

Remarks

Using analytic script type and name to sequentially order a series of scripts

You can use the `TYPE` and *name* parameters to sequentially order a series of analytic scripts in a robot task or in an analysis app. The resulting behavior depends on the client in which the scripts are run.

> **Note**
>
> The `TYPE` and *name* parameters affect the ordering of analytic scripts only. They do not affect auxiliary or helper scripts called using the `DO SCRIPT` command.

Client	TYPE parameter	*name* parameter	Analytic script execution sequence
Robots	Groups analytic scripts into separate areas in a task, in the following order: ◦ **Import** ◦ **Preparation** ◦ **Analysis**	Orders analytic scripts alphanumerically by name in a task, or in an area in a task	Execution sequence enforced. The sequence follows the order imposed by *name*, or by `TYPE` and *name*.
AX Client	No effect. Analytic script areas not supported.	Orders analytic scripts alphanumerically by name in an analysis app	Execution sequence not enforced. The user runs or schedules analytic scripts individually, guided by the order established by *name*.

Client	TYPE parameter	*name* parameter	Analytic script execution sequence
			Note AX Client supports separate functionality for creating analytic chains to enforce sequential execution of analytic scripts.
AX Web Client	Groups analytic scripts into separate areas in an analysis app, in the following order: ◦ **Import** ◦ **Preparation** ◦ **Analysis**	Orders analytic scripts alphanumerically by name in an analysis app, or in an area in an analysis app	Execution sequence not enforced. The user runs analytic scripts individually, guided by the order established by *name*, or by `TYPE` and *name*.
Analysis App window	Groups analytic scripts into separate areas in an analysis app, in the following order: ◦ **Import** ◦ **Preparation** ◦ **Analysis**	Orders analytic scripts alphanumerically by name in an analysis app, or in an area in an analysis app	Execution sequence not enforced. The user runs analytic scripts individually, guided by the order established by *name*, or by `TYPE` and *name*.

FILE tag

Specifies a non-Analytics file, such as an Excel file, or a delimited file, that provides input for an analytic script running in Robots, or on AX Server.

- **Robots** – the file must be located in the **Input/Output** tab in the same robot as the analytic script
- **AX Server** – the file must be located in the **Related Files** subfolder in the folder where the analytic script is located

> **Note**
> To specify a non-Analytics input file for an analytic script run in the Analysis App window, see "PARAM tag" on page 2517.

Syntax

```
//FILE filename
  <description>
```

Parameters

Name	Description
filename	The name of the file in the **Input/Output** tab (Robots), or in the **Related Files** subfolder (AX Server), to use as input for an analytic script. **Note** The *filename* value must exactly match the name of the file in the **Input/Output** tab, or in the **Related Files** subfolder. *filename* cannot include a path. You can use wildcard characters in *filename* to assist with matching a name. You cannot use a variable for *filename*. ### Unsupported characters in the file name Do not include any spaces in *filename*. Do not use any of the following characters in *filename*. They are not supported: `< > : " \ / |`

Name	Description
	## Wildcard characters Wildcards are supported when specifying the file name. Use a single asterisk (`*`) to substitute for zero or more consecutive characters. For example: ◦ `Inv12*` matches all of the following: `Inv12`, `Inv123`, and `Inv1234` ◦ `*.*` matches all files of all extensions ◦ `Inv_*.*` matches `Inv_Jan.pdf` and `Inv_Feb.xls` ◦ `*.xlsx` matches all Excel files with a `.xlsx` extension ## Analytics preferences file (AX Server) You can use the `FILE` tag to reference a `.prf` Analytics preferences file. When you do this, the preferences file in the **Related Files** subfolder is used to set the runtime environment settings rather than the global preferences file on AX Server. The preferences file must be from the latest version of Analytics that is compatible with your Analytics Exchange installation.
description optional	Descriptive text about the non-Analytics file or other information. The description can be multiline, but it cannot skip lines. The description appears in the analytic header only and is not visible to end users in client applications.

Examples

Basic examples

Specifies a specific file:

```
//FILE FlaggedAccounts.csv
```

Specifies all CSV files starting with "Flagged":

```
//FILE Flagged*.csv
```

Specifies all files:

```
//FILE *.*
```

Advanced examples

Import data from a related file

You run a monthly analysis of employee data contained in a delimited file that is manually uploaded to the **Related Files** subfolder on AX Server on a monthly basis. One of the analytic scripts in the analysis app imports the data from the delimited file to an Analytics table:

```
COMMENT
//ANALYTIC TYPE IMPORT employee_import
  Imports employee records from delimited file stored in Related Files
folder.
//FILE Employees.csv
END
IMPORT DELIMITED TO Employees "Employees.fil" FROM "Employees.csv" 0
SEPARATOR "," QUALIFIER '"' CONSECUTIVE STARTLINE 1 KEEPTITLE FIELD
"First_Name" C AT 1 DEC 0 WID 11 PIC "" AS "First Name"  FIELD "Last_
Name" C AT 12 DEC 0 WID 12 PIC "" AS "Last Name"
```

Remarks

To be used in a script, a non-Analytics file must first be imported into an Analytics table. Non-Analytics files cannot be used directly in a script.

The `FILE` tag is not supported for use in analytic scripts run in the Analysis App window. To specify an input file for analytic scripts run in the Analysis App window, use the `PARAM` tag. For more information, see "PARAM tag" on the facing page.

PARAM tag

Creates an input parameter for an analytic script, and defines the requirements for the input value.

An input parameter is a placeholder that allows the user to specify the actual value when scheduling or running an analytic script.

Syntax

```
//PARAM variable_name type <OPTIONAL> <MULTI> <SEPARATOR value>
<QUALIFIER value> <VALUES value_list> label
  <description>
```

Parameters

Name	Description
variable_name	The name of the variable that stores the input value(s) selected or specified by the user. Use *variable_name* in the analytic script to reference the input value. For example: ◦ `v_start_date` ◦ `v_regions` ◦ `v_input_file` Also serves as the unique identifier for the parameter. **Note** When an analytic script is run, the variable is created only if the user provides an input value. If a parameter is optional, and the user skips it, the variable is not created. If subsequent logic in the analytic script requires the variable to exist, you can test for its existence, and if it does not exist, create it and initialize it. For more information, see "Designing optional input parameters" on page 2525. ### Unsupported characters in the parameter variable name Do not include any spaces in *variable_name*. Do not use any of the following characters in *variable_name*. They are not supported:

Name	Description
	`` ` ~ ! @ # $ % ^ & * ( ) - + = { } [ ] | \ : ; ' " < > , . / ? ``
type	The data type of the parameter, which controls what sort of input values can be entered. The following types can be specified using uppercase letters: ◦ `C` - character data ◦ `N` - numeric data ◦ `D` - date subtype of datetime data ◦ `DT` - datetime subtype of datetime data ◦ `T` - time subtype of datetime data ◦ `L` - logical data **Note** Qualifying character input values is required for an analytic script to run successfully. **How `PARAM... F` works** You can also specify that a file upload utility, or a Windows file browser, opens: ◦ `F` - opens a file upload utility, or a Windows file browser, and allows a user to select a non-Analytics input file for the analytic script when running in AX Web Client or the Analysis App window Upon selection, the file name is automatically entered as a Character input value. Specify `F` only. Do not specify `F C`. For example: `//PARAM v_input_file F...` For more information, see "Specifying or selecting a non-Analytics input file for an analytic script" on page 2528. **Note** A *type* of `F` is not supported for use in analytic scripts run in Robots or AX Client. To specify an input file for these environments, use the `FILE` tag. For more information, see "FILE tag" on page 2514.
OPTIONAL optional	Specifies that the parameter is optional and the user does not need to enter a value. For more information, see "Designing optional input parameters" on page 2525.
MULTI optional	Specifies that the parameter accepts multiple input values. **Note** `MULTI` cannot be used if *type* is `L` (Logical) or `F` (File). **MULTI and VALUES**

Name	Description
	`MULTI` can be used with or without the `VALUES` option: `MULTI` ✔ `VALUES` ✔ — The user can select one or more values from a list of values. `MULTI` ✔ `VALUES` ✘ — The user can manually enter one or more values. For more information, see "Summary of the MULTI and VALUES options" on page 2525. **Multiple character input values** If you specify `MULTI`, and *type* is `C` (Character), you can also specify the `SEPARATOR` and `QUALIFIER` options to automatically insert separators (delimiters) and text qualifiers in a string of input values. **Note** Delimiting and qualifying multiple character input values is required for an analytic script to run successfully. The separators and qualifiers can be inserted automatically, or manually by the user.
SEPARATOR *value* optional	`SEPARATOR` can be used only when `MULTI` is specified, and *type* is `C` (Character). Specifies that a separator character is automatically inserted between multiple character input values, creating a delimited list that is passed to the analytic script for processing. *value* specifies the separator character to use. A commonly used separator, or delimiter, is the comma `,`. If `SEPARATOR` is omitted, a single space is used as the separator by default. The space character cannot be specified as *value*. For more information, see "Delimiting and qualifying character input values" on page 2526.
QUALIFIER *value* optional	`QUALIFIER` can be used only when `MULTI` is specified, and *type* is `C` (Character). Specifies that a text qualifier character is automatically inserted at the start and end of each character input value in a delimited list that is passed to the analytic script for processing. Any text enclosed within the qualifier characters is treated as plain text. *value* specifies the qualifier character to use. A commonly used qualifier is the single quotation mark `'`. If `QUALIFIER` is omitted, there is no default qualifier used. You cannot specify a space character as *value*. For more information, see "Delimiting and qualifying character input values" on page 2526.

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>

> **Note**
> Analytic input parameters currently do not support the use of the double quotation mark (") as a text qualifier. You can use the single quotation mark (') instead. Specifying a double quotation mark qualifier will cause the `PARAM` tag to malfunction.

</td></tr>
<tr><td>VALUES value_list
optional</td><td>

A list of values that the user can select from when running the analytic script.

Use the following syntax to specify the values:

```
VALUES |Value 1|Value 2|Value 3|Value n|
```

VALUES and MULTI

`VALUES` can be used with or without the `MULTI` option:

<table>
<tr><td>`VALUES` ✔
`MULTI` ✔</td><td>The user can select one or more values from the list of values.</td></tr>
<tr><td>`VALUES` ✔
`MULTI` ✘</td><td>The user can select a single value from the list of values.</td></tr>
</table>

For more information, see "Summary of the MULTI and VALUES options" on page 2525.

Format of values in *value_list*

<table>
<tr><td>Character values</td><td>◦ can contain spaces and punctuation</td></tr>
<tr><td>Numeric values</td><td>◦ can be positive or negative
◦ must be specified using decimal notation, and without a thousands separator
For example, `1500.00` or `-1500.00`</td></tr>
<tr><td>Datetime values</td><td>◦ Date - must be specified using the format MM/DD/YYYY
For example, `12/31/2014`
◦ Datetime - must be specified using the format MM/DD/YYYY hh:mm:ss
For example, `12/31/2014 23:59:59`
◦ Time - must be specified using the format hh:mm:ss
For example, `23:59:59`</td></tr>
<tr><td>Logical values</td><td>`VALUES` cannot be used if type is `L` (Logical)</td></tr>
</table>

</td></tr>
<tr><td>label</td><td>The user interface label for the parameter.
In client applications, label is displayed with the input field.</td></tr>
</table>

Name	Description
description optional	Descriptive text that provides additional information about the parameter. In client applications, *description* is displayed with the input field. *description* can provide instructions that assist the user. For example, "Enter the cutoff date for the payroll period". *description* must be entered on the next line after the associated `PARAM` tag. The text can be multiline, but it cannot skip lines. Line breaks are not preserved when displayed in client applications.

Examples

Basic examples

Allows the user to optionally specify a date range:

```
//PARAM v_start_date D OPTIONAL Start Date (Optional)
  Enter the start date for the analysis
//PARAM v_end_date D OPTIONAL End Date (Optional)
  Enter the end date for the analysis
```

Requires the user to select a maximum number of transactions to process:

```
//PARAM v_maxTrans N VALUES |250|500|750|1000| Maximum transactions to process
```

Requires the user to specify one or more merchant category codes:

```
//PARAM v_codes C MULTI SEPARATOR , QUALIFIER ' MC Codes to include
  Specify one or more merchant category codes. Press "Enter" after each code,
so that each code is on a separate line. Do not enclose codes in quotation
marks.
```

Requires the user to select one or more merchant category codes:

```
//PARAM v_codes C MULTI SEPARATOR , QUALIFIER ' VALUES |4121 Tax-
is/Limousines|5812 Restaurants|5813 Drinking Places - Alcoholic Beverages|5814
Fast food restaurants| MC Codes to include
  Select one or more merchant category codes.
```

Advanced examples

Require a user to specify an amount range

You need to classify the records in a table that fall between a minimum and maximum amount range. This range changes occasionally, so you provide input parameters that allow the user who runs the analytic script to define the range when scheduling or running the script:

```
COMMENT
//ANALYTIC test_analytic
//PARAM v_min_amount N Minimum Amount
  Enter a minimum amount
//PARAM v_max_amount N Maximum Amount
  Enter a maximum amount
END

CLASSIFY ON %v_FieldA% IF BETWEEN(AMOUNT, v_min_amount, v_max_amount)
SUBTOTAL AMOUNT TO "Classified_%v_AnalysisTable%.FIL"
```

Allow the user to optionally exclude one or more customer numbers

You need to classify the records in a table but you want to give the user the option to exclude some customers from the analysis.

To do this, you provide an optional character parameter. Your script tests whether or not the value is provided, and if so, those customer numbers are excluded from the classify command:

```
COMMENT
//ANALYTIC test_analytic
//PARAM v_cust_no C OPTIONAL MULTI SEPARATOR , QUALIFIER ' Customer Num-
ber(s) to exclude (optional)
  Specify one or more customer numbers. Press "Enter" after each number,
so that each number is on a separate line. Do not enclose numbers in quo-
tation marks.
END

IF FTYPE("v_cust_no") = "U" v_cust_no = ""
```

```
GROUP IF v_cust_no = ""
  CLASSIFY ON %v_FieldA% SUBTOTAL AMOUNT TO "Classified_%v_Ana-
lysisTable%.FIL"
ELSE
  CLASSIFY ON %v_FieldA% IF NOT MATCH(CUSTNO, %v_cust_no%) SUBTOTAL
AMOUNT TO "Classified_%v_AnalysisTable%.FIL"
END
```

Allow the user to select an input file (AX Web Client or the Analysis App window only)

You are distributing an analysis app to colleagues who will run it in the Analysis App window. When they run the analytic script in the app, you want to provide them with a Windows file browser to select a Microsoft Excel file to import data from:

```
COMMENT
//ANALYTIC test_analytic
//PARAM v_input_file F Input File
  Select an input file
END

IMPORT EXCEL TO Trans_May_raw Trans_May_raw.fil FROM "%v_input_file%"
TABLE "Trans2_May$" CHARMAX 100 KEEPTITLE
```

Require the user to specify an input file path and file name (the Analysis App window only)

You are distributing an analysis app to colleagues who will run it in the Analysis App window. When they run the analytic script in the app, you want them to specify a filepath and filename to use as an import file:

```
COMMENT
//ANALYTIC test_analytic
//PARAM v_input_file C Input File Path and Name
  Enter an absolute file path and a file name, for example: C:\User-
s\username\Documents\ACL Data\Sample Data Files\ Trans_May.xls
```

```
END

IMPORT EXCEL TO Trans_May_raw Trans_May_raw.fil FROM "%v_input_file%"
TABLE "Trans2_May$" CHARMAX 100 KEEPTITLE
```

Using default values for optional parameters

You are creating an analytic script that extracts transaction records to a results table. You want to give the user who runs the script the option of providing a date range as well as a list of entities for filtering the records to extract.

To do this, you create three optional parameters:

- v_start_date
- v_end_date
- v_entity_list

In the opening lines of the script, you test if these values are set. If they are not set, you set default values of the minimum and maximum dates as well as a default flag to test for with v_entity_list.

In your EXTRACT command, you use the values to filter the records:

```
COMMENT
//ANALYTIC test
  This analytic script tests the PARAM
//RESULT TABLE t_results
//PARAM v_start_date D OPTIONAL Enter Start Date
//PARAM v_end_date D OPTIONAL Enter End Date
//PARAM v_entity_list C MULTI OPTIONAL |entity1|entity2|
END

IF NOT ISDEFINED("v_start_date") v_start_date = `19000101`
IF NOT ISDEFINED("v_end_date") v_end_date = `99991231`
IF NOT ISDEFINED("v_entity_list") v_entity_list = "'all'"

EXTRACT FIELDS ALL TO t_results IF BETWEEN(transaction_date v_start_date
v_end_date) AND (MATCH(entity_field %v_entity_list%) OR v_entity_list =
"'all'")
```

Remarks

Designing optional input parameters

If you use `OPTIONAL` with the `PARAM` tag, the variable associated with the input parameter may or may not be created when the analytic script runs:

- **variable automatically created** – if the user specifies an input value
- **variable not created** – if the user skips the optional parameter and does not specify an input value

Test for the existence of the parameter variable

If subsequent logic in the analytic script depends on being able to evaluate the contents of the parameter variable, including evaluating an empty or null state, you need to test for the existence of the parameter variable. If the parameter variable does not exist, you need to create it and initialize it to null.

Use the IF command with the FTYPE( ) function, or the ISDEFINED( ) function, to perform the test and create the variable if it does not exist:

```
IF FTYPE("var_name") = "U" var_name = ""
```

```
IF NOT ISDEFINED("var_name") var_name = ""
```

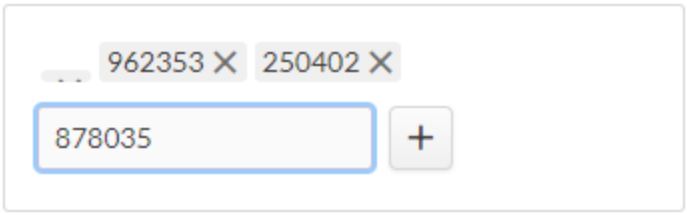
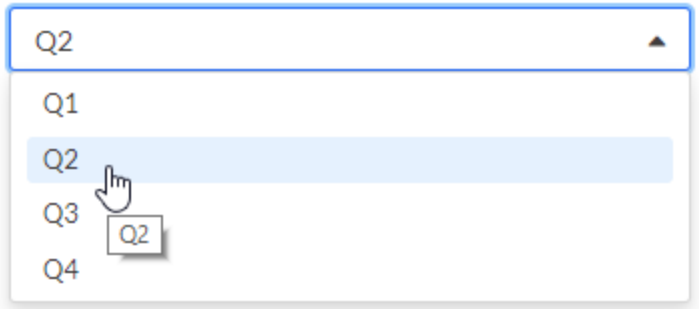
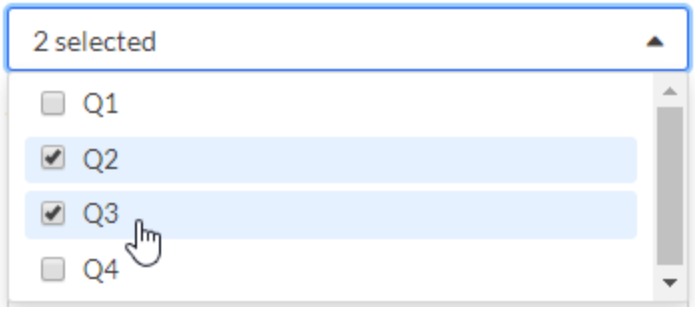
When to perform the test

Perform the test after the analytic header and before any Analyticsscript logic that depends on the existence of the parameter variable.

Summary of the MULTI and VALUES options

The table below summarizes the effect of the MULTI and VALUES options on the user input control in the user interface.

User input control (Robots)	Parameter design	`MULTI`	`VALUES`
	A single input value manually entered in a field	✕	✕

User input control (Robots)	Parameter design	MULTI	VALUES
	One or more input values manually entered in a field	✔	✘
	A single input value selected from a drop-down list of values	✘	✔
	One or more input values selected from a checklist of values	✔	✔

Delimiting and qualifying character input values

For an analytic script to run successfully, character input values must be delimited by a separator if there is more than one value, and values must be qualified.

Avoid nested text qualifiers

When you create character input parameters, and when you instruct users of the analytic script how to enter character input values, you need to be careful to avoid creating redundant or nested text qualifiers (qualifiers within qualifiers). Redundant text qualifiers will cause the input parameter to malfunction.

Methods for inserting text qualifiers

There are four different methods available for inserting text qualifiers around character input values. Depending on the method, a separator is also inserted between the input values.

As you develop an analytic script, you may need to experiment with different methods to find what works best for the character values that users will input.

> **Note**
>
> One or more of the methods may not be applicable, depending on how you are using the MULTI and VALUES options.
>
> Each input parameter must use **only one** of these methods.

1	Use SEPARATOR and QUALIFIER	Include the SEPARATOR and QUALIFIER options in the PARAM tag. For example: `//PARAM v_regions C MULTI SEPARATOR , QUALIFIER '` Not applicable if you use VALUES without MULTI. **Tip** Use this method whenever possible. It is the least labor-intensive and the least error-prone.
2	Manually specify separators and qualifiers	Require the user of the analytic script to manually specify separators and qualifiers in addition to the actual input values. For example: `'North America','Europe','Asia'` Not applicable if you use VALUES with or without MULTI.
3	Include qualifiers in the value_list	Include qualifiers with each value in the *value_list* specified with the VALUES option. For example: `VALUES |'Asia'|'Europe'|'Middle East'|'North America'|` Not applicable if you use MULTI without VALUES.
4	Enclose the parameter variable in qualifiers	In the syntax of the Analytics script, enclose the parameter variable in text qualifiers. For example:

		`IF MATCH(REGIONS, "%v_regions%")` Use this method only if you are using VALUES without MULTI.
	Note Analytic input parameters currently do not support the use of the double quotation mark (") as a text qualifier. You can use the single quotation mark (') instead with the QUALIFIER option, in the *value_list*, or when manually specifying qualifiers around input values. Double quotation marks can be used as text qualifiers in the body of an Analytics script.	

When to use the different methods

The table below summarizes when to use the different methods for inserting text qualifiers.

	`MULTI` ✔ `VALUES` ✘	`MULTI` ✘ `VALUES` ✔	`MULTI` ✔ `VALUES` ✔
Method 1 Use the SEPARATOR and QUALIFIER options	If used, do not use Method 2	Not applicable	If used, do not use Method 3
Method 2 Manually specify separators and qualifiers	If used, do not use Method 1	Not applicable	Not applicable
Method 3 Include qualifiers in the *value_list*	Not applicable	If used, do not use Method 4	If used, do not use Method 1
Method 4 Enclose the parameter variable in qualifiers	Do not use	If used, do not use Method 3	Do not use

Specifying or selecting a non-Analytics input file for an analytic script

The table below summarizes the different methods for specifying or selecting a non-Analytics input file for an analytic script. The method you choose is partly dependent on which client application will be used to run the analytic script.

Method	Details	Robots	AX Client	AX Web Client	The Analysis App window
PARAM tag with type of 'F'	◦ **AX Web Client** - user selects the input file using a file upload utility The file name is automatically specified as the input value. The file is automatically uploaded to the appropriate **Related Files** subfolder on AX Server. ◦ **Analysis App window** - user selects the input file using the Windows file browser The file path and the file name are automatically specified as the input value. This method is the best option because it combines flexibility, ease of use, and precision.			✔	✔
PARAM tag with type of 'C'	The user manually specifies an input file path and file name as an input value. This method provides flexibility because the file path and the file name are not specified in advance. However, it is laborious and error prone because it requires the user to manually enter these values.				✔
FILE tag (For more information, see "FILE tag" on page 2514)	◦ **Robots** - the input file must be located in the **Input/Output** tab in the robot ◦ **AX Client, AX Web Client** -the input file must be located in the appropriate **Related Files** subfolder on AX Server	✔	✔	✔	
Input file path and file name hard-coded in the analytic script	This method avoids use of the PARAM tag, however it is the least flexible. On every computer where the analytic script is run, the user must ensure that the input file has a file path and a file name identical to those specified in the analytic script.				✔

PASSWORD tag

Creates a password input parameter for an analytic script. The parameter provides encrypted storage of a password for subsequent use in an ACLScript command.

The user is prompted to specify the required password value when they schedule or start the analytic script so that user intervention is not required as the analytic script is running.

Syntax

```
//PASSWORD identifier label
  <description>
```

Parameters

Name	Description
identifier	The numerical identifier associated with the password. The value must be from 1 to 10.
label	In client applications, the interface label that users see when prompted to enter the password. For example, `SAP Password:`
description optional	In client applications, descriptive text about the required password that users see. The description can help the user enter the correct password. The description can be multiline, but it cannot skip lines. The description must be entered on the line below the associated `PASSWORD` tag.

Examples

Create a password input parameter for a Direct Link SAP query

The analytic header specifies a password input parameter that prompts the user to enter an SAP password. The stored password is used in the subsequent RETRIEVE command in the

body of the script.

```
COMMENT
//ANALYTIC SAP Password Example
//PASSWORD 1 SAP Password:
//DATA RSADMIN
END
SET SAFETY OFF
RETRIEVE RSADMIN PASSWORD 1
OPEN RSADMIN
SET SAFETY ON
```

> **Note**
>
> The password input parameter and the password parameter in the RETRIEVE command are linked by using the same numerical identifier:
>
> ```
> //PASSWORD 1 SAP Password:
> .
> .
> .
> RETRIEVE RSADMIN PASSWORD 1
> ```

Create a password input parameter for an export to Results

The analytic header specifies a password input parameter that prompts the user to enter a HighBond password. The stored password is used in the subsequent EXPORT command in the body of the script.

```
COMMENT
//ANALYTIC HighBond Password Example
//PASSWORD 3 HighBond Password:
END
SET SAFETY OFF
OPEN AR_Exceptions
EXPORT FIELDS No Due Date Ref Amount Type ACLGRC PASSWORD 3 TO
"10926@us"
SET SAFETY ON
```

Remarks

Password storage and encryption

Password values are associated with individual users, and are encrypted at rest. Passwords remain secure throughout analytic script processing, and are encrypted in any temporary files created in the deployment environment.

Testing in Analytics

If you test an analytic script that has one or more `PASSWORD` tags in Analytics, Analytics automatically generates a PASSWORD command and prompts you to enter the appropriate password. This auto-generated command saves you the labor of inserting PASSWORD commands in the script portion of an analytic script for the purposes of testing, and then removing them again before delivering the analytic script to users.

The auto-generated PASSWORD command is saved in the log, without the password value.

Password values are not saved when you run an analytic script in Analytics, so you must specify the password or passwords each time you run the analytic script, including running or stepping through the analytic script from the cursor position.

TABLE tag

Defines an Analytics table that the user selects as input for an analytic script.

The `TABLE` tag can be followed by zero or more `FIELD` tags entered on sequential lines.

> **Note**
>
> The `TABLE` and `FIELD` tags require that an Analytics table pre-exists in the storage location in order to be available to be selected. For more information, see "DATA tag" on page 2545.
>
> Use the `TABLE` and `FIELD` tags if you want to create variables that allow users to specify different tables or fields for use with the same analytic script. If the script is designed to always work with the same table and set of fields, with names that do not change, you can hardcode the table and field names into the script and avoid use of the `TABLE` and `FIELD` tags.

Syntax

```
//TABLE variable_name label
  <description>
```

Parameters

Name	Description
variable_name	The name of the variable that stores the input table name selected by the user. Use *variable_name* in the analytic script to reference the table. Do not include any spaces in *variable_name*. Do not use any of the following characters in *variable_name*. They are not supported: `` ` ~ ! @ # $ % ^ & * ( ) - + = { } [ ] | \ : ; ' " < > , . / ? ``
label	In client applications, the interface label that users see when prompted to select the table. For example, `Payments Table`
description optional	In client applications, descriptive text associated with the input field that users see. The description can be multiline, but it cannot skip lines. The description can help the user select the correct table. For example, `Select a table that lists payments and includes a check number column.`

Name	Description
	The description must be entered on the line below the associated `TABLE` tag.

Examples

Basic examples

`TABLE` tag with description to help user select the correct input table:

```
//TABLE v_table_payments Payments Table
  Select a table that lists payments and includes a check number column.
```

Advanced examples

Using a table defined in a TABLE tag in the script

The following script runs an AGE command on a table that is selected by the user from the data tables in the project:

```
COMMENT
//ANALYTIC example_script
//TABLE v_table_payments Payments Table
  Select a table that lists payments and includes a check number column.
END

OPEN %v_table_payments%
AGE ON payment_date CUTOFF 20141231 INTERVAL 0,30,60,90,120,10000
SUBTOTAL Payment_Amount TO r_output
CLOSE %v_table_payments%
```

FIELD tag

Defines a field that the user selects as input for an analytic script.

The field must be part of the table defined in the preceding `TABLE` tag. The first `FIELD` tag must immediately follow a `TABLE` tag, and can be followed by additional `FIELD` tags entered on sequential lines.

> **Note**
>
> The `TABLE` and `FIELD` tags require that an Analytics table pre-exists in the storage location in order to be available to be selected. For more information, see "DATA tag" on page 2545.
>
> Use the `TABLE` and `FIELD` tags if you want to create variables that allow users to specify different tables or fields for use with the same analytic script. If the script is designed to always work with the same table and set of fields, with names that do not change, you can hardcode the table and field names into the script and avoid use of the `TABLE` and `FIELD` tags.

Syntax

```
//FIELD variable_name type label
  <description>
```

Parameters

Name	Description
variable_name	The name of the variable that stores the input field name selected by the user. Use *variable_name* in the analytic script to reference the field. Do not include any spaces in *variable_name*. Do not use any of the following characters in *variable_name*. They are not supported: `` ` ~ ! @ # $ % ^ & * ( ) - + = { } [ ] | \ : ; ' " < > , . / ? ``
type	The types of fields that can be selected. Any type, or combination of types, from the following list can be selected: ◦ `C` - character data ◦ `N` - numeric data ◦ `D` - date, datetime, or time subtype of datetime data ◦ `L` - logical data

Name	Description
	Any computed fields in a table can be selected regardless of the *type* specified.
label	In client applications, the interface label that users see when prompted to select the field. For example, `Payment Date Field`
description optional	In client applications, descriptive text associated with the input field that users see. The description can be multiline, but it cannot skip lines. The description can help the user select the correct field. For example, `Select the column that contains the check payment date.` The description must be entered on the line below the associated `FIELD` tag.

Examples

Basic examples

Specifies a character field:

```
//FIELD v_last_name C Last Name Field
```

Specifies a character or numeric field:

```
//FIELD v_inv_num CN Invoice Number
```

Advanced Examples

TABLE with two accompanying FIELD tags

The following analytic header allows the user to specify two input fields from the *v_table_payments* table when the script runs:

```
COMMENT
//ANALYTIC test analytic
//TABLE v_table_payments Payments Table
  Select a table that lists payments and includes a check number column.
```

```
//FIELD v_check_num CN Check Number Field
//FIELD v_payment_date D Payment Date Field
  Select the column that contains the check payment date.
END

OPEN %v_table_payments%
EXTRACT FIELDS %v_check_num%, %v_payment_date% TO t_analyze
```

RESULT tag

Specifies that the results output by an analytic script are available in client applications to end users.

Output results, even if they exist, are not automatically made available.

> **Note**
> If your organization uses an on-premise Robots Agent, specifying a `RESULT` tag may upload data from the agent to the cloud-based Robots app in HighBond. For information, see "Uploads to the cloud-based Robots app" on page 2542.

Syntax

```
//RESULT type name
  <description>
```

Parameters

Name	Description
type	The type of result item: ◦ `TABLE` - an Analytics table and associated data file (.fil) ◦ `LOG` - an analytic log file ◦ `FILE` - a non-Analytics file **Note** Do not use `//RESULT LOG` or `//RESULT FILE` if your organization uses an on-premise Robots Agent and has disabled file uploads to Robots. For more information, see "Uploads to the cloud-based Robots app" on page 2542. For more information about logs, see "How log files are output" on page 2543.
name	The name of the result item. **Note** The *name* value must exactly match the name of the result item in the analytic script. You are not naming an item with *name*, you are matching a name specified in the script. You can use wildcard characters in *name* to assist with matching a name in the script. You cannot use a variable for *name*.

Name	Description
	Table name The *name* value specifies an Analytics table name. You must specify the table name, not the source data file name. Correct: `//RESULT TABLE Missing_Checks` Incorrect: `//RESULT TABLE Missing_Checks.fil` Do not include any spaces in *name*. Do not use any of the following characters in *name*. They are not supported: `! @ # $ % ^ & ( ) - + = { } [ ] | \ : ; ' " < > , / . ` ~` **Log name** Optional. The *name* value specifies an analytic log file name. If you do not specify *name*, the default log name is used: *analytic_name*.log. **Note** If you specify a log name, `SET LOG TO log_name` must appear in the script. Do not include any spaces in *name*. Do not use any of the following characters in *name*. They are not supported: `< > : " \ / |` **File name** The *name* value specifies a non-Analytics file name. You must specify the appropriate file extension for the type of non-Analytics file being output. Correct: `//RESULT FILE Missing_Checks.xlsx`

Name	Description

	Incorrect:
	`//RESULT FILE Missing_Checks`
	Do not include any spaces in *name*.
	Do not use any of the following characters in *name*. They are not supported:
	`< > : " \ / |`
	## Wildcard characters
	Use one or more wildcard characters in *name* to assist with matching a table, log, or file name in the script. Use a single asterisk (`*`) to substitute for zero or more consecutive characters.
	Patterns created by mixing wildcard and literal characters allow you to match all items of a particular type (for example, `*.xlsx`), or items where part of the name may change based on a variable definition in the script.

description optional	Descriptive text about the result or other information. The description can be multiline, but it cannot skip lines. The description appears in the analytic header only and is not visible to end users in client applications.

Examples

Basic examples

`RESULT` tag for an Analytics table:

```
//RESULT TABLE Missing_Checks
```

`RESULT` tag for an analytic log with the default name:

```
//RESULT LOG
```

`RESULT` tag for an analytic log with a specified name:

```
//RESULT LOG My_log_name
.
.
.
SET LOG TO My_log_name
```

`RESULT` tag for a specific Excel file:

```
//RESULT FILE Missing_Checks.xlsx
```

`RESULT` tag for all Excel files:

```
//RESULT FILE *.xlsx
```

Advanced examples

Table name with a month that varies

An output table name includes the month (invoices-jan, invoices-feb, and so on), so you specify `invoices-*` to ensure that the table is made available in the results regardless of the month suffix:

```
//RESULT TABLE invoices-*
```

Log name with a date that varies

A log file name includes a datestamp (prepare_invoice_table_31072019, and so on), so you specify `prepare_invoice_table_*` to ensure that the log file is made available in the results regardless of the datestamp:

```
//RESULT LOG prepare_invoice_table_*
```

File name with a month that varies

An output file name includes the month (invoices-jan.xlsx, invoices-feb.xlsx, and so on), so you specify `invoices-*.xlsx` to ensure that the file is made available in the results regardless of the month suffix:

```
//RESULT FILE invoices-*.xlsx
```

File name with a month and a format that vary

An output file name includes the month and is output in different formats (invoices-jan.xlsx, invoices-jan.del, and so on), so you specify `invoices-*.*` to ensure that the files are made available in the results regardless of the month suffix or file type:

```
//RESULT FILE invoices-*.*
```

Remarks

Uploads to the cloud-based Robots app

If your organization uses an on-premise Robots Agent, specifying a `RESULT` tag in an analytic header may upload data from the agent to the cloud-based Robots app in HighBond. All data is encrypted in transit, and when stored in Robots.

The **Permitted file uploads** configuration setting in Robots controls whether output results specified by the `RESULT` tag are:

- uploaded to Robots
- restricted to being output locally on the server where the Robots Agent is installed

For more information about the configuration setting, see Configuring a Robots Agent.

Analytic tag	"Permitted file uploads" setting:		
	Result files and logs only	Result tables, files, and logs	File uploads not permitted
`//RESULT TABLE`	Analytics result table layouts only are	Analytics result tables (layout and data) are	Analytics result table layouts only are

Analytic tag	"Permitted file uploads" setting:		
	Result files and logs only	Result tables, files, and logs	File uploads not permitted
	uploaded (field name, data type, field length) Result table data remains on the server in your network	uploaded	uploaded (field name, data type, field length) Result table data remains on the server in your network
`//RESULT LOG`	Analytics log files, for both successful and failed tasks, are uploaded	Analytics log files, for both successful and failed tasks, are uploaded	Do not specify, causes an analytic script to fail
`//RESULT FILE`	Non-Analytics result files (such as Excel) are uploaded	Non-Analytics result files (such as Excel) are uploaded	Do not specify, causes an analytic script to fail

How log files are output

How log files for analytic scripts are output depends on:

- whether a script succeeded or failed
- the application the script is running in
- the **Permitted file uploads** configuration setting (on-premise Robots Agent only)

  For more information about the configuration setting, see Configuring a Robots Agent.

Analytic script succeeded

On-premise Robots Agent	Cloud-based Robots Agent	AX Server	Analysis App window
◦ `RESULT LOG` **specified** log file uploaded to cloud-based Robots app, unless **Permitted file uploads** setting = "File uploads not permitted" ◦ `RESULT LOG` **not specified** no log file	◦ `RESULT LOG` **specified** log file output to cloud-based Robots app ◦ `RESULT LOG` **not specified** no log file	◦ `RESULT LOG` **specified** log file output to AX Server (available in client applications) ◦ `RESULT LOG` **not specified** no log file	◦ `RESULT LOG` **specified** log file output to Results tab ◦ `RESULT LOG` **not specified** no log file

Analytic script failed

On-premise Robots Agent	Cloud-based Robots Agent	AX Server	Analysis App window
◦ `RESULT LOG` tag not considered log file automatically uploaded to cloud-based Robots app, unless **Permitted file uploads** setting = "File uploads not permitted"	◦ `RESULT LOG` tag not considered log file automatically output to cloud-based Robots app	◦ `RESULT LOG` tag not considered log file automatically output to AX Server (available in client applications)	◦ `RESULT LOG` tag not considered log file automatically output to Results tab

Result file size limitation on AX Server

Result files are limited to a maximum of 2 GB for analytic scripts running on AX Server. If the file exceeds this size, results are not saved.

Result file storage and availability during script execution on AX Server

When you use the `RESULT FILE` tag, the file that is created is available for download from AX Web Client and AX Client once your script completes. This file is stored in the AX database and is not available on the file system of AX Server when the script is not running.

During the execution of your script, the file is available temporarily on the file system of AX Server and you can work with it using external processes, such as those you invoke using the `EXECUTE` command. While your script is running, external processes can access the file from the analytic job subfolder.

> **Note**
> By default, analytic job subfolders are located inside `ACL\Data\jobs`. Once the script completes, the analytic job subfolder is removed and the file is stored in the database.

DATA tag

Specifies that an Analytics table output by an analytic script is copied to a data subfolder (a storage location) in the deployment environment.

Typically, you store Analytics tables so that they can be used as input tables for subsequent analytic scripts.

> **Note**
> ACL Robotics with a cloud-based Robots Agent does not include a storage location for Analytics tables. The `DATA` tag is ignored in analytic scripts run with a cloud-based agent.

Syntax

```
//DATA table_name
  <description>
```

Parameters

Name	Description
table_name	The name of the Analytics table to be stored. **Note** The *table_name* value must exactly match the name of the Analytics output table in the analytic script. You are not naming a table with *table_name*, you are matching a table name specified in the script. You can use wildcard characters in *table_name* to assist with matching a table name in the script. You cannot use a variable for *table_name*. You must specify the table name, not the source data file name. Correct: `//DATA Missing_Checks` Incorrect:

Name	Description
	`//DATA Missing_Checks.fil` **Note** If an existing Analytics table in the data subfolder has the same name as the value you specify, the existing table is overwritten. ## Unsupported characters in the data table name Do not include any spaces in *table_name*. Do not use any of the following characters in *table_name*. They are not supported: `! @ # $ % ^ & ( ) - + = { } [ ] | \ : ; ' " < > , / . ` ~`  ## Wildcard characters  You can use wildcard characters in *table_name* if part of the table name may change. For example, if the table name depends on the month (invoices-jan, invoices-feb, and so on), specifying `invoices-*` ensures that the table is copied to the data subfolder regardless of the month suffix.  You can specify a single wildcard character to copy all Analytics output tables in the analytic script to the data subfolder:  `//DATA *` **Caution** Be careful when using wildcards characters. You may unintentionally overwrite existing data tables if the wildcard pattern that you specify matches unintended tables. As a best practice, make the value of *table_name* as specific as possible. Use wildcard characters only where they are required. ## Uploads to Robots For information about uploads to Robots, see "Uploads to the cloud-based Robots app" on page 2549.
description optional	Descriptive text about the output table or other information. The description can be multiline, but it cannot skip lines. The description appears in the analytic header only and is not visible to end users in client applications.

Examples

Copying an Analytics table to the storage location

The following analytic header specifies that the Invoices table, which is output in the associated script, is copied to the storage location:

```
COMMENT
//ANALYTIC Import Table
//DATA Invoices
END
IMPORT DELIMITED TO Invoices "Invoices.fil" FROM "Invoices.csv" 0
SEPARATOR "," QUALIFIER '"' CONSECUTIVE STARTLINE 1 KEEPTITLE ALLCHAR
ALLFIELDS
```

Remarks

Storing output tables

Output tables are not automatically copied to the storage location. You must use a `DATA` tag for each table that you want to store. You can include multiple `DATA` tags in an analytic header if necessary.

When should I use the DATA tag?

Two situations require use of the `DATA` tag and storing Analytics tables:

- output tables are used as input for subsequent analytic scripts
- users can select input tables or fields when scheduling an analytic script, or running it ad hoc

> **Note**
>
> If an entire data analysis process is completed using a single analytic script, use of the `DATA` tag is not required.
>
> The `DATA` tag is not intended to be used for specifying result tables. Use the `RESULT` tag instead. For more information, see "RESULT tag" on page 2538.

Output tables are used as input for subsequent analytic scripts

Depending on the deployment environment, and the structure of associated scripts, you may need to use the `DATA` tag to store an Analytics output table that you want to use in a subsequent analytic script.

During analytic script processing, Robots and AX Server use a temporary directory to store and access Analytics output tables, so you may not need to use the `DATA` tag.

The table below provides guidance.

Deployment environment	Use the DATA tag if…	Do not need the DATA tag if…
Robots (Enterprise Edition only)	◦ an Analytics table output in one robot task is required as input in another robot task	◦ an Analytics table is output and subsequently input during a sequence of analytic scripts run in a single robot task ◦ an entire data analysis process is completed using a single analytic script
AX Server	◦ an Analytics table output by one analytic script is required as input for another analytic script	◦ an entire data analysis process is completed using a single analytic script
Analysis App window	◦ an Analytics table output by one analytic script is required as input for another analytic script	◦ an entire data analysis process is completed using a single analytic script

Users can select input tables or fields

The `TABLE` and `FIELD` analytic tags create input parameters that allow a user to select an Analytics table, and to select fields from the table, for use as input to an analytic script. However, a table must pre-exist in the storage location in order to be available to be selected.

If you are developing an analytic script that allows a user to choose one or more input tables and fields, a prior analytic script with the `DATA` tag must run and save the appropriate table or tables to the storage location.

Locate output tables in the Source tables section in Robots

You can optionally add the `src_` prefix to an output table name to locate the output table in the **Source tables** section of the **Input/Output** tab in Robots.

```
//DATA src_Invoices
```

You must add the prefix to the table name in both the `DATA` tag and in the accompanying script.

The **Source tables** section allows you to visually separate tables that provide input for subsequent scripts. If no output table names have the `src_` prefix, the **Source tables** section does not appear in the **Input/Output** tab and all tables are located by default in the **Other tables** section.

Uploads to the cloud-based Robots app

With analytic scripts run in Robots installations, specifying `DATA` uploads the table layout only (field name, data type, field length) from the on-premise Robots Agent to the cloud-based Robots app in HighBond. Table data remains on your organization's network, within the Robots Agent directory.

All information is encrypted in transit.

Overwriting linked or shared tables in AX Server

If an output table overwrites a linked or shared table in AX Server, the table is changed to a standalone table.

PUBLISH tag

Specifies a file that contains metadata defining which Analytics tables to publish to AX Exception when an analytic script is finished processing.

Syntax

```
//PUBLISH filename
```

Parameters

Name	Description
filename	The name of the file containing publishing metadata for AX Exception.

Examples

The analytic header and the text file that specifies the publishing details for the analytic script

The `FILE` tag is required if the publish file is stored in the AX folder, so that the file is retrieved when the analytic script is processed.

```
COMMENT
//ANALYTIC Publish Results
//RESULT TABLE Results
//FILE ex_publish.txt
//PUBLISH ex_publish.txt
END
EXTRACT RECORD TO Results
```

The `ex_publish.txt` file uploaded to the **Related Files** subfolder in the collection contains the following line of text. The values must be quoted, and must use the following syntax: "table name","entity name","analytic name"

```
"Results","MyEntity","MyAnalytic"
```

Converting analytic scripts to Unicode

If you are migrating from the non-Unicode edition of Analytics to the Unicode edition, existing regular scripts and analytic scripts are automatically converted to Unicode. However, you must verify that the logic of the scripts remains the same when applied to double-byte Unicode data.

What is Unicode?

Unicode is a standard for encoding text that uses two or more bytes to represent each character, and characters for all languages are contained in a single character set. The Unicode editions of Galvanize products allow you to view and work with files and databases that contain Unicode-encoded data in all modern languages.

> **Note**
> Analytics and the AX Engine support little-endian (LE) encoded Unicode data. These products cannot be used to analyze big-endian (BE) encoded data.

Migrating to Unicode Analytics Exchange

- encryption of Unicode scripts is not currently supported
- Analytics project files and log files are encoded as Unicode data (UTF-16 LE) and cannot be used with the non-Unicode edition of Analytics
- when you use Analytics to define print image and delimited files that contain ASCII or EBCDIC-encoded text, the fields in the Analytics table containing this data are assigned the Unicode data type by default

Required analytic scripts changes

Update any parameters that specify a value in bytes

Characters in the non-Unicode edition of Analytics are one byte in length. Characters in the Unicode edition, if they are Unicode data, are two bytes in length. When you specify field length or starting position in bytes in the non-Unicode edition of Analytics, the number of bytes is equal to the number of characters. This is not true for Unicode data in the Unicode edition of Analytics.

To convert analytic scripts for use in Unicode Analytics, you must adjust the numeric value of any parameters that specify field length or starting position in bytes. For example, for an

`IMPORT DELIMITED` command that specifies a `WID` value of 7 in non-Unicode Analytics, you must double the `WID` value to 14 to produce the same result in Unicode Analytics.

In addition, for Unicode data, specify an odd-numbered starting byte position for fields, and an even number of bytes for field lengths. Specifying an even-numbered starting position, or an odd-numbered length, can cause characters to display incorrectly.

Recreate all instances of IMPORT PRINT and IMPORT DELIMITED

You need to recreate all instances of the `IMPORT PRINT` and `IMPORT DELIMITED` commands by importing the source data file using the Data Definition Wizard in the Unicode version of Analytics and reimporting the projects into AX Server. Using the Data Definition Wizard ensures that all syntax is valid.

Change all instances of the ZONED( ) and EBCDIC( ) functions

You need to change all instances of the `ZONED()` and `EBCDIC()` functions as follows so that the ASCII values returned by the functions are correctly converted to Unicode data:

- **Computed fields** – wrap the `BINTOSTR()` function around `ZONED()` or `EBCDIC()` instances
- **Static expressions** – wrap the `BINTOSTR()` function around `ZONED()` instances

```
BINTOSTR(ZONED(%result%, 5), 'A')
```

Change all instances of the OPEN FORMAT command

You need to modify all instances of the `OPEN FORMAT` command. You need to use the `SKIP` parameter to skip the first two bytes of the Unicode file you are opening. This is required because the first two bytes of UTF-16 encoded files are reserved as byte order marks and are separate from the text in the file.

Non-Unicode

```
OPEN "ascii_test.txt" FORMAT template_table CRLF
DEFINE FIELD full_rec ASCII 1 10
```

Unicode

```
OPEN "utf-16_test.txt" FORMAT template_table CRLF SKIP 2
DEFINE FIELD full_rec UNICODE 1 20
```

Verifying converted analytic scripts

Verify that the Unicode versions of the analytic scripts produce results that are identical to the results produced by the non-Unicode analytic scripts. The best way to do this is to use a Diff tool to compare the log files produced in the analysis. The Diff tool identifies any differences between the files.

What if the results are not the same?

If you cannot produce the same results with the Unicode version of an analytic script as the non-Unicode version, you may be able to isolate the problem by comparing the log outputs of the scripts at each step of the analysis.

Checking for Unicode compatibility

When upgrading to a Unicode edition, you need to verify that any custom logic you have added to scripts will produce the same results when run against Unicode data. There are predictable areas where scripts may be affected when they are run against Unicode data.

Bit and Character functions

Each of the functions listed below returns values based on byte locations or byte counts. You need to check to ensure that these functions are still being used correctly when you move from the single-byte representation of characters used in the non-Unicode edition to the double-byte character encoding used for Unicode data:

- ASCII( )
- BIT( )
- BYTE( )
- CHR( )
- DIGIT( )
- HEX( )
- MASK( )
- SHIFT( )

Byte length does not equal character length

You need to check the way the following functions are used in your scripts to ensure that they do not assume a one-to-one correspondence between the number of characters in data and the number of bytes.

If you find any instances where the logic assumes a one-to-one correspondence between characters and bytes, you must adjust the logic to work correctly with Unicode data, which uses two bytes to represent each character. Numbers specified as string function parameters, such as 4 in `STRING (1000, 4)` refer to the number of characters, so standard usage of these functions will not cause problems.

Conversion Functions

- PACKED( )
- STRING( )
- UNSIGNED( )

- VALUE( )
- ZONED( )

String functions

- AT( )
- BLANKS( )
- INSERT( )
- LAST( )
- LENGTH( )
- REPEAT( )
- SUBSTRING( )

Miscellaneous functions

- FILESIZE( )
- LEADING( )
- OFFSET( )
- RECLEN( )

Substituting Unicode-specific functions

Galvanize Unicode products support six Unicode-specific functions that support conversions between non-Unicode and Unicode data. The following functions are available in Galvanize Unicode products:

- **BINTOSTR()** – converts ZONED or EBCDIC data to its corresponding Unicode string. This ensures that values encoded as ZONED or EBCDIC data can be displayed correctly
- **DHEX( )** – returns the hexadecimal equivalent of a specified Unicode field value. This function is the inverse of HTOU( )
- **DBYTE( )** – returns the Unicode character interpretation of a double-byte character at a specified position in a record
- **DTOU( )** – converts a date value to the correct Unicode string display based on the specified locale setting
- **HTOU( )** – returns the Unicode equivalent of a specified hexadecimal string. This function is the inverse of DHEX( )
- **UTOD( )** – converts a locale-specific Unicode string to an Analytics date value

Analytic engine error codes

The following table lists the error codes that you may encounter when running analytic scripts.

Analytic engine startup errors

Error Code	Description
202	System error.
203	The evaluation period has expired.
204	The evaluation period has expired.
205	Activation failed.
206	Maximum number of sessions reached.
207	Memory initialization problem(s).
209	Unknown script error.
210	Database profile password is missing.
211	Server connection failure.
212	An unsupported command was encountered.
213	A dialog box was generated by the script.
256	The AX Engine failed to start.

Analytic engine error codes

Error Code	Description
1000	No preference file was specified. A new default preference file was created.
1001	There is a problem with the preference file. A new default preference file was created.
1002	The project has been upgraded from an earlier version. A copy was saved with a .old extension.

Error Code	Description
1003	The project file could not be processed. The last saved project was used.
1004	No project file specified.
1005	The specified project file does not exist.
1006	The specified project file is read-only.
1007	The specified project is currently being used by another application.
1008	The specified .old project file cannot be used. You must specify a project file with the .ACL extension.
1009	The specified project file is not an Analytics project file.
1011	The specified project file cannot be saved as an earlier version.
1012	Unable to open the log file for writing.
1013	No script was specified.
1014	The specified script does not exist.
1015	The Analytics license was not found or is invalid.
1016	A required library file (.dll) was not found.
1017	An unknown error occurred.

Command errors

The following table lists the error codes that are returned when an analytic script fails because of an invalid ACLScript command. The error code number returned identifies the command that failed.

Error Code	Command
1	SAMPLE
2	EXTRACT
3	LIST
4	TOTAL
5	DEFINE

Error Code	Command
6	COMMENT
7	QUIT
8	STOP
9	BYE
10	USE
11	OPEN
12	SAVE
13	DISPLAY
14	ACTIVATE
15	CLOSE
16	HELP
17	COUNT
18	STATISTICS
19	HISTOGRAM
20	STRATIFY
21	SUMMARIZE
22	EXPLAIN
23	GROUP
24	ELSE
25	END
26	CANCEL
27	SUBMIT
28	DELETE
29	RANDOM

Error Code	Command
30	SORT
31	FIND
32	DIRECTORY
33	TYPE
34	DUMP
35	INDEX
37	SET
40	DO
41	TOP
42	EXPORT
43	VERIFY
44	SEEK
45	JOIN
46	MERGE
47	SEQUENCE
48	CALCULATE
49	PRINT
50	LOCATE
51	RENAME
54	COPY
55	REPORT
56	EJECT
58	LET
59	ACCUMULATE

Error Code	Command
63	ACCEPT
64	ASSIGN
65	AGE
66	CLASSIFY
67	PROFILE
68	DO REPORT
69	LOOP
70	PAUSE
71	SIZE
72	EVALUATE
73	DIALOG
74	IF
75	GAPS
76	DUPS
77	SQLOPEN
78	PASSWORD
79	IMPORT
80	REFRESH
81	NOTIFY
82	CONNECT
83	RETRIEVE
84	FIELDSHIFT
85	BENFORD
86	CROSSTAB

Error Code	Command
87	(not used)
88	ESCAPE
89	NOTES
90	FUZZY DUPLICATE
91	EXECUTE
92	ACCESSDATA32
93	ACCESSDATA64
94	APPEND
95	RCOMMAND
96	CVSPREPARE
97	CVSSAMPLE
98	CVSEVALUATE
99	OUTLIER
100	FUZZYJOIN
101	CLUSTER
102	TRAIN
103	PREDICT

Analytic job processing errors

Error Code	Error message
-10	The analytic results could not be saved because the destination results folder was deleted after the analytic started running.
-11	Job was stopped.
-12	Stopped due to server shut down.

Error Code	Error message
-13	Failed to create results.
-16	Could not run due to server properties configuration error.
-17	Unable to create uniquely named results directory.
-19	Job was skipped.
-20	Could not prepare publish result tables.
-21	Could not publish results to *AX* Exception.
-22	Publish failed. Invalid table name.
-23	Publish failed. One or more of the table's column names are too long.
-24	Publish failed. Invalid values within data cells within an Analytics table.
-25	Publish failed. Not supported data types within table fields.
-26	Publish failed. Could not connect to *AX* Exception server.
-27	Job did not run. The user was removed or does not have permission.
-28	Job did not run. Unexpected error. See the server log and Analytics log for details.
-29	Could not copy data files. The analytic failed because the required data files could not be copied to the jobs directory.
-30	Job did not run. The analytic link is broken.
-31	Publish failed. The exception mapping file could not be located.
-32	Publish failed. Failed to parse the exception mapping file.
-34	Failed to store job results. Check that there is sufficient space on the drive storing the jobs folder and that no data files are locked.

This page intentionally left blank

ACL for Windows Installation and Activation Guide

ACL for Windows Installation and Activation Guide

This guide provides detailed instructions for installing ACL for Windows, which contains the following components:

- Analytics
- The Analysis App window
- Offline Projects

ACL for Windows installation quick start

(For detailed installation steps, see "Install ACL for Windows" on page 2583)

> **Note**
>
> When you install or upgrade Analytics, any existing Analytics sample data files are overwritten if they are in the Analytics working directory you specify during the installation or upgrade.
>
> If you have made changes to any of the sample projects or data files that you want to keep, save the files elsewhere before installing or upgrading, or rename the folder containing them. Do the same with the associated command log files if you want to preserve them.

Add certificate authority URLs to your network allowlist

If you are installing Analytics behind a network firewall, you may need to add certificate authority URLs to your network allowlist.

For more information, see "Add certificate authority URLs to your network allowlist" on page 2584.

Download the installer

Sign in to Launchpad (www.highbond.com) and download the ACL for Windows installation package (`ACLforWindows15.exe`).

If you do not have Launchpad sign-in credentials, see "Install ACL for Windows" on page 2583.

Extract the installation files

1. Double-click the ACL for Windows installation package (`ACLforWindows15.exe`).
2. If a security warning dialog box appears, verify the information listed, and click **Yes**.
3. Select the language you want to use for your installation and click **OK**.
4. In the **Setup Extraction Location** page, click **Extract**.

   After the files are extracted the installer starts automatically.

Install prerequisites, if required

If you are prompted to install prerequisites, click **Install**.

After the prerequisites are installed, the installer automatically proceeds.

Perform the ACL for Windows installation

Follow the onscreen instructions to install ACL for Windows.

In the **ACL Edition Selection** page, select the edition you want to install:

- **Non-Unicode**
- **Unicode**

> **Caution**
>
> Ensure that the edition you install is the correct edition for your company.
>
> For more information, see "Should I install the non-Unicode or Unicode edition of Analytics?" on page 2570

> **Note**
>
> If you are installing Analytics side by side with an earlier version, the installer enforces selecting the same edition as the currently installed version.
>
> Non-Unicode and Unicode editions cannot be installed side by side. For more information, see "Installing different versions or editions side by side" on page 2572.

Install optional Analytics data connectors and Python engine

Analytics offers data connectors to access data from a wide variety of supported data sources and a Python engine to enable Analytics machine learning commands. Some of these connectors and the Python engine are optional and you can choose to install them during the ACL for Windows installation.

> **Note**
> If you do not install the optional data connectors or the Python engine and want to use either of them later, you must uninstall and re-install ACL for Windows.

Activate Analytics

How you activate Analytics depends on your company's Launchpad authentication method:

- **Standard authentication** – If you have a single Analytics subscription, you probably use the standard authentication.
- **Authentication using a custom domain** – If your company uses Single Sign-On (SSO), you authenticate using a custom domain.

> **Note**
> If your company uses a custom domain, **do not** enter an email and password on the first screen of the Launchpad sign-in screen.

Standard authentication

1. Double-click the **ACL for Windows 15** shortcut on your desktop.
2. Enter your Launchpad sign-in credentials and click **Sign in**.
3. Select your HighBond instance, if you are prompted to do so, and click **Activate Analytics.**

   ACL for Windows opens.

Authentication using a custom domain

1. Double-click the **ACL for Windows 15** shortcut on your desktop.
2. Click **Sign in to a custom domain** at the bottom of the Launchpad sign-in screen.
3. Enter your company's custom domain and click **Continue**.

   If you do not know your custom domain, contact the Analytics account administrator in your company.
4. Enter your SSO credentials.
5. Make sure your HighBond instance is selected, and click **Activate Analytics**.

   ACL for Windows opens.

Start Analytics

To start working with Analytics, do one of the following:

To create a new, empty Analytics project	Under **Create**, click **Analytic Project**
To open an existing Analytics project	Under **Open**, click **Analytic Project**

To open a recently opened or a sample Analytics project (.acl)	Under **Recent Analytics Files**, or **Sample Files**, click the name of a project

ACL for Windows installation and activation overview

This section provides general information about the installation and activation of ACL for Windows, including guidance about whether to install the non-Unicode or Unicode editions.

ACL for Windows contains the following components:

- Analytics
- The Analysis App window
- Offline Projects

Should I install the non-Unicode or Unicode edition of Analytics?

Analytics is available in non-Unicode and Unicode editions. Both editions are contained in the same installation package, and during the installation you specify which edition to install.

You should install the non-Unicode edition, unless you have a requirement to view or analyze Unicode data. Unicode data can only be opened in the Unicode edition of Analytics.

You are more likely to encounter Unicode data if you work in an environment with global information systems, or you analyze data that contains multiple languages.

When the Unicode edition is required

You need to install the Unicode edition to view or analyze data with:

- Asian characters
- a combination of non-Unicode, or traditional, character encodings

  For example, some combination of languages from at least two of these character encodings:

  - Latin 1 (English and Western European)
  - Latin 2 (Central European)
  - Cyrillic
  - Greek
  - Arabic

> **Note**
>
> If you want to use the Chinese or Japanese Analytics user interface, the only option is to install the Unicode edition. This requirement is related to the language of the user interface, not to the language of the data.

Which edition of Analytics am I currently using?

To identify the edition of Analytics you are currently using, select **Help > About** to open the dialog box containing the product and subscription information. **Unicode** or **non-Unicode** appears after the version number.

Changing between non-Unicode and Unicode editions of Analytics

You cannot use the ACL for Windows installer to upgrade non-Unicode Analytics to Unicode Analytics, or vice versa.

To change editions of Analytics at the same time as you upgrade versions requires that you first uninstall the existing installation of Analytics, and then perform a fresh installation of the new version.

Language support

ACL for Windows is available in English and six other languages.

The table below summarizes the available languages and support for non-Unicode and Unicode editions across the available languages.

Language	Editions supported
Chinese	Unicode
English	non-Unicode, Unicode
French	non-Unicode, Unicode
German	non-Unicode, Unicode
Japanese	Unicode
Portuguese	non-Unicode, Unicode
Spanish	non-Unicode, Unicode

Installation – summary of tasks

Perform the following tasks to install ACL for Windows:

1. **Check requirements** – Check the configuration of the computer where ACL for Windows will be installed to ensure that it meets the minimum software and hardware requirements.

For more information, see "ACL for Windows system requirements" on page 2611.

2. **Admin rights** – Ensure that you have Administrator rights on the computer on which you are performing the installation.
3. **Download** – Using the information in the Welcome email from Galvanize, or the email notification from HighBond, download the ACL for Windows installation package from Launchpad (www.highbond.com).
4. **Install** – Install ACL for Windows using the instructions in this guide.
5. **Activate** – Activate Analytics.

   For detailed information about installing and activating Analytics, see "Install ACL for Windows" on page 2583.

   For information about activation and licensing, see Launchpad Help.

Future upgrades

Once you have installed or upgraded to Analytics 15, you are automatically informed of all new versions as they become available, and given the option to download and install the upgrade.

Installing different versions or editions side by side

Analytics 15 can be installed side by side with any previous version of Analytics or ACL Desktop prior to version 15.

Unicode and non-Unicode editions of Analytics or ACL Desktop can never be installed side by side, regardless of the version number.

> **Caution**
>
> Data connectors in a previous version of Analytics may no longer work if Analytics 15 is installed side by side with the previous version. If you want to ensure that the data connectors in the previous version continue to work, do not install the two versions side by side on the same computer.
>
> The data connectors are contained in the Data Access window.

Silent installation

IT administrators can install ACL for Windows silently, without requiring user interaction. For more information, see "Install ACL for Windows using silent installation" on page 2591.

Activation and account administration

ACL for Windows is subscription-based software. During the installation, you are not required to enter a serial number. Instead, you must activate the software before first use.

You activate the software by signing in to Launchpad (www.highbond.com), the portal for downloading all Galvanize on-premise software, and accessing customer services and resources. Instructions for signing in to Launchpad are included in a welcome email from Galvanize, sent to all licensed users.

Analytics account administrator

The Analytics account administrator is typically the primary contact that Galvanize has on file for a company. They are responsible for managing their company's subscription. A different account administrator, or additional account administrators, can be designated if required. For more information, contact Support.

Individual customers with a single Analytics license are by default the Analytics account administrator.

Centralized account management using Launchpad

Launchpad provides centralized account management, which allows Analytics account administrators to perform the following tasks:

- Invite Analytics users in their company to one or more HighBond instances

  Inviting a user to an instance automatically adds the user to Launchpad. Once added to Launchpad, users are eligible to activate an available ACL for Windows license.
- Remove users from instances
- Revoke ACL for Windows licenses from individual users
- View subscription information
- Update organization settings

A HighBond instance can correlate directly with a customer's company, or a company can set up two or more HighBond instances to reflect different divisions or operating units.

For more information, see HighBond Help.

Distributing the ACL for Windows installer

Galvanize software is subscription based and Analytics account administrators can download the ACL for Windows installation package and distribute it to Analytics users within their company. Distributing the installation package internally can be easier than requiring each user to download it individually.

There is no restriction on the number of users who can install the software. However, the terms of your company's software subscription dictate the maximum number of licenses that can be assigned at any one time. For more information, see Galvanize on-premises software subscriptions.

Installed software that has not been activated using an available license is non-functional.

Multi-device installation and activation

Each user with an ACL for Windows subscription can install the software on multiple computers **as long as the software is for their own use only, on all computers on which it is installed**. For example, any of the following multi-device installations are permitted:

- a work computer and a home computer
- a desktop computer and a laptop computer used when traveling
- two work computers, one used for running scripts, and another used for developing scripts or performing ad hoc analysis

The software must be activated on each computer where it is installed. Software activation is tracked by Galvanize. If two different users want to use the software, two separate licenses are required. As the Galvanize Master Subscription Agreement states:

A Named User's ID and password may not be shared with any other individual. Sharing or pooling a Named User's access between multiple individuals to allow for temporary use by multiple users in a department or organization is strictly prohibited.

Settings for configurable options

Configurable options are the settings in the **Options** dialog box in Analytics.

Global settings

Any changes you have made to the global settings for configurable options in Analytics, versions 10 and onward, are replicated in a version 15 installation.

Changes made to global settings in ACL Desktop 9.3 or earlier are not replicated in version 15. After completing the version 15 installation, you can recreate any customized settings in the **Options** dialog box.

Project-specific settings

If you have Analytics projects with project-specific settings for configurable options, these settings are preserved when you install Analytics 15.

Side by side installations of Analytics

If Analytics 15 is installed side by side with a previous version of Analytics, global or project-specific preference settings subsequently specified in either version are treated separately and do not affect one another.

More information

For more information about global and project-specific settings for configurable options, see "How Analytics preferences files work" on page 146.

Offline Projects included in the ACL for Windows installer

Offline Projects is included in the ACL for Windows installation package and it is automatically installed when you install Analytics.

Offline Projects is a desktop-based application that allows HighBond users to continue to work with fieldwork sections from Projects while disconnected from HighBond.

If you are not a HighBond user, Offline Projects is disabled on your computer.

Galvanize Unicode products

The Unicode editions of Galvanize products allow you to view and work with files that contain Unicode data.

Unicode is an industry-standard method of character encoding that supports most of the languages of the world.

Should I install the non-Unicode or Unicode edition of Analytics?

Analytics is available in non-Unicode and Unicode editions. Both editions are contained in the same installation package, and during the installation you specify which edition to install.

You should install the non-Unicode edition, unless you have a requirement to view or analyze Unicode data. Unicode data can only be opened in the Unicode edition of Analytics.

You are more likely to encounter Unicode data if you work in an environment with global information systems, or you analyze data that contains multiple languages.

When the Unicode edition is required

You need to install the Unicode edition to view or analyze data with:

- Asian characters
- a combination of non-Unicode, or traditional, character encodings

  For example, some combination of languages from at least two of these character encodings:

  - Latin 1 (English and Western European)
  - Latin 2 (Central European)
  - Cyrillic
  - Greek
  - Arabic

> **Note**
>
> If you want to use the Chinese or Japanese Analytics user interface, the only option is to install the Unicode edition. This requirement is related to the language of the user interface, not to the language of the data.

Unilingual data

If the data you work with is English-only, or uses only one of the Western European languages, you should most likely install the non-Unicode edition. You should be aware, however, that it is possible

for an English-only file to be Unicode.

> **Note**
> Contact your IT department if you are uncertain about the character encoding you might encounter when working with your organization's data.

Using non-Unicode Analytics with Unicode data

In some situations it is possible, and preferable, to use non-Unicode Analytics with Unicode data.

If all the characters in the Unicode data you work with are supported by one of the traditional character encodings – for example, English-only data – there is no need to use Unicode Analytics. When you import this data into non-Unicode Analytics, text fields are automatically converted from Unicode to ASCII, with no loss or corruption of data.

For the reasons why this approach is preferable, see "Drawbacks of the Unicode edition" on the next page.

> **Note**
> Data corruption occurs if you import Unicode data to non-Unicode Analytics and the data contains characters not supported by the extended ASCII character set.

The language of the data is what matters

The language, or languages, of the data you work with is generally what dictates the edition of Analytics you should install, not the language of the Analytics user interface.

For example, your organization might use the Spanish Analytics interface, but the decision about whether to install the non-Unicode or Unicode edition depends on the language or languages you expect to encounter in the data.

The Chinese and Japanese Analytics user interfaces are an exception to the general guideline about choosing an edition of Analytics. Both interfaces are available in the Unicode edition only. For information about localized Analytics interfaces, and Unicode support, see "Language support" on page 2571.

Which edition of Analytics am I currently using?

To identify the edition of Analytics you are currently using, select **Help > About** to open the dialog box containing the product and subscription information. **Unicode** or **non-Unicode** appears after the version number.

Robots or Analytics Exchange users

You need to install the edition of Analytics that matches the edition of Robots or Analytics Exchange that your organization uses. Analytics cannot interact with Robots or Analytics Exchange if the editions are mismatched.

Drawbacks of the Unicode edition

The Unicode edition of Analytics has these drawbacks:

- **Larger data file sizes** – Unicode data requires approximately double the storage space of non-Unicode data because each character is represented using two bytes instead of one.
- **Possible slower performance** – With large data files, some Analytics commands may take noticeably longer to execute because twice the amount of data is being processed by the Unicode edition.

Because of these drawbacks, you should install the Unicode edition only if you actually need it to work with Unicode data.

Single-byte versus double-byte data in Analytics

Non-Unicode Analytics

When reading and writing data files, the non-Unicode edition of Analytics works with single-byte character sets (SBCS) only. In a single-byte character set, one byte of data is used to represent each character, and a maximum of 256 different characters are supported.

The single-byte character set used by non-Unicode Analytics depends on the language specified by your computer's **system locale** setting. If the system locale specifies English or one of the Western European languages, the Windows-1252 character set is used. Windows-1252 is also known as "Windows Latin 1". You can set your system locale in the Windows Control Panel.

Other common ways of referring to single-byte character sets are "ANSI", "ANSI character set", or "extended ASCII".

> **Note**
> The character set that non-Unicode Analytics uses for processing data is not necessarily the same as the character set used by the text on the Analytics user interface.

Unicode Analytics

Reading data

The Unicode edition of Analytics can read double-byte or single-byte character sets. Double-byte Unicode characters use two bytes of data to represent each character. By using two (or more) bytes of data to encode characters, Unicode has the capacity to represent all the characters of the world's languages in a single character set.

Writing data

For write operations that create output files, Unicode Analytics typically uses double-byte UTF-16 character encoding. For some operations, the output file retains any single-byte character encoding that is present in the source file.

Number of bytes versus number of characters

When working with double-byte Unicode data, keep in mind the distinction between the length of a field in bytes, which appears in the **Table Layout** dialog box, and the length of a field in characters.

For example, if the length of a Unicode field is 44 bytes in the **Table Layout** dialog box, the field actually contains 22 characters.

Why bytes and characters matter in ACLScript

When you use functions such as STRING( ) and SUBSTRING( ), which include a field length parameter, you specify the length in characters, not bytes. Conversely, some commands, such as DEFINE FIELD, require that you specify field length in bytes, not characters.

In non-Unicode Analytics, one byte equals one character, so the distinction between bytes and characters does not matter. But in Unicode Analytics, when working with double-byte Unicode data, two bytes equal one character, so the distinction does matter.

Details about which type of unit to use are included in the ACLScript documentation for particular commands and functions.

Importing text files to Unicode Analytics

The character encoding of a text file affects how it is imported to Unicode Analytics, and the data type used for character fields in the resulting Analytics table.

When importing ASCII and EBCDIC files to Unicode Analytics you have two choices:

- Convert the character data type to UNICODE and create an Analytics data file

  If you subsequently change the UNICODE data type to ASCII or EBCDIC, the characters in the fields will not display correctly.
- Retain the ASCII or EBCDIC character encoding, and create an Analytics table layout only without an Analytics data file

  The Analytics table layout continues to be linked to the source text file.

Text file character encoding	Data Definition Wizard option	Character data type in Analytics table	Character length
UTF-16 LE (Unicode)	**Unicode Text**	UNICODE	double-byte character
UTF-8 (Unicode)	**Encoded Text** + the appropriate character set (code page) for the data file	UNICODE	double-byte character
extended ASCII (ANSI character set)	**ASCII > Delimited text file** **ASCII > Print Image (Report) file**	UNICODE	double-byte character
	ASCII > Other file format	ASCII	single-byte character
EBCDIC	**EBCDIC > Print Image (Report) File**	UNICODE	double-byte character
	EBCDIC > Other file format	EBCDIC	single-byte character

Little-endian and big-endian data

“Little-endian” (LE) and “big-endian” (BE) are terms that refer to two different methods of encoding Unicode data. Unicode data that originates from Microsoft Windows computers is typically encoded as little-endian. If you use Analytics on a Windows computer, you cannot analyze big-endian data.

Conversion of non-Unicode Analytics projects to Unicode

You can open a non-Unicode Analytics project in the Unicode edition of Analytics, but you cannot do the reverse: open a Unicode Analytics project in non-Unicode Analytics.

	Open in non-Unicode Analytics	Open in Unicode Analytics
non-Unicode project	Yes	Yes
Unicode project	No	Yes

Project conversion

When you open a non-Unicode Analytics project in Unicode Analytics, you are prompted to automatically convert the project and the associated log file to Unicode. If you proceed with the conversion, copies of the original non-Unicode project and the log file are saved with the file extension .OLD, and are not altered.

> **Note**
> Once you convert a non-Unicode Analytics project to Unicode, you can no longer open it in the non-Unicode edition of Analytics, and you cannot convert the project back to non-Unicode. If required, you can recover the non-Unicode version of the project using the .OLD file.

Analytics data files

When you convert a non-Unicode Analytics project to Unicode, any associated Analytics data files (.fil) are not converted to Unicode. They remain as single-byte ASCII (ANSI) data in the Unicode project.

> **Note**
> In Unicode Analytics, byte position or byte length of fields in the unconverted single-byte data work the same way as they do in non-Unicode Analytics. One byte equals one character. Keep this difference in mind if you execute any commands against the unconverted data that reference byte position or byte length.

Unicode-specific functions in Analytics

Analytics has six Unicode-specific functions to aid with data analysis and conversion. The functions are summarized in the table below. The functions are only included in the Unicode edition of Analytics.

Function	Purpose
BINTOSTR( )	Returns Unicode character data converted from ZONED or EBCDIC character data. Abbreviation for "Binary to String". This conversion ensures that values encoded in ZONED or EBCDIC can be displayed correctly.

Function	Purpose
DBYTE( )	Returns the Unicode character located at the specified byte position in a record.
DHEX( )	Converts a Unicode string to a hexadecimal string. The inverse of HTOU( ).
HTOU( )	Converts a hexadecimal string to a Unicode string. Abbreviation for "Hexadecimal to Unicode". The inverse of DHEX( ).
DTOU( )	Converts an Analytics date value to a Unicode string in the specified language and locale format. Abbreviation for "Date to Unicode". The inverse of UTOD( ).
UTOD( )	Converts a Unicode string containing a formatted date to an Analytics date value. Abbreviation for "Unicode to Date". The inverse of DTOU( ).

Install ACL for Windows

The procedure below outlines the steps for installing ACL for Windows.

> **Note**
> IT administrators can also install ACL for Windows silently, without requiring user interaction. For more information, see "Install ACL for Windows using silent installation" on page 2591.

If you are the primary contact at your company

If you are the primary contact at your company, you should have received a welcome email from Galvanize with instructions for signing in to your Launchpad account.

If you have not received the welcome email, contact Support for assistance.

> **Note**
> You must be able to sign in to a Launchpad account in order to use ACL for Windows, and to grant licenses to other ACL for Windows users in your company.

If you are not the primary contact

If you are not the primary contact, you should have received an email notification when the primary contact, or your company's Analytics account administrator, added you to Launchpad. The email notification contains instructions for signing in to Launchpad. **You must be able to sign in to Launchpad in order to use ACL for Windows.**

If you have not received the email notification, contact your company's Analytics account administrator. Also check your spam filter. The email notification should be sent from notifications@highbond.com.

Important information about the Microsoft .NET Framework prerequisite

Windows 8.1 Update KB2919355 is required by Microsoft .NET Framework 4.6.x, which in turn is required by ACL for Windows 15.

If you are using Windows 8.1, and .NET 4.6.x is not installed, and you have not run Update KB2919355, the ACL for Windows installer terminates with an error message during the .NET 4.6.2 prerequisite installation.

You need to download and install Update KB2919355 before you can continue with the ACL for Windows installation.

Alternatively, you can install Update KB2919355 before you begin the ACL for Windows installation and avoid the error message.

Analytics projects and data files

When you install or upgrade Analytics, any Analytics projects you previously created, and any data files, are not affected and are available to use in the new version.

Analytics sample data files

When you install or upgrade Analytics, any existing Analytics sample data files are overwritten if they are in the Analytics working directory you specify during the installation or upgrade.

The default location for the working directory is:

`C:\Users\user_account_name\Documents\ACL Data\Sample Data Files`

The default location for the working directory prior to version 11 is:

`C:\ACL DATA\Sample Data Files`

> **Caution**
> If you have made changes to any of the sample projects or data files that you want to keep, save the files elsewhere before installing or upgrading, or rename the folder containing them. Do the same with the associated command log files if you want to preserve them.

Add certificate authority URLs to your network allowlist

If you are installing Analytics behind a network firewall, the application's digital certificate must be able to connect to URLs associated with the third-party certificate authority. If required, add the certificate authority URL or URLs to your network allowlist. You can allow the certificate authority's web site generally – for example, *.digicert.com – or you can allow only specific URLs. After installing Analytics, inspect the installed certificate to get the required URLs.

Show me how

1. In the Analytics installation directory, right-click `ACLWin.exe` and select **Properties**.

   The default installation directory is: `C:\Program Files (x86)\ACL Software\ACL for Windows 15\`
2. In the **ACLWin.exe Properties** dialog box, select the **Digital Signatures** tab.
3. In the **Signature list**, double-click **Galvanize**, then click **View Certificate**.
4. In the **Details** tab in the **Certificate** dialog box, select the following fields, copy the URLs from the display area at the bottom of the dialog box, and save them in a text file:

Field	Value	URL (example)
CRL Distribution Points	CRL Distribution Point	http://crl3.digicert.com/sha2-assured-cs-g1.crl
	CRL Distribution Point	http://crl4.digicert.com/sha2-assured-cs-g1.crl
Authority Information Access	Access Method=On-line Certificate Status Protocol	http://ocsp.digicert.com

5. Select the **Certification Path** tab, select the intermediate entry in the certification path (entry 2 of 3) and click **View Certificate**.
6. In the **Details** tab, select the following fields, copy the URLs from the display area at the bottom of the dialog box, and save them in the text file:

Field	Value	URL (example)
Authority Information Access	Access Method=On-line Certificate Status Protocol	http://ocsp.digicert.com
CRL Distribution Points	CRL Distribution Point	http://crl4.digicert.com/DigiCertAssuredIDRootCA.crl
	CRL Distribution Point	http://crl3.digicert.com/DigiCertAssuredIDRootCA.crl

7. Add the URLs that you have collected to your network allowlist.

Background

The Analytics application executable uses a digital certificate chain to ensure its authenticity and integrity. The certificate chain requires an Internet connection to the third-party certificate authority so that the certificates in the chain can be verified. Your network firewall must allow access to the appropriate URLs associated with the certificate authority in order for this process to work. Without the required access, you will not be able to use Analytics.

Installation steps

> **Caution**
>
> If you are prompted to restart your computer at any point during the installation process, do so right away. **Do not ignore messages to restart your computer.**
>
> If you do not restart your computer when you are prompted, you may cause problems with the installation of .NET, other prerequisites, or ACL for Windows.

Download the installer and begin the installation

1. Download the ACL for Windows installation package (`ACLforWindows15.exe`) from Launchpad:
   a. Sign in to Launchpad (www.highbond.com).
   b. Under **Resources**, click **Downloads**.
   c. In the top menu, click **ACL for Windows**.
   d. Click **Download Version 15**.

   A company's Analytics account administrator can also supply other users with the installation package.

   The installation package contains both the non-Unicode and Unicode editions of Analytics.
2. Close all other Windows applications.
3. Double-click the installation package.
4. If a security warning dialog box appears, verify the information listed, and click **Yes**.
5. Select the language you want to use for your installation and click **OK**.

Extract the installation files

In the **Setup Extraction Location** page, specify the folder where the installation files will be extracted, and click **Extract**.

You can click **Browse** to select a folder, or accept the default location:

`C:\Users\`*`user_account_name`*`\Downloads\ACL Installers\ACLforWindows15`

The setup files are extracted to the destination folder you specified. After the files are extracted, the installer starts automatically.

> **Note**
>
> If the installer fails to start automatically for any reason, you can use Windows Explorer to navigate to the folder where the setup files were extracted and double-click `setup.exe`.

Install prerequisites, if required

If you are prompted to install prerequisites, click **Install**.

After the prerequisites are installed, the installer automatically proceeds.

For a list of prerequisites, see "Automatically installed prerequisites" on page 2614.

> **Note**
> If the ACL for Windows installer terminates with an error message during the .NET 4.6.2 prerequisite installation, you need to download and install Windows 8.1 Update KB2919355, restart your computer, and then restart the ACL for Windows installer.

Perform the main ACL for Windows installation

1. In the **Welcome** page, click **Next**.
2. In the **License Agreement** page, select **I agree to the above terms and the Galvanize Master Subscription Agreement** and click **Next**.
3. In the **ACL Edition Selection** page, select the edition you want to install and click **Next**:
   - **Non-Unicode**
   - **Unicode**

   > **Caution**
   > Ensure that the edition you install is the correct edition for your organization.
   >
   > For more information, see "Should I install the non-Unicode or Unicode edition of Analytics?" on page 2570

   > **Note**
   > If you are installing Analytics side by side with an earlier version, the installer enforces selecting the same edition as the currently installed version.
   >
   > Non-Unicode and Unicode editions cannot be installed side by side.

4. In the **Optional Analytics data connectors** page, click **Next** to allow installing the optional data connectors for Analytics.

   If you do not want to install the optional data connectors listed on the page, deselect **Install optional Analytics data connectors** before clicking **Next**.

   If you want to install the optional data connectors in the future, you will need to uninstall and re-install ACL for Windows.

5. In the **Enable machine learning** page, click **Next** to allow installing the Python engine to enable Analytics machine learning commands.

   If you do not want to install the optional Python engine, deselect **Enable machine learning (install Python engine for Analytics)** before clicking **Next**.

If you want to install the Python engine in the future, you will need to uninstall and re-install ACL for Windows.

6. In the **Destination Folder** page, specify the locations where the ACL for Windows application files and the Analytics sample data files will be installed and then click **Next**.

   If necessary, click **Change** to modify either, or both, of the default locations.

   The location where you install the sample data files becomes the default Analytics working directory.

   The default locations:

ACL for Windows application files (64-bit operating systems)	`C:\Program Files (x86)\ACL Software\ACL for Windows 15\`
ACL for Windows application files (32-bit operating systems)	`C:\Program Files\ACL Software\ACL for Windows 15\`
Analytics sample data files	`C:\Users\`*`user_account_name`*`\Documents\ACL Data\Sample Data Files\`

7. In the **Ready to Install the Program** page, click **Install**.

   ACL for Windows is installed or upgraded.

8. When the installation process is complete, click **Finish**.

Activate Analytics

> **Note**
> Analytics may already be activated if you are installing it side by side with an earlier version.

How you activate Analytics depends on your company's Launchpad authentication method:

- **Standard authentication** – If you have a single Analytics subscription, you probably use the standard authentication.
- **Authentication using a custom domain** – If your company uses Single Sign-On (SSO), you authenticate using a custom domain.

> **Note**
> If your company uses a custom domain, **do not** enter an email and password on the first screen of the Launchpad sign-in screen.

Standard authentication

1. Double-click the **ACL for Windows 15** shortcut on your desktop.

   The Launchpad sign-in screen opens.
2. Enter your Launchpad user name (email) and password and click **Sign in**.

   You should already have these credentials based on the instructions in the welcome email or the email notification.
3. Select your HighBond instance if you are prompted to do so, and click **Activate Analytics**.

   ACL for Windows opens.

   You can also click **Continue without activating a license** if you only need to use Offline Projects.

Authentication using a custom domain

1. Double-click the **ACL for Windows 15** shortcut on your desktop.

   The Launchpad sign-in screen opens.
2. Click **Sign in to a custom domain** at the bottom of the Launchpad sign-in screen.

   > **Note**
   > This method of signing in is only available if it has been set up by your company.

3. Enter your company's custom domain and click **Continue**.

   If you do not know your custom domain, contact the Analytics account administrator in your company.
4. Enter your SSO credentials.
5. Make sure your HighBond instance is selected, and click **Activate Analytics**.

   ACL for Windows opens.

   You can also click **Continue without activating a license** if you only need to use Offline Projects.

Start Analytics

To start working with Analytics, do one of the following:

To create a new, empty Analytics project	Under **Create**, click **Analytic Project**
To open an existing Analytics project	Under **Open**, click **Analytic Project**
To open an existing analysis app	Under **Open**, click **Analysis App**
To open a recently used or a sample Analytics project	Under **Recent Analytics Files**, or **Sample Files**, click

(.acl) or analysis app (.aclx)	the name of a file
◦ If you create or open an Analytics project, it opens in Analytics. ◦ If you open an analysis app, it opens in the Analysis App window.	

Display the toolbar and command line

In Analytics, to display the toolbar or command line, select **Window > Tool bar** or **Window > Command Line**.

Install ACL for Windows using silent installation

> **Important**
>
> If you intend to silently install ACL for Windows on a computer with the Windows 8.1 operating system, you must first install Windows Update KB2919355.
>
> This requirement applies only to those computers that do not have the Microsoft .NET 4.6.x prerequisite already installed.
>
> For more information, see "ACL for Windows system requirements" on page 2611.

> **Note**
>
> Silent installation of ACL for Windows does not support excluding the Python engine or the optional data connectors. Silent installation automatically installs these optional components.
>
> If you want to exclude the optional components, you must perform the installation using the installation wizard.
>
> For more information, see "Perform the main ACL for Windows installation" on page 2587.

If you need to deploy ACL for Windows to a large number of workstations, you can use silent installation to run the installer without requiring user interaction.

You can use silent installation to perform a fresh installation of ACL for Windows 15.

After you perform the silent installation, you need to activate ACL for Windows. For the steps to activate ACL for Windows, see "Activate Analytics" on page 2588.

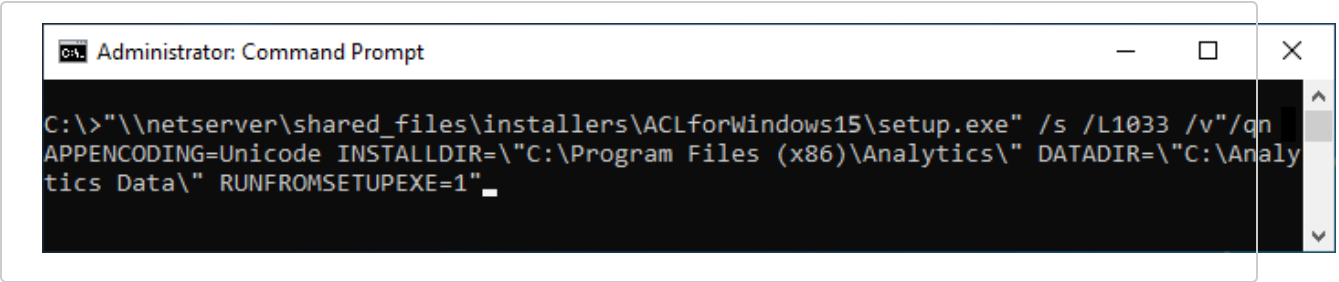
How it works

Silent installation uses the Windows command line to run the ACL for Windows installer with preselected options and no user interface.

The command uses either of the installer files from the installation package. See "Two installer files" on the next page to decide which one you should use.

Example

The figure below provides an example of silent installation syntax in the Windows command line.



Extracting the installer files

In order to access the installer files, download the ACL for Windows installation package (`ACLforWindows15.exe`) from Launchpad (www.highbond.com). Run the installation package to extract the two installer files.

The installation package is designed to automatically start the installation process after the files are extracted. If you are using the silent installation option, click **Cancel** as soon as the files have been extracted.

The default location for the extracted files is:

`C:\Users\user_account_name\Downloads\ACL Installers\ACLforWindows15\`

Two installer files

The ACL for Windows installation package (`ACLforWindows15.exe`) contains two installer files:

- `setup.exe`
- `ACL for Windows.msi`

The installer file you use to perform the silent installation depends on whether the required software prerequisites are already installed on the target computer.

For a list of the software prerequisites, see "ACL for Windows system requirements" on page 2611.

setup.exe

`setup.exe` installs the required software prerequisites on the target computer if they are not already installed.

ACL for Windows.msi

All software prerequisites must already be installed on the target computer in order to use `ACL for Windows.msi`.

Run the silent installation

1. Double-click the installation package `ACLforWindows15.exe` and click **Cancel** as soon as the installers are extracted.
2. Open the Windows command prompt as an administrator.

   There are different methods for opening the command prompt as an administrator, depending on your version of Windows.

   This method works for all versions of Windows:

   In the `C:\windows\system32` sub-folder in Windows Explorer, right-click `cmd.exe` and select **Run as administrator**.
3. Run one of the following silent installation commands.

> **Note**
>
> Specify the full path to an installer file. Enclose the path in quotation marks if the path includes any spaces.
>
> The examples below provide generic syntax only. Detailed syntax guidelines, and additional information about silent installation, appear in subsequent sections.

To do this	Use this file	With this command
Install ACL for Windows 15	`setup.exe`	`"setup_exe_path_and_filename" /s /L<language ID> /v"/qn APPENCODING=<Analytics Edition> RUNFROMSETUPEXE=1"`
	`ACL for Windows.msi`	`msiexec /i "msi_path_and_filename" TRANSFORMS=<language ID>.mst APPENCODING=<Analytics Edition> /qn`

Silent installation guidelines

Review and follow the guidelines for entering the silent installation command syntax in the Windows command line.

> **Caution**
>
> Not following any of the guidelines will cause the silent installation to fail.

Guideline	Details
Run as administrator	To use the .msi file to silently install ACL for Windows you need to run the Windows command line as an administrator. Right-click **cmd.exe** and select **Run as administrator**.
Wrapping syntax	Do not enter any line breaks in the syntax. Allow the Windows command line to automatically wrap syntax to the next line.
Separating parameters	Separate parameters with a space. For example: `TRANSFORMS=1033.mst INSTALLDIR="C:\Program Files\Analytics"` The one exception is the following piece of syntax, which must be entered without any spaces: /v"/qn"
Values with spaces	Any parameter values or network paths that include spaces must be enclosed in quotation marks. For example: ◦ DATADIR="C:\Analytics Data" ◦ "\\nas-server-2\installers\ACLforWindows15\ACL for Windows.msi"
Multiple parameters following the /v parameter	Multiple parameters following the /v parameter must be enclosed in quotation marks. For example: `/v"/qn INSTALLDIR=C:\Analytics DATADIR=C:\Analytics_Data"`
Quotation marks within quotation marks	Quotation marks within quotation marks must use a backslash as an escape character (\"). For example: `/v"/qn DATADIR=\"C:\Analytics Data\""` In the example above, quotation marks are required around the entirety of the two parameters (/qn and DATADIR), and quotation marks are also required around a parameter value that includes a space (C:\Analytics Data). The escape character must precede each quotation mark in the internal set of quotation marks.
Format of quotation marks	If you copy and paste syntax into the Windows command line, make sure any quotation marks are straight quotation marks rather than curly quotation marks.

Command line parameters

`setup.exe` parameters

Parameter	Details
"*setup_exe_path_and_file-name*"	Specifies the network or local path to the `setup.exe` installer file. The path must include the file name and file extension. Use this parameter if the mandatory prerequisites have not yet been installed on the end user's computer.
/s	Specifies that `setup.exe` runs in silent mode.
/L<*language ID*>	Specifies the language of the ACL for Windows user interface.
/v	Specifies that parameter values will be passed to the installer. ◦ Do not include a space between /v and the first parameter. ◦ If more than one parameter follows /v, the list of parameters must be enclosed in quotation marks.
/RUNFROMSETUPEXE=1	Use this parameter and a value of 1 to specify that the silent installation is using the `setup.exe` installer file.

`ACL for Windows.msi` parameters

Parameter	Details
msiexec /i "*msi_path_and_file-name*"	Specifies the network or local path to the `ACL for Windows.msi` installer file. The path must include the file name and file extension. Use this parameter if all prerequisites have already been installed on the end user's computer.
TRANSFORMS=*language ID*	Specifies the language of the ACL for Windows user interface.

General parameters - `setup.exe` and `ACL for Windows.msi`

Parameter	Details
/qn	Specifies that the installer runs in silent mode with no user interface.

Parameter	Details
INSTALLDIR=*path to ACL for Windows application files folder*	Specifies the destination folder for application files. If you omit the parameter, the default location is used: ◦ **64-bit operating systems** - `C:\Program Files (x86)\ACL Software\ACL for Windows 15\` ◦ **32-bit operating systems** - `C:\Program Files\ACL Software\ACL for Windows 15\`
DATADIR=*path to sample data files folder*	Specifies the destination folder for Analytics sample data files, which is also the Analytics working directory. If you omit the parameter, the default location is used: `C:\Users\user_account_name\Documents\ACL Data\Sample Data Files\` **Note** Make sure that end users have read and write permissions to the data files folder you specify.
APPENCODING=*Analytics edition*	Specifies which edition of Analytics is installed. ◦ APPENCODING=NonUnicode specifies that the Non-Unicode edition of Analytics is installed. Omitting the parameter does the same thing. ◦ APPENCODING=Unicode specifies that the Unicode edition of Analytics is installed.

Specifying the language of the user interface

To specify the language of the ACL for Windows user interface:

- use the /L parameter with `setup.exe`
- use the TRANSFORMS parameter with `ACL for Windows.msi`

You need to specify the appropriate language ID with each parameter.

For example:

- Use /L1033 to specify the English user interface if you are using `setup.exe`
- Use /L1034 to specify the Spanish user interface if you are using `setup.exe`
- Use TRANSFORMS=1034.mst to specify the Spanish user interface if you are using `ACL for Windows.msi`

Note

It is recommended that you always use the language parameter even if you are installing the English version of ACL for Windows on an English operating system.

Omitting the language parameter

If you omit the language parameter, the installation defaults to the language of the computer's operating system, or to English, depending on which installer file you use.

Behavior of language parameters and installer files

Parameter used	`setup.exe`	`ACL for Windows.msi`
/L	ACL for Windows uses the language specified	n/a
TRANSFORMS	n/a	ACL for Windows uses the language specified
parameter omitted	ACL for Windows uses the operating system language or English, if ACL for Windows does not support the operating system language	ACL for Windows uses English

Silent installation syntax for the languages supported by ACL for Windows

Language	`setup.exe` syntax	`ACL for Windows.msi` syntax
Chinese	/L2052	TRANSFORMS=2052.mst
English	/L1033	TRANSFORMS=1033.mst
French	/L1036	TRANSFORMS=1036.mst
German	/L1031	TRANSFORMS=1031.mst
Japanese	/L1041	TRANSFORMS=1041.mst
Portuguese	/L1046	TRANSFORMS=1046.mst
Spanish	/L1034	TRANSFORMS=1034.mst

Specifying the non-Unicode or Unicode edition

> **Note**
>
> - You cannot use the installer to upgrade non-Unicode Analytics to Unicode Analytics, or vice versa.
> - Unicode and non-Unicode editions of Analytics or ACL Desktop cannot be installed side by side.

To specify which edition of Analytics is installed, use the APPENCODING parameter with either `setup.exe` or `ACL for Windows.msi`. The parameter is optional if you are installing the non-Unicode edition.

- Use APPENCODING=NonUnicode, or do not use the parameter, to install the non-Unicode edition of Analytics.
- Use APPENCODING=Unicode to install the Unicode edition of Analytics.

> **Note**
>
> The Chinese and Japanese versions of Analytics are Unicode-only.

Syntax examples

If you are installing ACL for Windows on computers running 32-bit operating systems, and you are specifying an installation directory rather than using the default location, substitute `C:\Program Files\` for `C:\Program Files (x86)\` in the examples below.

Non-Unicode with default settings using setup.exe

The following example installs the English non-Unicode edition and required software prerequisites:

```
\\nas-server-2\shared_files\installers\ACLforWindows15\setup.exe /s /L1033
/v"/qn RUNFROMSETUPEXE=1"
```

Unicode with default settings using setup.exe

The following example installs the Unicode edition in the operating system language with required software prerequisites:

```
"\\nas-server-2\shared_files\installers\ACLforWindows15\setup.exe" /s /v"/qn
APPENCODING=Unicode RUNFROMSETUPEXE=1"
```

Non-Unicode with two custom settings using setup.exe

The following example installs the English non-Unicode edition with required software prerequisites:

```
\\nas-server-2\shared_files\installers\ACLforWindows15\setup.exe /s /L1033
/v"/qn APPENCODING=NonUnicode INSTALLDIR=\"C:\Program Files (x86)\Analytics\"
DATADIR=\"C:\Analytics Data\" RUNFROMSETUPEXE=1"
```

Unicode with default settings using setup.exe

The following example installs the German Unicode edition with required software prerequisites:

```
\\nas-server-2\shared_files\installers\ACLforWindows15\setup.exe /s /L1031
/v"/qn RUNFROMSETUPEXE=1"
```

Non-Unicode with one custom setting using ACL for Windows.msi

The following example installs the English non-Unicode edition (required software prerequisites must already be installed):

```
msiexec /i "\\nas-server-2\shared_
files\installers\ACLforWindows15\ACL for Windows.msi"
INSTALLDIR="C:\Program Files (x86)\Analytics" TRANSFORMS=1033.mst APPENCODING-
G=Unicode /qn
```

Non-Unicode with default settings using ACL for Windows.msi

The following example installs the Chinese Unicode edition (required software prerequisites must already be installed):

```
msiexec /i "\\nas-server-2\shared_files\installers\ACLforWindows15\ACL for Win-
dows.msi"
TRANSFORMS=2052.mst APPENCODING=Unicode /qn
```

Uninstall ACL for Windows

When you uninstall ACL for Windows, you uninstall all components:

- Analytics
- The Analysis App window
- Offline Projects

All ACL for Windows application files are removed from your computer, but Analytics project files, data files, logs, and any project-specific preferences files remain in the Analytics project folders.

> **Note**
> You must be logged in as a Windows user with Administrator rights to uninstall the application.

1. In the Windows Control Panel, open **Programs and Features**.
2. Select **ACL for Windows** and click **Uninstall**.
3. Click **Yes** in the confirmation dialog box.
4. If a security warning dialog box appears, verify the information listed, and click **Yes**.
5. Click **OK** if either or both of these prompts appear:
   - **The setup must update files or services that cannot be updated while the system is running.**
   - **The following applications should be closed before continuing the installation.**

   ACL for Windows is uninstalled.
6. Restart your computer to complete the uninstallation.

Configuring Python for use with Analytics

> **Note**
> These configuration instructions refer to a customer-installed instance of Python required to use Analytics Python functions. This instance of Python is not the same as the instance of Python that can be installed as part of the Analytics installation to support machine learning commands.

To configure Python to work with Analytics, you must install a compatible version of Python and add the Python executable to your computer's PATH environment variable. You must also set the ACLPYTHONDLL and PYTHONPATH system environment variables.

How it works

To run Python scripts, Analytics must be able to call the Python executable and find the scripts it is instructed to run. Analytics uses the PATH environment variable to locate Python and the PYTHONPATH environment variable to locate scripts.

Install Python (32-bit)

1. From the Python downloads page, download one of the following versions of Python to your computer or the server:
   - 3.4.x
   - 3.5.x
   - 3.6.x

   > **Note**
   >
   > The listed versions of Python have been tested and verified to work with Analytics or the Robots Agent.
   >
   > Any version of Python from 3.4.x onward should work. However, versions other than those listed are not guaranteed to work.

2. On your computer or the server, double-click the installer.
3. In the installer, select **Add Python *versionNumber* to PATH**.
4. Click **Install** and follow the on-screen instructions.
5. Reboot the computer or the server before running any Python scripts called by an Analytics script.

Set the ACLPYTHONDLL and PYTHONPATH environment variables

1. In the `C:\` drive of the operating system, create one or more folders to house your Python scripts.

   Example - `C:\python_scripts`

2. From the operating system, open the **System Properties** dialog box and click **Environment Variables**.

3. In the **System variables** section, click **New** and enter the following variables:

Variable name	Variable value
PYTHONPATH	The full path to the folder(s) you created to house the Python scripts. Separate multiple folder paths with a semi-colon. Example: `C:\python_scripts;C:\dev;C:\tmp`
ACLPYTHONDLL	The full path and filename of the Python DLL file in the Python installation folder that you want to use with Analytics or the Robots Agent. Example: `c:\python_install\python35.dll` **Note** Python adds the DLL to the system folder (`c:\windows-\system32\python35.dll`) rather than the installation folder. You must copy the DLL from the system folder to the installation folder, and use the installation folder location as the variable value so that Analytics can access the DLL. You may also need to remove any read-only settings from the installation folder.

4. To save the variable, click **OK** and then in the **System Properties** dialog box, click **OK**.

Use Python in Analytics Python functions

From Analytics, use the Analytics Python functions to call functions in scripts that exist in your PYTHONPATH.

For more information, see "Python" on page 2031.

> **Note**
> If you make any edits to a Python script, you must refresh the view in your Analytics project to use the latest version of the Python script. The simplest way to refresh the view is to close the table you are working with and then re-open it.

Troubleshooting installation and activation

If you are unable to install ACL for Windows, or activate the software once you have installed it, the troubleshooting information in this section may provide a solution.

> **Note**
> Please review the information here before contacting Support.

Installation troubleshooting

There are a variety of issues that may prevent installation of ACL for Windows. The issues typically fall into one of three categories:

- **Administrator rights** – You do not have Administrator rights to your computer, which prevents the installation of the software.
- **IT approval** – Your company prevents the installation of new software, or new versions of existing software, until the software has been approved by your IT department.
- **Technical requirements** – Your computer does not meet the software or hardware requirements necessary to install ACL for Windows.

  For more information, see "ACL for Windows system requirements" on page 2611.

In any of the situations above, you will probably need to work with your IT department to resolve the problem.

Activation troubleshooting

There are a variety of issues that may prevent activation of ACL for Windows. You cannot use the software without activating it. The issues typically fall into one of two categories:

- **Connecting or signing in** – You cannot connect to or sign in to Launchpad.

  Activating ACL for Windows requires that you sign in to Launchpad (www.highbond.com), Galvanize's cloud-based portal for managing your access to Galvanize products and services.
- **Licensing** – You can sign in to Launchpad but you cannot acquire an ACL for Windows license.

> **Note**
> ACL for Windows creates an activation log that may be helpful when troubleshooting activation issues:
> `C:\Users\user_account_name\AppData\Local\ACL\activation.log`
> For more information, see "ACL for Windows activation log" on the next page.

The following table contains troubleshooting information related to activation.

Problem	Possible solutions
The **ACL for Windows** sign-in dialog box does not contain the sign-in fields and displays an error message. OR When you enter your sign-in credentials in the **ACL for Windows** sign-in dialog box, no error message is displayed and you cannot activate the software, or the message "activation error occurred" is displayed.	◦ Make sure that your computer is connected to the Internet. ◦ If your computer is behind a firewall, or connects to the Internet through a proxy server, your IT department may need to add the Analytics (ACLWin.exe) and Internet Explorer executables to the list of applications allowed to access the Internet. Analytics needs to access Launchpad (*.highbond.com) using port 443. Note that Analytics is only compatible with proxy servers and firewalls that provide silent authentication. If your company's proxy server or firewall prompts you to authenticate outgoing connections, Analytics activation may fail. ◦ If your proxy server allows session connections, try opening a browser and connecting to www.highbond.com before starting the activation process. **Note** For more information, see "Connecting to HighBond over a proxy server" on page 2609.
Content in the **ACL for Windows** sign-in dialog box does not display correctly (for example, the content is too large for the dialog box), or a pop-up window indicates a script error.	This issue can occur if the level of Internet Explorer security settings is increased. To avoid this issue, you can add https://*.highbond.com to your list of trusted sites in the Internet Explorer settings. Alternatively, you can restore the default security settings in Internet Explorer.
When you enter your sign-in credentials in the **ACL for Windows** sign-in dialog box, one of the following messages is displayed: ◦ "Invalid credentials" ◦ "You do not belong to an organization" ◦ "Your subscription has expired"	a. Open a supported browser on your computer (IE 9+, Chrome, Firefox, or Safari) and navigate to Launchpad (www.highbond.com). b. Sign in using your Launchpad credentials. If signing in fails, it could be for one of the following reasons: • Your credentials are invalid. Confirm that you are using the correct e-mail domain. You should be using your company's e-mail domain (for example, *my_company.com*), not a domain such as gmail.com or hotmail.com. If you have forgotten your password, use the **Reset password** option. • You do not have a Launchpad account. Contact your company's Analytics account administrator and have that person create your user account, or contact Support. c. If you are able to sign in to Launchpad, click **Options > Organization** at the

Problem	Possible solutions
	top-right corner of the window. • Review the entries under **Ordered Subscription Types** to ensure your company has purchased an ACL for Windows subscription. You should see **ACL for Windows** in this list, with a number indicating how many licenses your company has purchased, and how many are currently assigned. For example: **5/8** If all licenses are currently assigned, your company's Analytics account administrator should sign in to Launchpad to see who the licenses are assigned to, and whether a license can be released for you to use. If you don't see an **ACL for Windows** subscription entry, it means your company has not purchased ACL for Windows, or the order has not yet been processed. For further assistance, contact your company's Analytics account administrator, your Galvanize account representative, or Support. d. If you continue to have issues, take a screen capture of the entries in one of the following locations in your computer's registry, and contact Support: • If you use 32-bit Windows: HKEY_LOCAL_MACHINE\SOFTWARE\ACL Software • If you use 64-bit Windows: HKEY_LOCAL_MACHINE\SOFTWARE\WoW6432Node\ACL Software

ACL for Windows activation log

The ACL for Windows activation log may help you, or your IT department, troubleshoot an activation issue that is preventing you from using Analytics.

Location

The activation log is located in the `AppData` folder on your computer:

`C:\Users\`*`user_account_name`*`\AppData\Local\ACL\activation.log`

Open the activation log in a text editor and scroll to the bottom of the log to see the most recent information.

Activation log settings

By default, the activation log is set to record only a minimal amount of information. In order to troubleshoot effectively, you may need to set the activation log to record a greater amount of information.

Activation log setting	Location to set log	Details
0	Windows Registry	Turns off all logging Setting persists until manually changed
1	Windows Registry	Minimal logging (default) Setting persists until manually changed
`start aclwin.exe /debugactiv-ation`	Windows command prompt	More detailed logging Setting persists for current Analytics session only
2	Windows Registry	Full logging, including all Analytics network traffic Setting persists until manually changed

Change activation log setting in the Windows Registry

Use the Windows Registry Editor to specify an activation log setting that persists until manually changed.

> **Note**
> You must have Administrator rights on the computer to make this change.

1. Open the Registry Editor.

   There are different methods for opening the Registry Editor. This method should work for your version of Windows:

   In the `C:\windows\system32` sub-folder in Windows Explorer, right-click `regedt32.exe` and select **Run as administrator**.
2. Navigate to: `Computer\HKEY_LOCAL_MACHINE\SOFTWARE\WOW6432Node\ACL Software`
3. Double-click **LogSubscription**.
4. In the **Value data** field, enter an activity log setting (0, 1, or 2) and click **OK**.
5. Exit the Registry Editor.

   The next time you use Analytics, the activation log uses the setting you specified.

Change activation log setting using the Windows command prompt

Use the Windows command prompt to specify more detailed logging for the duration of an Analytics session.

> **Note**
> You must have Administrator rights on the computer to make this change.

1. Open the Windows command prompt as an administrator.

   There are different methods for opening the command prompt as an administrator depending on your version of Windows.

   This method works for all versions of Windows:

   In the `C:\windows\system32` sub-folder in Windows Explorer, right-click `cmd.exe` and select **Run as administrator**.

2. Type the following command syntax and press Enter: `start aclwin.exe /debugactivation`

   Analytics opens and for the duration of the session the activation log uses the setting you specified.

Connecting to HighBond over a proxy server

Analytics and Analytics Exchange require HTTPS connections to HighBond for activation and data import and export. If you connect to the Internet over a proxy server, you must configure the proxy server to allowlist HighBond websites.

Allowing HighBond website connections

Analytics and Analytics Exchange use a number of sites on the HighBond domain. The simplest way to allow these connections through a proxy server is to allowlist by domain name: *.highbond.com.

The connection must be allowed over port 443.

ACL for Windows configurations

ACL for Windows connects to HighBond for two purposes:

- **activation** – connects using Internet Explorer settings
- **sharing data (upload and download)** – may require additional configuration if proxy settings are not transparent, such as proxy authentication that does not use integrated Windows authentication

Analytics Exchange configurations

As server proxy settings are often strict and the ACL Analytics Exchange Service account that runs analytics scripts has limited permissions, proxy access can be challenging for AX Server. To enable connections to HighBond over a proxy server, use one of the following approaches:

- **IP/Host rule** – allow requests that originate from the AX Server IP address to connect to *.highbond.com and do not change the ACL Analytics Exchange Service account permissions
- **Change service account permissions** –
  1. Grant the ACL Analytics Exchange Service account permission to access the proxy server.
  2. Set up the proxy settings in the **Internet Options** for the ACL Analytics Exchange Service account.
  3. Use the remaining configuration outlined in "ACL for Windows configurations" above.

Additional resources

For more information about the defaultProxy element, see the system.Net Element article on the Microsoft Developer Network.

ACL for Windows system requirements

Before proceeding with the installation, ensure that the computer on which ACL for Windows will be installed meets the requirements outlined below.

Software requirements

Requirements that must be confirmed or installed by the user

> **Note**
>
> ACL for Windows has been tested and verified to work with the listed versions of operating systems and third-party software. ACL for Windows may work with versions other than those listed, however there is no guarantee that it will.
>
> Use of ACL for Windows with versions other than those listed is considered "an unsupported installation" and our Support team may not be able to find a resolution if issues arise.

Requirement	Additional information
Windows operating system One of the following operating systems: ◦ Microsoft Windows 10 (64-bit) ◦ Microsoft Windows 8.1 (64-bit)	ACL for Windows is a 32-bit application that runs on the 64-bit versions of Windows. **Note** Windows XP and Windows 7 operating systems are no longer supported for ACL for Windows installation.

Requirement	Additional information
Windows patch For Microsoft Windows 8.1 users: Windows 8.1 Update KB2919355	**Important** Windows 8.1 Update KB2919355 is required by Microsoft .NET Framework 4.6.x, which in turn is required by ACL for Windows 15. If you are using Windows 8.1, and .NET 4.6.x is not installed, and you have not run Update KB2919355, the ACL for Windows installer terminates with an error message during the .NET 4.6.2 prerequisite installation. You need to download and install Update KB2919355 before you can continue with the ACL for Windows installation. Alternatively, you can install Update KB2919355 before you begin the ACL for Windows installation and avoid the error message. **Caution** If you are prompted to restart your computer at any point during the installation process, do so right away. **Do not ignore messages to restart your computer.** If you do not restart your computer when you are prompted, you may cause problems with the installation of .NET, other prerequisites, or ACL for Windows.
R, 64-bit (optional) To use Analytics functions that integrate with the R programming language, you must install and configure R. The following versions of R have been tested and work with Analytics: ◦ 4.0.3 ◦ 3.4.4 ◦ 3.3.2 ◦ 3.3.1 ◦ 3.2.5 ◦ 3.2.3 You can use either CRAN R or Microsoft R. **Note** Other versions of R may work as well. However, they are not guaranteed to work.	If you are using one of the CRAN R packages, you may need to add the path to the R binary folder to the PATH environment variable on your computer. For example: `C:\Program Files\R\R-<version>\bin\x64` **Note** You do not need to install R if you do not intend to use the Analytics R functions or RCOMMAND.

Requirement	Additional information
Note Two optional features of Analytics require installing two separate instances of Python: • **Analytics Python functions** - require a customer-installed instance of Python • **Analytics machine learning commands** - require an instance of Python that can be installed as part of the Analytics installation Additional details appear below.	
Python, 32-bit (optional) To use Analytics functions that integrate with the Python programming language, you must install and configure: ◦ Python ◦ PYTHONPATH environment variable ◦ ACLPYTHONDLL environment variable The following versions of Python have been tested and work with Analytics: ◦ 3.4 ◦ 3.5 ◦ 3.6 **Note** Any version of Python from 3.4.x onward should work with Analytics. However, versions other than those listed are not guaranteed to work.	When installing Python, you must also configure it to run on your computer. For more information, see "Configuring Python for use with Analytics" on page 2601. **Note** You do not need to install Python if you do not intend to use the Analytics Python functions.
Python, 32-bit (optional) To enable Analytics machine learning commands, you must install: ◦ Python engine for Analytics (3.7.9)	If you select the **Enable machine learning (install Python engine for Analytics)** check box during ACL for Windows installation, the installer installs the Python engine for Analytics. **Note** This instance of Python is not intended for use with the Analytics Python functions, or for general Python use. You must install a separate instance of Python for these purposes.
Oracle Instant Client (optional) To use the ACL Connector for Oracle, you must install: ◦ Oracle Instant Client 11g or 12c	◦ You do not need to install Oracle Instant Client if you do not intend to use the ACL Connector for Oracle. ◦ The bitness of Oracle Instant Client must match the operating system's bitness. If the 32-bit Instant Client is installed on a 64-bit machine, the connection fails. ◦ If you are using the connector with Analytics Exchange and you install the Oracle Instant Client after AX Server, you must restart the

Requirement	Additional information
	ACL Analytics Exchange Service before you can use the connector.

Automatically installed prerequisites

If the following software prerequisites are not already installed on your computer, they are automatically installed by the ACL for Windows installer:

- Microsoft .NET Framework 4.6.2

> **Note**
> If your computer already has .NET 4.6.0 or 4.6.1, ACL for Windows uses the installed version of .NET and does not install version 4.6.2.

- Microsoft Visual C++ 2015-2019 Redistributable Package (x64 and x86)
- Microsoft Visual C++ 2013 Redistributable Package (x64) (only if optional data connectors are installed)
- Microsoft Visual C++ 2012 Redistributable Package (x64) (only if optional data connectors are installed)
- Microsoft Access Database Engine 2016 (32-bit)

Requirements contained in the Analytics installation directory

Local copies of the following components are installed with Analytics and contained in the Analytics installation directory:

- Amazon Corretto 8 (OpenJDK Platform binary 8.275.01.1) – used by the Visualizer in the Analysis App window
- Python Engine 3.7.9 and TPOT 0.10.2 – used by the Analytics machine learning commands

Requirements installed as part of the supported operating system

The following components are also Analytics prerequisites, but they are installed as part of the supported operating systems:

- Microsoft Data Access Components (MDAC) 2.8
- Microsoft Jet 4.0 Database Engine (MSJet)
- Microsoft XML Core Services (MSXML) 6.0
- Internet Explorer 9 or higher (required for XML-based formatting of command results that are output to screen)

Analytics data connectors

The drivers listed below are installed for use as Analytics data connectors.

Standard data connectors

The following data connectors are installed by default when installing ACL for Windows.

- ACL Connector for Active Directory
- ACL Connector for Amazon Athena
- ACL Connector for Amazon S3
- ACL Connector for AWS Data Management
- ACL Connector for Azure Data Management
- ACL Connector for Azure Table Storage
- ACL Connector for Box
- ACL Connector for DocuSign
- ACL Connector for Dynamics 365 Business Central
- ACL Connector for Dynamics 365 Finance and Operations
- ACL Connector for Dynamics 365 Sales
- ACL Connector for Dynamics CRM
- ACL Connector for Dynamics GP
- ACL Connector for Dynamics NAV
- ACL Connector for Edgar Online
- ACL Connector for Email
- ACL Connector for Epicor ERP
- ACL Connector for Exact Online
- ACL Connector for Exchange
- ACL Connector for Jira
- ACL Connector for JSON
- ACL Connector for LDAP
- ACL Connector for LinkedIn
- ACL Connector for Marketo
- ACL Connector for MySQL
- ACL Connector for NetSuite
- ACL Connector for OData
- ACL Connector for Open Exchange Rates
- ACL Connector for Oracle Eloqua
- ACL Connector for Oracle Sales Cloud
- ACL Connector for Presto
- ACL Connector for Qualys
- ACL Connector for QuickBooks
- ACL Connector for QuickBooks Online
- ACL Connector for QuickBooks POS
- ACL Connector for REST
- ACL Connector for Rsam
- ACL Connector for RSS/ATOM
- ACL Connector for Sage 50 UK
- ACL Connector for Sage Cloud Accounting
- ACL Connector for Sage Intacct

- ACL Connector for SAP
- ACL Connector for SAP ByDesign
- ACL Connector for SAP Hybris Cloud for Customer
- ACL Connector for SAP SuccessFactors
- ACL Connector for ServiceNow
- ACL Connector for SFTP
- ACL Connector for SharePoint
- ACL Connector for Slack
- ACL Connector for Snowflake
- ACL Connector for Splunk
- ACL Connector for Square
- ACL Connector for Stripe
- ACL Connector for SugarCRM
- ACL Connector for SurveyMonkey
- ACL Connector for Sybase
- ACL Connector for Sybase IQ
- ACL Connector for Tenable.sc
- ACL Connector for Twitter
- ACL Connector for UPS
- ACL Connector for USPS
- ACL Connector for xBase
- ACL Connector for Zendesk

Optional data connectors

The following data connectors are optional, and you can opt to not install these when installing ACL for Windows.

- ACL Connector for Amazon Redshift
- ACL Connector for Cassandra
- ACL Connector for Concur
- ACL Connector for Couchbase
- ACL Connector for Drill
- ACL Connector for DynamoDB
- ACL Connector for Google BigQuery
- ACL Connector for HBase
- ACL Connector for Hive
- ACL Connector for Impala
- ACL Connector for MongoDB
- ACL Connector for Oracle
- ACL Connector for Salesforce
- ACL Connector for Spark
- ACL Connector for SQL Server
- ACL Connector for Teradata

Using Tableau with the ACL Connector for Analytics

You can use the ACL Connector for Analytics to extract data from Analytics projects into Tableau.

To optimize the integration between Tableau and Analytics, perform the following steps:

1. Install ACL for Windows.
2. Copy `ACL Connector for Analytics.tdc` from:

   `C:\Program Files (x86)\ACL Software\ACL for Windows 15\ACL ODBC`

   to:

   `..\Documents\My Tableau Repository\Datasources`

   > **Note**
   > These file paths are default locations. Your installations of Analytics and Tableau may use different locations.

3. Restart Tableau.

Hardware requirements

> **Note**
> For best performance of Analytics in a production environment, you may need to provide resources greater than the minimum specifications listed.

Component	Minimum	Recommendation
Processor	1.8 GHz	
Memory (RAM)	2 GB	◦ 64-bit operating systems: 8 GB or more, especially if sorting large files ◦ 32-bit operating systems: 4 GB, especially if sorting large files
Hard disk space (Analytics application files)	1.1 GB	
Hard disk space (software prerequisites)	8 GB	
Hard disk space (data storage)		100 GB or more In addition to the hard disk space required to install Analytics application files and prerequisites, significant additional space is required if a computer will be used to store data extracts, flat files, and results.

Connectivity

TCP/IP connectivity is required for the following purposes:

- post-installation, for activating ACL for Windows
- for accessing context-sensitive online help
- periodically required for ongoing software subscription validation
- periodically required for automatic software updates

Connection requirements

> **Note**
>
> If your company uses a proxy server, a firewall, or any other network security measure that prevents ACL for Windows from connecting to the Internet, see "Connecting to HighBond over a proxy server" on page 2609, and "Troubleshooting installation and activation" on page 2604.

ACL for Windows requires an Internet connection in order to perform the following functions:

- Activate the software before first use
- Validate the software subscription on an ongoing basis
- Share data with HighBond
- Provide application-level communication between Analytics software components
- Send notification of software updates
- Access context-sensitive online help

The specific connections required by the various executable files within ACL for Windows are summarized as follows.

Application name (executable file)	Required connection	Reason(s) for the connection
Analytics (ACLWin.exe)	https://*.highbond.com, port 443	◦ Initial activation of Analytics ◦ Ongoing software subscription validation ◦ Sharing data with HighBond
The Analysis App window (ACLscript.exe)	https://*.highbond.com, port 443	◦ Ongoing software subscription validation (if ACLWin.exe is not used) ◦ Sharing data with HighBond
The Analysis App window (aclx.exe)	https://*.highbond.com, port 443 localhost (dynamic ports)	◦ Application-level communication between Analytics software components ◦ Sharing data with HighBond
Software update notification	https://*.highbond.com, port 443	◦ Automatic notification of software updates
(ACL-service.exe)	localhost (dynamic ports)	◦ Application-level communication between Analytics software components
Internet Explorer (lexplore.exe)	https://*.highbond.com, port 443	◦ Initial activation of Analytics

This page intentionally left blank

Automating and sharing

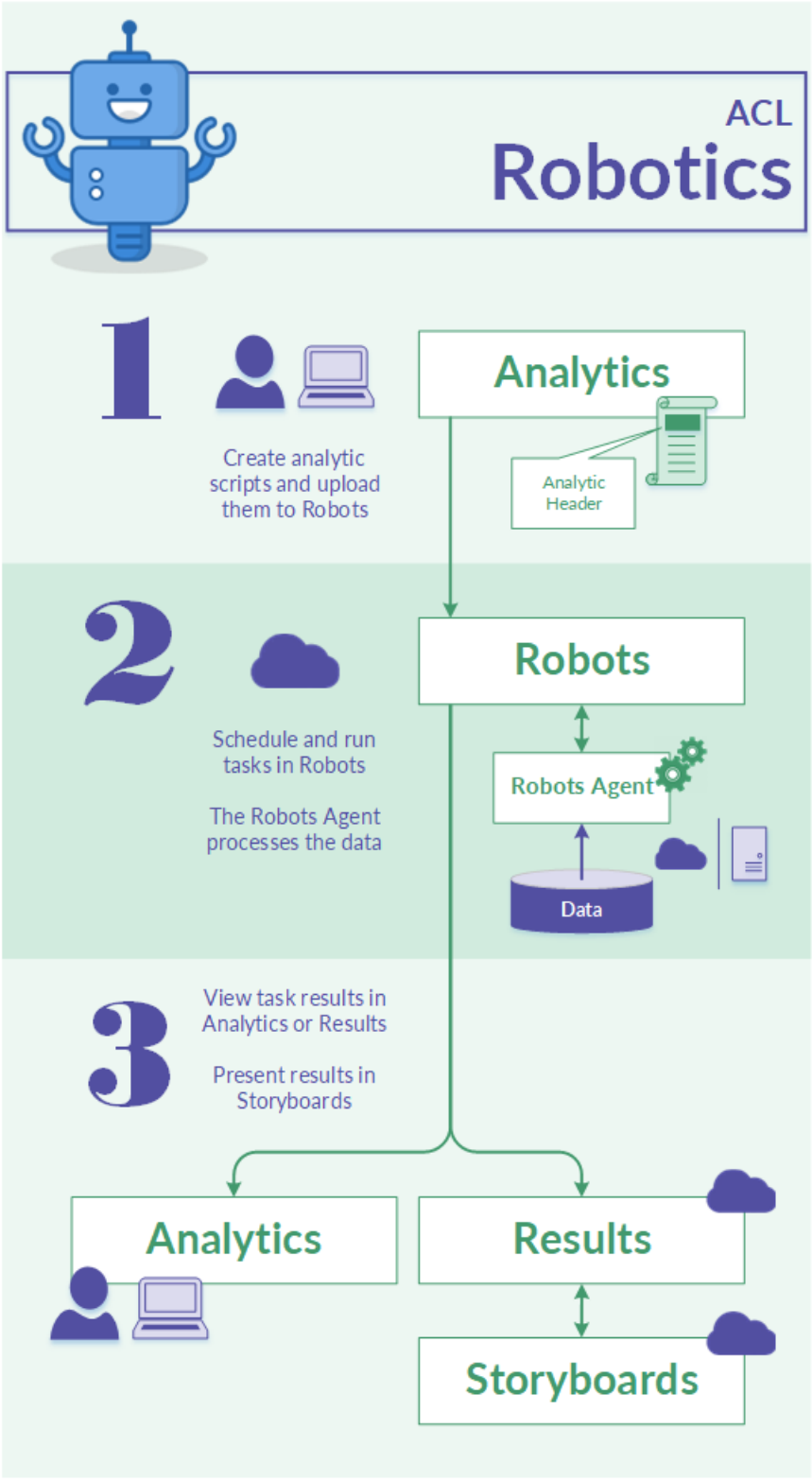
Automating and sharing

You can increase the value of your data analysis work in Analytics by making it available to others as part of a broader workflow. Several options exist for sharing and extending Analytics projects, and the work they contain.

> **Note**
> With the exception of the Analysis App window, the features or products listed below are separate pieces of software that integrate with Analytics.

Publishing data to Results	Results is a HighBond app that you can upload records to for additional processing and issue remediation. It contains workflow automation tools such as triggers, questionnaires, and metrics. You can also visualize data.
"Publishing data to Storyboards" on page 2625	Storyboards is a HighBond app and communication platform that displays multiple visualizations and rich text content in a single presentation, which can easily be shared with executives or other stakeholders.
Automating with Robots	Robots is a HighBond app that you can use to automate repetitive tasks using analytic scripts built in Analytics. Robots handles the recurring tasks according to your configuration.
"Working with analysis apps" on page 2642	The Analysis App window is a freestanding component of Analytics that provides a simple, modern user interface for running scripts configured as "analytics". The interface guides users as they provide input values for the scripts. The Analysis App window also allows users to create data interpretations and visualizations based on the output results of the scripts, or based on other tables in an analysis app, or in an Analytics project.
AX (Analytics Exchange)	Analytics Exchange is a server-based product that lets users schedule and run analytic scripts in an automated fashion. The full processing power of one or more servers can be harnessed for large-scale data analysis. **Note** Galvanize will end support for Analytics Exchange on January 1, 2023. Learn more or upgrade to Robots.

ACL Robotics infographic


ACL
Robotics
1
Create analytic scripts and upload them to Robots
Analytics
Analytic Header
2
Schedule and run tasks in Robots
The Robots Agent processes the data
Robots
Robots Agent
Data
3
View task results in Analytics or Results
Present results in Storyboards
Analytics
Results
Storyboards

Publishing data to Results

Results is part of HighBond. It is a remediation and workflow automation app that manages exception data, adds human context through questionnaires, and makes your monitoring continuous with triggers and metrics.

Results is an ideal destination for your work in Analytics and ACLScript, as you can use it to build visualizations and automate activity.

Features and functionality

- **Collaborate with others** – Invite a team to help investigate, remediate, and track issues identified by Data Analytics
- **Automate workflows** – Create triggers to automate your company's remediation process
- **Integrate questionnaire responses into data analytics** – Send questionnaires to gather additional evidence and analyze responses
- **Visualize data** – Build visualizations and interpretations to communicate your data insights and present them in storyboards
- **Provide oversight** – Combine preventative controls with detective controls for complete oversight over processes or programs
- **Store data** – Use reference tables to bring data from other sources into HighBond and reference it in the platform

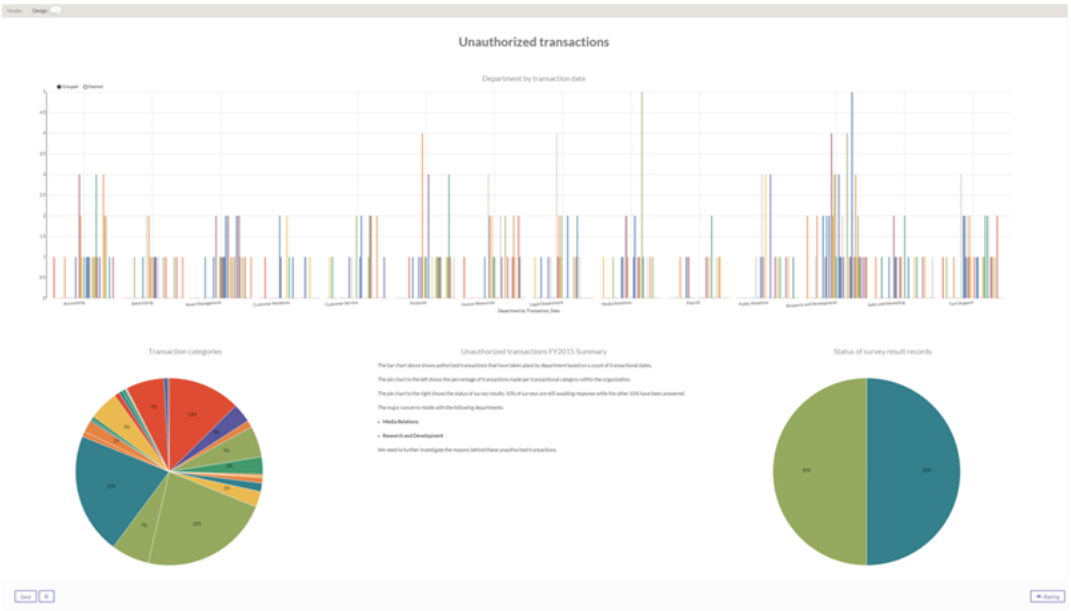
Getting started with Results

All Analytics users have access to HighBond and, by extension, Results.

- Exporting exceptions to Results
- Scripting exports to Results with EXPORT
- Logging into HighBond
- General help using Results

Publishing data to Storyboards

Storyboards is an app in HighBond. A storyboard is a communication platform that displays multiple visualizations and rich text content in a single presentation, which can easily be shared with executives or other stakeholders.



How storyboards work

Storyboards display visualizations and rich text content in rows and columns. Once you populate your data in Results, you can use it to create visualizations and metrics. You then include those visualizations and metrics on your storyboard.

- Logging into HighBond
- General help using Results
- General help using Storyboards

Automating with Robots

Robots is a HighBond app that you use to automate repetitive tasks using scripts built in Analytics. Once you create the scripts, you upload them to Robots, where you configure the task automation that you need. Robots handles the recurring tasks according to your configuration.

For detailed information about the Robots app, see Automating work with Robots in HighBond Help. For help logging into HighBond, see Accessing your account.

How do I automate using Robots?

To automate repetitive tasks using Robots, you must first create a project in Analytics that contains at least one **analytic script**. An analytic script is a regular Analytics script that uses an analytic header to declare certain properties and instructions for running the script.

For more information about analytic scripts and analytic headers, see "Analytic scripts overview" on page 2460.

Committing scripts and creating a robot

Once you have written the analytic script or scripts, you upload them to the Robots app. The action of uploading scripts from Analytics to Robots is called **committing scripts**. Committing scripts for the first time causes **a robot** to be created in the Robots app. A robot is a container that houses committed analytic scripts, any auxiliary scripts, and related files. The robot is the object that you configure to carry out scheduled, automated tasks.

Development mode versus production

Scripts are committed to Development mode only in Robots, never to production. This design protects the production scripts, which are kept completely separate from Development mode.

Once you have committed the final version of a script or scripts to Development mode, you must explicitly activate the version into production.

An alternate approach

In the Robots app, you can create an empty robot and then commit scripts from Analytics to the empty robot. Whether you manually create the robot in the Robots app, or automatically create the robot when you commit scripts for the first time, the result is the same.

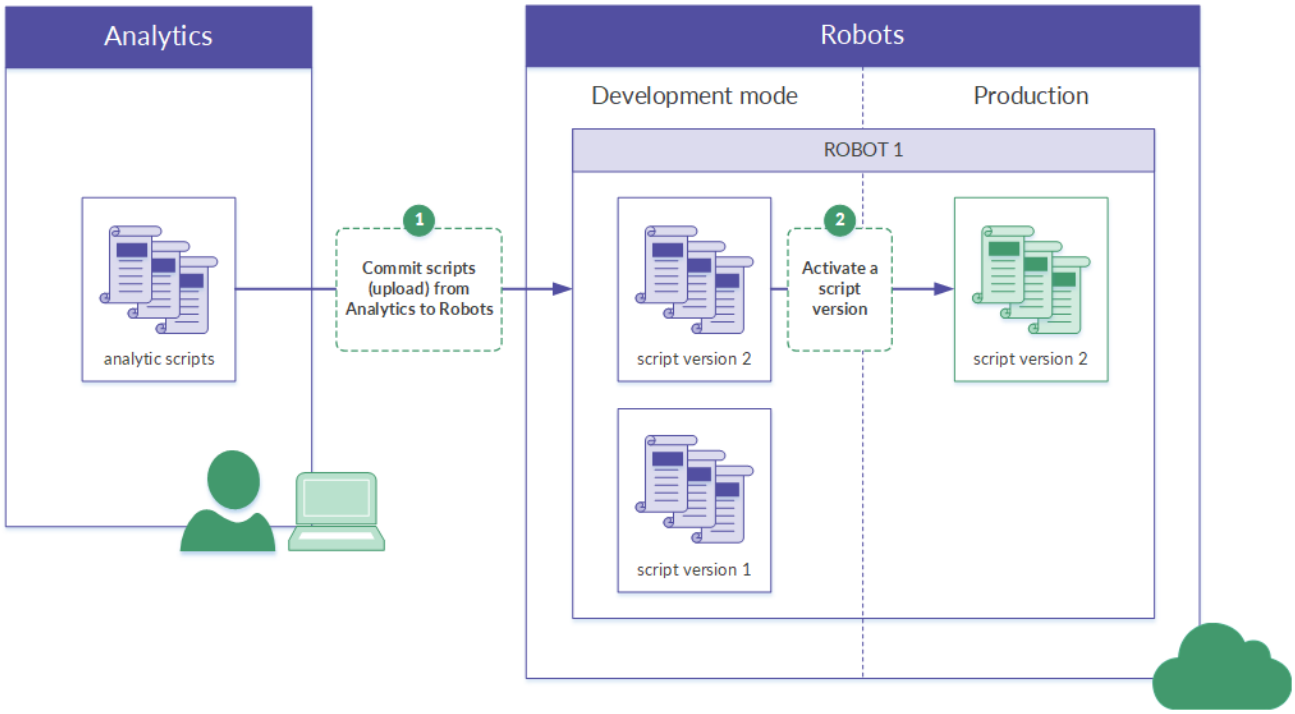
Script development workflow in Analytics and Robots

Analytic script authoring takes place in the Analytics app. You create analytic scripts in Analytics, and as required you update them in Analytics.

When you are ready to upload analytic scripts from Analytics to the Robots app, you **commit** the scripts to a specific robot. For detailed information, see "Committing scripts (uploading) from Analytics to Robots " on page 2633.

When you commit scripts, a new version of the scripts is added to the robot and the version is accessible in development mode. The new version contains the exact content of the scripts that you commit. The version is self contained, and does not merge with any previous version of the scripts. If you removed a script in Analytics, it is no longer available in the new version of the scripts in Robots.

Once you are satisfied that a script version is working correctly in development mode, you can activate it for use in production.



An iterative workflow

You can edit one or more scripts and re-commit them to Robots. Each time you re-commit scripts, you create a new version of the scripts. You can use either of these methods for editing and re-committing scripts:

- Edit an existing script or scripts in an Analytics project associated with a robot and commit the scripts again.
- Download scripts from a robot to Analytics, edit one or more of the scripts, and commit the scripts.

How script versions work

- **Versions are sequentially numbered**

  Each time you commit one or more scripts to the same robot you create a new, sequentially numbered version of the scripts: version 1, version 2, and so on.

  By saving successive versions of scripts, the Robots app ensures that you do not lose any of your scripting work, and allows you to easily access older versions, if necessary.

- **Versions are self-contained**

  Each committed version is completely self-contained. Earlier versions of scripts are never overwritten, and scripts are never merged across versions.

  If you remove a script from a project, all subsequent versions that you commit do not contain the script.

- **Versions contain all scripts**

  A version contains all the scripts that are in a project when you commit the scripts to Robots: all the analytic scripts, and any auxiliary scripts. You cannot selectively commit scripts from a project.

- **Version changes are recorded**

  In the **Script versions** tab in development mode, you can select a script version to see the names and categories of the individual scripts that are in the version. Names of newly added, deleted, or modified scripts are highlighted.

Example of script versions

Scenario

You want to automate a set of analytical tests that your department currently performs manually on the bi-weekly payroll file. The tests check that employees are paid correctly, proper payroll deductions are made, no phantom employees are paid, and so on.

The script development workflow

- In Analytics, you develop a script that imports the payroll file, performs some preliminary data preparation tasks, and then performs all the analytical tests.

- You commit the script to Robots, where you run it in development mode against a copy of the actual data to ensure that it is working correctly.
- Once the script is working correctly, you activate it for use in production and schedule it to run automatically every two weeks.

Analytics	Action/Result	Robots
 **Commit Scripts >>**	◦ **Commit** - You commit script 1, which imports, prepares, and analyzes bi-weekly payroll data. ◦ **Test** - You test script 1 in Robots development mode, and it seems to be working correctly. ◦ **Problem** - You recognize that script 1 contains so much functionality that it might be difficult for someone else to easily understand or edit.	 **>> Version One**
 **Commit Scripts >>**	◦ **Edit** - You divide script 1 into three separate scripts (2, 3, 4): one each for the import, prepare, and analyze phases. You delete script 1. ◦ **Commit** - You commit the scripts in the project. ◦ **Test** - You test the three scripts, and you realize some of the results that should be in the output are not included. ◦ **Problem** - The data cleansing performed by script 3 is not comprehensive enough, and you are losing some results.	 **>> Version Two**
 **Commit Scripts >>**	◦ **Edit** - In script 3, you use Analytics functions to create computed fields that perform additional data cleansing. You make no changes to scripts 2 and 4.	 **>> Version Three**

Analytics	Action/Result	Robots
	◦ **Commit** - You commit the scripts in the project. ◦ **Test** - You test the three scripts, and all the results that should be in the output are now included. ◦ **Success** - You are satisfied that all three scripts, and the overall process, are working correctly. You activate Version Three of the scripts for use in production.	

 edited script  unchanged script

Best practices when editing and committing scripts

Treat the scripts on Robots as the master versions

Scripts are protected from alteration once they are committed to Robots. As a best practice, you should treat the scripts on Robots as the master versions. If you want to edit scripts, you should first download the scripts from Robots rather than using locally saved copies.

You are not limited to downloading the most recent version of the scripts on Robots. You can download whatever version of the scripts you want to work with.

The risk of beginning work with a local copy

The risk of beginning your editing work with a local copy of scripts is that the copy may not match the version on Robots:

- you may have inadvertently altered the local copy, or forgotten that you altered it
- someone else may have committed a version of the scripts to Robots, subsequent to your last commit of the scripts

When is it safe to skip downloading?

After downloading a script version, it is generally safe during a single scriptwriting session to commit iterative versions of the scripts directly from the Analytics project, without downloading between iterations.

If someone else could be working on the scripts at the same time as you, then download between iterations. It is recommended that only one person at a time work on a script version.

Test edited scripts that are part of scheduled tasks

You should always test an edited script or scripts that are part of a scheduled task in Robots to make sure that the edits have not broken the task.

For example, making any of the following edits to an analytic header in a script can break the associated task, and the task must be recreated:

- adding or removing an input parameter
- adding or removing an input file, table, or field
- changing a hard-coded input file, table, or field name

Syntax validation

Every time you commit scripts to Robots, Analytics automatically performs two types of syntax validation or checking:

- Analytic header validation
- Script syntax checking

If any script does not pass the validation or checking, committing the scripts is prevented, and a message appears that identifies the location of the problem.

Analytic header validation

The analytic header in an analytic script must conform to certain requirements. If it does not conform, the analytic script fails when run.

Analytic header validation cannot be disabled.

For more information, see "Working with analytic headers" on page 2473.

Script syntax checking

Certain elements in analytic scripts, such as run-time user interaction commands and absolute file paths, are not supported, or not recommended.

Script syntax checking is enabled by default. If you want to turn it off, select **Disable Script Syntax Check Before Commit Scripts** in the **Options** dialog box (**Tools > Options > Interface**).

For more information, see "Analytic development best practices" on page 2481.

Committing scripts (uploading) from Analytics to Robots

Several possibilities exist when you commit, or upload, scripts to the Robots app.

The easiest way to understand the various possibilities is to think of an Analytics project and a robot as two containers, each holding scripts, that can be associated. Once they are associated, you can commit successive versions of scripts from the project to the robot.

How it works

When you commit scripts to the Robots app, all scripts in the Analytics project are committed and together become "a version" in the associated robot. You cannot selectively commit scripts from a project.

You can use the Robots app to perform a first-time import of analytic scripts to a newly created robot. Subsequently, you must use Analytics to commit or upload scripts.

To commit scripts to Robots, you must be assigned to the appropriate role in the Robots app. For more information, see Robots app permissions.

Action	Result upon commit
"Commit scripts to a new robot" below	A new robot is created that contains version 1 of the committed scripts.
"Commit scripts to an existing robot" on the next page	The existing robot contains the committed scripts, with a version number that depends on whether the robot already contains scripts.
"Commit edited scripts" on page 2635	A new version of the scripts is committed to the robot associated with the project.
"Commit scripts to a different robot" on page 2636	The scripts are committed to either a newly created robot, or an existing robot. The association between the project and the previous robot is deleted.

Commit scripts to a new robot

Commit scripts to the Robots app for the first time to create a new robot that contains the committed scripts.

1. From your computer, open the Analytics desktop application.
2. From the Analytics main menu, select **File > Commit Scripts**.

   If an error message appears, there may be a problem with the analytic header, or the script syntax, in one or more of the scripts in the project.

   For more information, see Script development workflow in Analytics and Robots.
3. If required, in the **Select Destination** dialog box, double-click the appropriate HighBond instance.

   The **Robot Collection** appears with the list of existing robots.
4. Type a robot name in the **New Robot** field, and click **Create**.

   The robot is created, and a robot ID is automatically generated.

   The Analytics project that contains the scripts, and the new robot, are now associated so that subsequent commits do not require that you manually locate the robot.

   > **Note**
   > Do not use the following characters anywhere in the robot name: "$", "€".

5. Enter a short commit message that describes the committed scripts, and click **OK**.

   Version 1 of the scripts is committed to the newly created robot. The scripts exist in Development mode only at this point.
6. Optional. In the **Commit Scripts Successful** dialog box, click either of the links to inspect the newly created robot or the committed scripts.
7. Click **OK** to exit the dialog box.

Commit scripts to an existing robot

Commit scripts to a robot that already exists to populate the robot. The existing robot can already contain scripts, or it can be empty.

> **Note**
> Use this method if the project is not yet associated with a robot. If the project is already associated with a robot, see "Commit scripts to a different robot" on page 2636.

1. From your computer, open the Analytics desktop application.
2. From the Analytics main menu, select **File > Commit Scripts**.

   If an error message appears, there may be a problem with the analytic header, or the script syntax, in one or more of the scripts in the project.

   For more information, see Script development workflow in Analytics and Robots.
3. If required, in the **Select Destination** dialog box, double-click the appropriate HighBond instance.

The **Robot Collection** appears with the list of existing robots.

4. In the list of robots, select the robot that you want to commit the scripts to and click **OK**.

   The Analytics project that contains the scripts, and the existing robot, are now associated so that subsequent commits do not require that you manually locate the robot.

5. Enter a short commit message that describes the committed scripts, and click **OK**.

   The scripts are committed to the existing robot. The version number of the scripts depends on whether the robot already contains scripts, or whether it was previously empty.

   The scripts exist in Development mode only at this point.

6. Optional. In the **Commit Scripts Successful** dialog box, click either of the links to inspect the existing robot or the committed scripts.
7. Click **OK** to exit the dialog box.

Commit edited scripts

Commit edited scripts to create a new version of the scripts in the associated robot.

Two methods exist for editing scripts before committing them. You can edit the scripts in the associated project, or you can download the scripts from Robots to a new Analytics project, and edit the scripts in the new project.

> **Note**
>
> Downloading the scripts from Robots before beginning any editing work is a best practice. You can be sure that the downloaded scripts have not been altered, unlike scripts that are stored locally.
>
> During a single scriptwriting session, if you are the only person working on the scripts, it is generally safe to commit iterative versions of the scripts directly from the project, without downloading between iterations.

Download the scripts from Robots

Perform this portion of the procedure if you want to work with scripts contained in Robots, rather than scripts already on your local computer.

1. Open the Robots app.
2. From the dashboard in Robots, click the robot that contains the scripts you want to download.
3. In the top right corner of the robot, use the **Dev mode** toggle to select the environment to use.
4. In the **Script versions** tab, select the version of the scripts that you want to edit.
5. In the **Version details** panel, click **Download scripts**.

   The version of the scripts that you selected is downloaded to the default Downloads folder on your computer. The scripts are contained in a newly created Analytics project with the same name as the robot you downloaded from. The project and the robot are automatically associated.

> **Note**
> The robot is now associated with two projects: the project that was just created by downloading, and the project previously used to commit the scripts. One robot can be associated with multiple projects.

6. Optional. Move the project containing the downloaded scripts if you want to work with it in another folder.

Edit the scripts in Analytics and commit them

1. From your computer, open the Analytics desktop application.
2. Edit the scripts and save your changes.

   > **Note**
   > You can also add or delete scripts, if required.

3. From the Analytics main menu, select **File > Commit Scripts**.

   If an error message appears, there may be a problem with the analytic header, or the script syntax, in one or more of the scripts in the project.

   For more information, see Script development workflow in Analytics and Robots.
4. Enter a short commit message that describes the change to the committed scripts, and click **OK**.

   The scripts are committed to the associated robot where they are saved as a new version. Existing versions of the scripts in the robot are not overwritten.

   If an error message appears stating that the associated robot is not found, check that the robot exists in Robots, and that your role allows committing scripts.
5. Optional. In the **Commit Scripts Successful** dialog box, click either of the links to inspect the robot, or the new version of the scripts.
6. Click **OK** to exit the dialog box.

Commit scripts to a different robot

Commit scripts to a different robot to add a version of the scripts to the robot, and to change the robot associated with the project. The scripts can be either edited or unedited.

Two possibilities exist when you commit scripts to a different robot:

- commit scripts to a new robot
- commit scripts to an existing robot

1. From your computer, open the Analytics desktop application.
2. From the Analytics main menu, select **File > Commit Scripts As**.

   If an error message appears, there may be a problem with the analytic header, or the script syntax, in one or more of the scripts in the project.

   For more information, see Script development workflow in Analytics and Robots.
3. If required, in the **Select Destination** dialog box, navigate to the appropriate HighBond instance.

   The **Robot Collection** appears with the list of existing robots.
4. Do one of the following:
   - **Commit scripts to a new robot** – Type a robot name in the **New Robot** field, and click **Create**.

     The robot is created, and a robot ID is automatically generated. The Analytics project that contains the scripts, and the new robot, are now associated.

     > **Note**
     > Do not use the following characters anywhere in the robot name: "$", "€".

   - **Commit scripts to an existing robot** – In the list of robots, select the robot that you want to commit the scripts to and click **OK**.

     The Analytics project that contains the scripts, and the existing robot, are now associated.

   The association between the project and the previous robot is deleted.
5. Enter a short commit message that describes the committed scripts, and click **OK**.

   The scripts are committed to the newly created or the existing robot. The version number of the scripts depends on whether the robot already contains scripts, or whether it was previously empty.

   The scripts exist in Development mode only at this point.
6. Optional. In the **Commit Scripts Successful** dialog box, click either of the links to inspect the newly created or the existing robot, or the committed scripts.
7. Click **OK** to exit the dialog box.

Viewing Robots tables, logs, and files

You can download Analytics tables or logs, or non-Analytics files, from Robots and view them on your local computer. Different options exist depending on the table or file type.

Table or file type	Download in result package or Analytics project	Download individually	View in Robots
Analytics result table	✓	✗	✗
Analytics data table	✓	✗	✗
Analytics result log	✓	✓	✓
non-Analytics result file	✓	✓	✗
non-Analytics related file	✗	✓	✗

Viewing a table in Analytics

The contents of a downloaded result package, or Analytics project, and how you view downloaded Analytics tables, depends on the type of Robots Agent you are using.

For more information about the Robots Agent, see HighBond Help.

Agent type	Download and view details
On-premise Robots Agent	◦ **Download** - The downloaded package or project contains only the table layout. The table data remains on your organization's network, but you can use the data for analysis in Analytics. ◦ **View** - Using a server profile, you connect from Analytics to the table data on your organization's Robots Agent. ◦ **Supported tables** - Analytics result tables and Analytics data tables.
Cloud-based Robots Agent	◦ **Download** - The downloaded package or project contains both the table layout and the table data. ◦ **View** - You access the table data locally using only Analytics. ◦ **Supported tables** - Analytics result tables only.

Download task results

Task results can include Analytics result tables, other file types, and a log file.

Task results are specified using the //RESULT analytic tag. For more information, see "RESULT tag" on page 2538.

1. Open the **Robots** app.
2. From the dashboard in Robots, click the robot that contains the results.
3. In the top right corner of the robot, use the **Dev mode** toggle to select the environment to use.
4. Select the **Task runs** tab.
5. Select the task run with the results that you want to download.
6. In the **Task run details** panel, do one of the following:

Download a result file individually	Click **Download** beside the name of a non-Analytics result file, or an Analytics log file. **Result** - The file is downloaded to your local file system and can be opened in its native application. An Analytics log file can be opened in any text editor. You can also click **View** beside the name of a log file to view the log directly.
Download all result files in a package	Click **Download result package**. **Result** - A compressed file named `<robot_name>.zip`, which includes an Analytics project, is downloaded to your local file system. You can extract the contents of the compressed file, open the Analytics project in Analytics, and view the Analytics result tables. You can open any other result file types, such as Excel, in their native applications. If the task run failed, click **Download failed package** to download the log file. You can also click **View** beside the name of the log file to view the log directly. The log file can help you identify the reason the task failed.

Download an Analytics data table

> **Note**
> Analytics data tables are not supported by the cloud-based Robots Agent.

Data tables are specified using the //DATA analytic tag. For more information, see "DATA tag" on page 2545.

1. Open the **Robots** app.
2. From the dashboard in Robots, click the robot that contains the data table or tables.
3. In the top right corner of the robot, use the **Dev mode** toggle to select the environment to use.
4. Select the **Input/Output** tab.
5. In the **Other tables** list or the **Source tables** list, select the data table that you want to download.

6. In the **Table details** panel, click **View Table in AN**.

> **Tip**
> If you want to remove the table, click **Delete table > Delete**.

Result - An Analytics project named `<robot_name>.acl` is downloaded to your local file system. The project contains all the data tables from the **Input/Output** tab, not just the table you selected.

You can open the project in Analytics and view the data table.

Open a downloaded Analytics table (on-premise Robots Agent)

Before you start

To connect to data on a Robots Agent from Analytics the following prerequisites must be in place:

- Analytics users must have the appropriate Windows logon rights and folder permissions on the server where the Robots Agent is installed.

  For more information, see On-premise Robots Agent security.
- the **Enable Server integration** option in Analytics must be selected (**Tools > Options > Interface**)
- the **RobotsProfile** server profile must be configured in Analytics
- the edition of Analytics (Unicode or non-Unicode) must match the edition of the Robots Agent

Open the table

1. On your computer, navigate to the zipped or the downloaded Analytics project (`*.zip` or `*.acl`).
2. If the project is zipped, right-click it and choose an appropriate option to unzip it.
3. Double-click the downloaded or the unzipped project.

   The project opens in Analytics.
4. Double-click a Robots table to open it.
5. Enter your network password and click **OK**.

Troubleshooting

If you get a connection error when attempting to open a Robots table in Analytics, try these remedies:

- **Robots Data Service** - Check that the **Robots Data Service** is running on the server where the Robots Agent is installed. Start the service if it is stopped.

- **Robots server profile** – Delete the **RobotsProfile** server profile and close and reopen the Analytics project to automatically create a refreshed profile.

  To delete the **RobotsProfile**, on the Analytics main menu, go to **Server > Server Profiles**.

Open a downloaded Analytics table (cloud-based Robots Agent)

1. On your computer, navigate to the zipped Analytics project (`*.zip`).
2. Right-click the project and choose an appropriate option to unzip it.
3. Double-click the unzipped project.

   The project opens in Analytics.
4. Double-click a table to open it.

Working with analysis apps

Analysis apps are bundled sets of analytics that import and prepare source data, and then perform analysis. You run analysis apps in the Analysis App window, a freestanding component of Analytics.

Analytics are regular Analytics scripts with additional annotations that allow the scripts to run in the Analysis App window. If an Analytics project (.ACL) contains at least one analytic, it can be packaged and used as an analysis app. The packaging of analysis apps makes them easily portable and sharable.

For information about developing analytics, including converting scripts to analytics, and packaging analysis apps, see "Analytic scripts overview" on page 2460.

Extend the reach of Analytics

Users without any scriptwriting knowledge can use the Analysis App window to run analytics created by Analytics script writers in Analytics. Analytics script writers can design analytics that control user input and guide users through the process of running an analytic. This ability to share Analytics scripts and analytics in a controlled and easy-to-use manner extends the reach of Analytics to a broad range of users without requiring them to learn anything about scripting in Analytics, or even to open Analytics.

Data interpretations and visualizations

The Analysis App window includes a data interpretation and visualization capability. You can filter, sort, and highlight data, and create various charts based on the data. You can use interpretations and visualizations with any of following data:

- Results tables produced by analytics run in the Analysis App window
- Any data tables included with an analysis app
- Any tables in an Analytics project

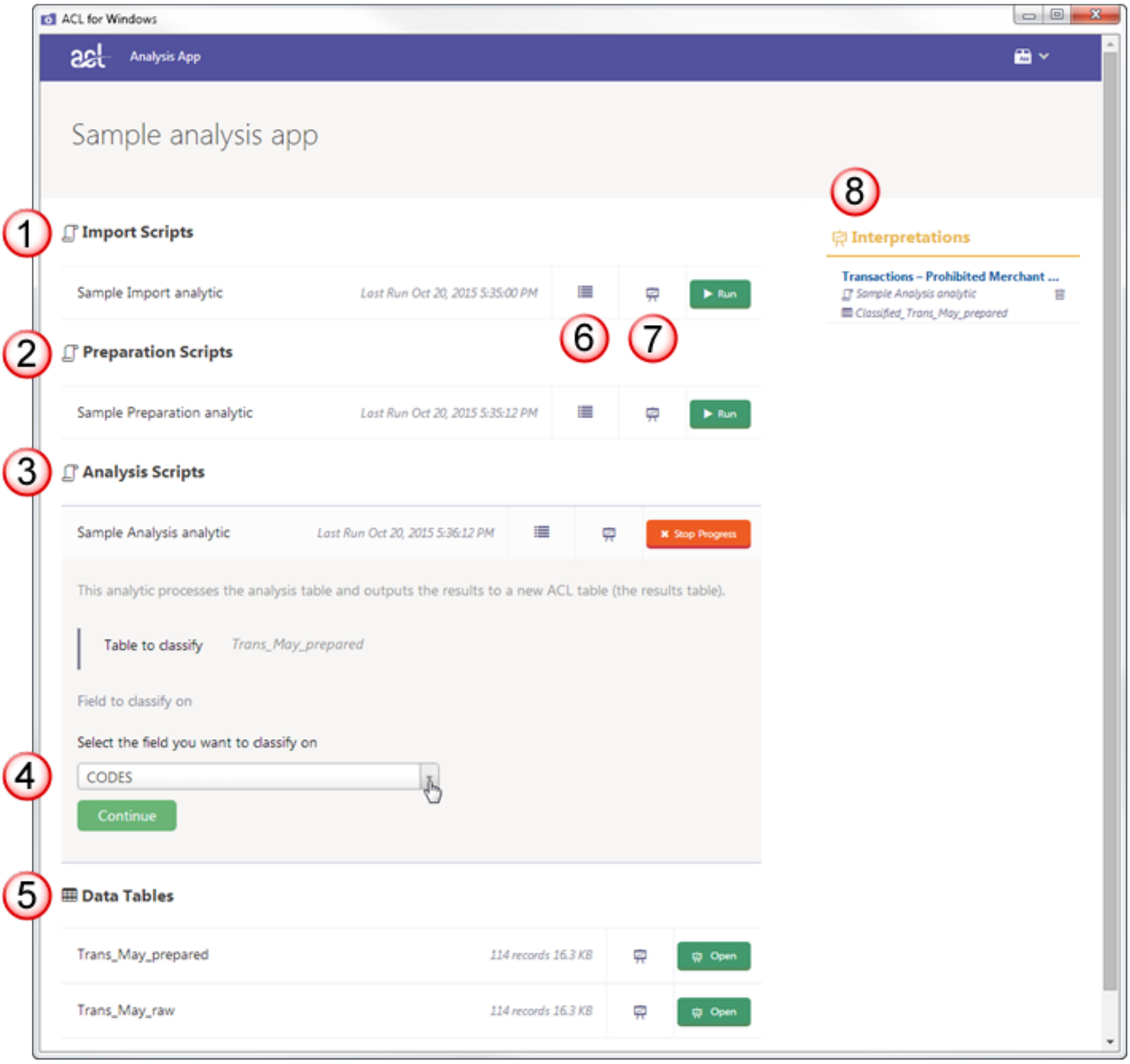
For more information, see "Interpretations and visualizations" on page 2652.

Overview of the Analysis App window

The Analysis App window is a freestanding component of Analytics that provides a simple user interface for running analytics, and bundled sets of analytics called analysis apps. If input values are required to run one or more of the analytics in an analysis app, the interface guides you through providing the values.

The Analysis App window also allows you to create data interpretations and visualizations based on the output results generated by analytics, or based on other tables in an analysis app, or in an Analytics project.

The Analysis App window appears below with a sample analysis app.



The Analysis App window includes the following user interface elements:

1	Import Scripts	Contains any analytics designated as import analytics
2	Preparation Scripts	Contains any analytics designated as preparation analytics
3	Analysis Scripts	Contains any analytics designated as analysis analytics, or analytics without a designated type
4	Input values	Fields that allow you to enter input values for analytics that require input values
5	Data Tables	Lists all data tables currently saved with the analysis app

6	Results tab (not shown)	Lists any results table or tables output by an analytic and provides an option for opening the table The tab also lists any log file output by an analytic.
7	Saved Inter-pretations tab (not shown)	Lists any interpretation based on a results table output by an analytic, and provides an option for opening the interpretation The tab also lists interpretations based on a data table.
8	Interpretations panel	Lists all the interpretations currently saved with the analysis app, and the name of the table associated with each interpretation. Interpretations associated with results tables also display the name of the associated analytic. Clicking an interpretation name opens the interpretation.

Running analytics in the Analysis App window

You can use the Analysis App window to run one or more analytics in an analysis app and output results. If an analytic requires input values such as table or field names, you are automatically prompted for the values. If input values from a previous run of an analytic are prefilled, you can accept the previous values, or provide updated values.

Only the most recent set of input values for an analytic is saved. Unlike analytics run on AX Server, you cannot save multiple sets of input values for an analytic in the Analysis App window.

To run an analytic in the Analysis App window:

1. Open an analysis app by doing one of the following:
   - Double-click an analysis app file with an **.aclx** extension.
   - Double-click a packaged analysis app file with an **.aclapp** extension to first install the analysis app and then automatically open it.
   - Double-click the desktop shortcut for ACL for Windows and select an analysis app (.aclx) under **Recent Analytics Files**, or select **Open Analysis App**, select .aclx or .aclapp from the file type drop-down list, navigate to an analysis app file or a packaged analysis app file, and double-click the file.
   - To open a different analysis app if an analysis app is already open in the Analysis App window, click **App**  in the top right corner, select **Open Analysis App**, click **Open an Analysis App** in the center of the window, navigate to an analysis app file (.aclx) or a packaged analysis app file (.aclapp), and double-click the file.
2. If you double-clicked a packaged analysis app file, in the **Browse For Folder** dialog box select a location to install the analysis app and click **OK**.

   The analysis app is installed in the location you specified and automatically opened in the Analysis App window.
3. Do one of the following:
   - If the analysis app contains a single analytic, click **Run** beside the name of the analytic.
   - If the analysis app contains multiple analytics, review the analytics to see if they should be run in a specific order.

     Once you have confirmed the appropriate order, click **Run** beside the name of the analytic that should run first.

     Analytics arranged in sections should be run in the following order: **Import Scripts**, **Preparation Scripts**, **Analysis Scripts**. Within these individual sections, analytics may have been arranged in alphanumeric order by the analytic author, which implies running the analytics in the same order.

> **Note**
> Running analytics out of order will probably cause one or more of the analytics to fail.

4. Do one of the following:
   - If the **Run** button appears, click **Run**. The analytic does not require input values.
   - If the **Continue** button appears, click **Continue**. The analytic requires input values.

   If no input values are required, the analytic runs. If the analytic runs successfully to completion, it displays a status of **Succeeded**. If the analytic does not run successfully to completion, it displays a status of **Failed**.
5. If you clicked **Continue**, do one of the following:
   - If previous input values are not prefilled, provide the input values in accordance with the prompts presented by the analytic, and click **Continue** after you have provided each value. When you have provided all the values, click **Run**.
   - If previous input values are prefilled, click **Run** to accept the previous values, or click the names of individual parameters to update values selectively. Click **Update** after you have updated each value. When you have finished updating values, click **Run**.

   > **Note**
   > If multiple input values can be entered in a single field, press **Enter** after each value, so that each value is on a separate line – unless the analytic instructs you to do something else.

   The analytic runs. If the analytic runs successfully to completion, it displays a status of **Succeeded**. If the analytic does not run successfully to completion, it displays a status of **Failed**.
6. If the analytic runs successfully, do one of the following:
   - If the analytic produced a results table, click **Open** to view the table, or to create an interpretation or visualization of the results.
   - If the analytic is part of an analytic sequence, run the next analytic in the sequence.
7. If the analytic fails, and produces a log file (*analytic_name*.log), click **Open** to review the log.

   The log should include an entry, marked with a red X  , that indicates why the analytic failed. You may be able to correct the error yourself, or you may need to refer the error to the analytic author for correction.

Opening Analytics tables in the Analysis App window

You can open any Analytics table in the Analysis App window, and create a data interpretation or visualization for the table, without having to manually create an analysis app. If the table's source data is updated, the interpretation or visualization dynamically updates when you reopen it.

When you open an Analytics table in the Analysis App window, an analysis app is automatically created in the same folder as the Analytics project. The analysis app has an .aclx file extension, and the same name as the Analytics project. A subfolder containing files required by the analysis app is also automatically created. The analysis app and the accompanying subfolder are where any data interpretations or visualizations are saved.

You can delete the analysis app and the accompanying subfolder at any point. Deleting them does not affect the Analytics project or its source data files in any way.

> **Note**
>
> The names of the Analytics project and the analysis app must be identical in order for them to be linked. If you change one or both names so that they are not identical, you will break the link and any saved interpretations or visualizations will not be available from the Analytics project.

To open an Analytics table in the Analysis App window:

1. Open a project in Analytics.
2. In the **Overview** tab, right-click any table, or the Analytics project entry, and select **Open as Analysis App**.

   The Analysis App window opens and contains all tables in the Analytics project.
3. In the Analysis App window, click **Open** beside any table for which you want to create an interpretation or visualization.

   The table opens in the Visualizer.
4. Do one or both of the following:
   - Click **Toggle filter configurations panel** to sort or filter the table.

     For more information, see .
   - Click **Add Visualization** and choose a chart type to begin creating a chart.

     For more information about creating charts, see .
5. To save your work, click **Save**, enter a title for the interpretation, and click **Save**.
6. To return to the Analysis App window, click **Back to Analysis App**.
7. To exit the Analysis App window, click **Close**.

Packaging analysis apps for use in the Analysis App window

To allow other users to run analytic scripts in the Analysis App window, package an Analytics project into an analysis app (an `.aclapp` file).

> **Note**
>
> If at least one script in an Analytics project contains an analytic header, the project can be packaged as an analysis app.
>
> Before packaging an analysis app make sure you validate the analytic header of each analytic script in the Analytics project.

Distributing a packaged analysis app to users

Use a packaged analysis app (`.aclapp`) to distribute a project to users. Users extract an `.aclx` file from the packaged analysis app and open the contents in the Analysis App window.

In the Analysis App window, users can run the analytic scripts and create interpretations based on tables and output results.

Include existing interpretations

To include the interpretations from an existing analysis app (`.aclx`), you merge the Analytics project with the existing `.aclx` file during the creation of the packaged analysis app (`.aclapp`).

> **Note**
>
> When you use an existing analysis app (`.aclx` file), the contents of the Analytics project take precedence. If there are scripts or tables in the `.aclx` file that no longer exist in the `.acl` file, they are not included in the resulting packaged analysis app (`.aclapp` file).

File size limitation

To use a packaged analysis app successfully, you must ensure that the sum of all file sizes included in the package does not exceed 800 MB before packaging the analysis app. If the prepackaged files exceed this combined size, data files may be corrupted when unpacking the analysis app.

Package a new analysis app

1. In Analytics, right-click the project entry in the **Overview** tab of the **Navigator** and select **Package Analysis App**.

   The Analytics project is the top-level folder in the tree view.

2. In the **Select Tables** dialog box, do the following:
   a. If you want to include one or more of the project tables and associated data files in the analysis app, select the table(s) and the data file(s) to include.

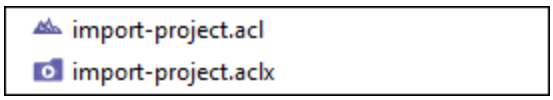
      > **Note**
      > Generally you should include only static tables and data files that are required by one or more of the analytic scripts in the analysis app, such as a master vendor table, or a list of merchant category codes.

   b. Click **To** and navigate to the location where you want to save the packaged analysis app.
   c. In the **Save As** dialog box, enter a **File name** with the `.aclapp` file extension and click **Save**.
   d. Click **OK**.

Result – the packaged analysis app is saved to the location you specified. Other users can retrieve the packaged analysis app from this location, or you can distribute it by email, or by other appropriate method.

Package an analysis app with existing interpretations

1. Ensure that the following files are in the same folder on your computer, and that they have the same name:
   - the Analytics project file (`.acl`)
   - the analysis app file (`.aclx`) containing the existing interpretations that you want to include

2. In Analytics, right-click the project entry in the **Overview** tab of the **Navigator** and select **Package Analysis App**.

   The Analytics project is the top-level folder in the tree view.

3. In the **Select Tables** dialog box, do the following:
   a. If you want to include one or more of the project tables and associated data files in the analysis app, select the table(s) and the data file(s) to include.

> **Note**
> Generally you should include only static tables and data files that are required by one or more of the analytic scripts in the analysis app, such as a master vendor table, or a list of merchant category codes.

b. Optional. To include the interpretations from the existing analysis app, select **Include Interpretations**.

   Interpretations that are associated with tables or scripts that do not exist in the new package are not included.

c. Click **To** and navigate to the location where you want to save the packaged analysis app.
d. In the **Save As** dialog box, enter a **File name** with the `.aclapp` file extension and click **Save**.
e. Click **OK**.

Result – the packaged analysis app is saved to the location you specified. Other users can retrieve the packaged analysis app from this location, or you can distribute it by email, or by other appropriate method.

Interpretations and visualizations

An interpretation is a bundled collection of filters, visualizations, and statistics based on a table in a collection. Use them to interpret and visualize results to gain a deeper understanding of the facts and insights hidden in the data.

> **Note**
> Interpretations are available in the Analysis App window.

How it works

The structure of interpretations

An interpretation contains a **table view** of the data that you can filter, sort, or format. You can then add one or more charts to the interpretation to visualize the data in the table view.

Each chart is based on the data in the **table view** and any changes you make in the table view are applied to the charts as well.

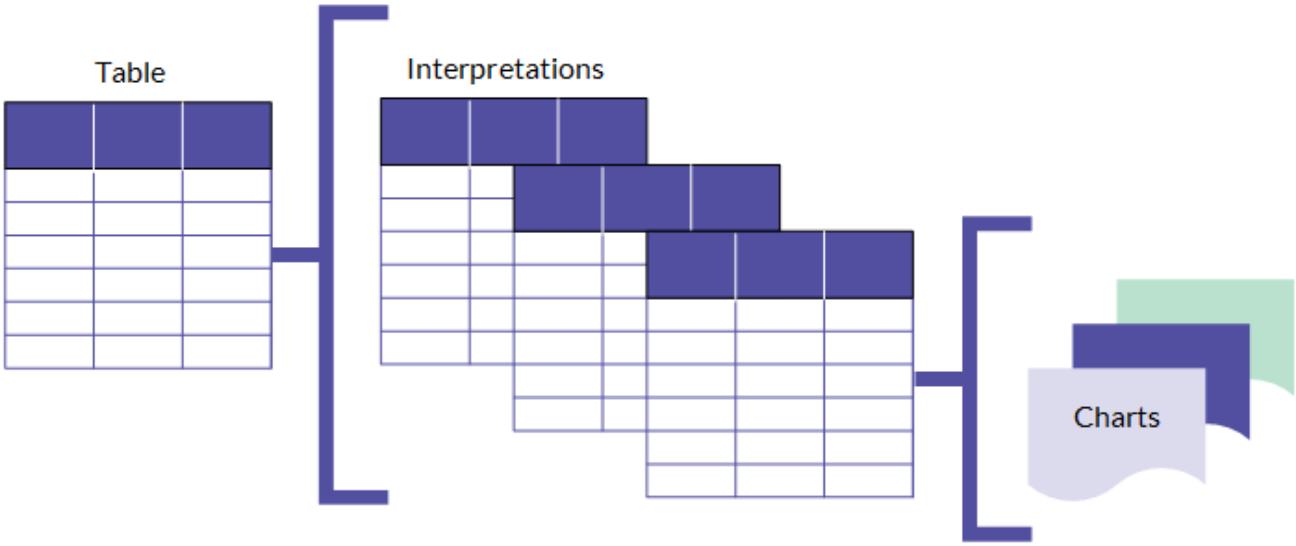


Table and field names

Interpretations and visualizations use the table names and field names that are specified in any analytics you run, or that are specified in the Analytics project that was used to create the analysis app. You cannot change tables names or field names in an analytic, an interpretation, or a visualization.

The field names you see in the Table View in the Analysis App window, and in visualizations, are display names. The analysis app also stores the physical field names.

> **Tip**
> To view the physical field name from the Table View, click the column header. The physical field name is listed as the Field Name value.

Interpreting results data

Open an interpretation

To open an interpretation, do one of the following:

- In the **Interpretations** panel, on the right side of the Analysis App window, click the name of the interpretation you want to open.
- Click **Saved Interpretations** beside the name of an analytic or a data table, and then click **Open** beside the name of the interpretation you want to open.

> **Note**
> If the message **Failed to load table** appears, and the saved interpretation does not open, the table associated with the interpretation cannot be found.

Result – The saved interpretation opens, including the **Table View** and any filters, charts, and statistics that were saved as part of the interpretation.

Create or save an interpretation

On the navigation bar, click **Save** and then do one of the following:

- To save the interpretation, select **Save** and enter a name for the interpretation.
- To save the interpretation as a new interpretation, select **Save As** and enter a name for the interpretation.

Delete an interpretation

If you no longer need an interpretation, you can delete it. Deleting an interpretation also deletes any visualizations created from that interpretation and removes those visualizations from storyboards. It does not delete the table this interpretation exists in.

To delete an interpretation, do one of the following:

- In the **Interpretations** panel, on the right side of the Analysis App window, click **Delete** beside the name of the interpretation you want to delete.
- Click **Saved Interpretations** beside the name of an analytic or a data table, and then click **Delete** beside the name of the interpretation you want to delete.

Result – The interpretation is deleted.

Viewing table data

Use the **Table View** to display table data. In the **Table View**, you can re-order, sort, filter, format, and export data.

Open the table view

To open the **Table View**, click **Open** beside a data table, a results table, or a saved interpretation.

Reorder or hide columns

1. On the right-hand side of the page, click  **Configure**.
2. In the **Configure Table View** panel, do any of the following and click **Apply**:
   - To hide all fields, switch off the **ALL** toggle.
   - To hide a single field, switch off the toggle next to the field name.
   - To display a single field, switch on the toggle next to the field name.
   - To reorder columns, click and drag the field name up or down to the new position.
     The order in list matches the order in the Table View.
3. Click to close the **Configure Table View** panel.

Sort column data

1. Do one of the following:
   - Click the header of the column you want to sort.

     > **Tip**
     > If you only want to sort data, this method is quickest.

   - Click the filter icon  to open the **View & Add Filters** side panel.
     If you want to create nested filters in addition to sorting data, you need to use this method.
2. If you are sorting using a column header, click the Ascending or Descending toggle button.
   The data in the **Table View** is sorted.
   To remove the sort, click the column header and depending on which type of sort you applied, click the Ascending or Descending toggle button again.
   You can also switch the sort order. If you save the interpretation, the sort order is saved.

3. If you are sorting using the **View & Add Filters** panel, do the following:
   a. Click the arrow in front of **Sort by Column.**
   b. Click **Select field** and select a field from the drop-down list.
   c. Click **Normal order A>Z** (Ascending) or **Reverse order Z-A** (Descending).
   The data in the **Table View** is sorted. To remove the sort, click **Remove sort** .
   You can also switch the sort order, or select a different column to sort by.
   You can leave the **View & Add Filters** panel open, or click in the title bar to close the panel.
   If you save the interpretation, the sort order is saved.

Format column data

1. Click the header of the column you want to format and in the dialog box select **Options**.
2. From the list of display format options, select the format you want to apply to the data in the column.

   Result - The window closes and the formatting is applied to the table.
3. To save the interpretation, on the right-hand side of the page, do one of the following:
   - Click **Save**.
   - Click **More Actions** and select **Save As**.

   When you save, you can edit the title and optionally enter a summary for the visualization.

For more information about formatting column data, see "Data formatting options" on page 2658.

Apply conditional formatting to a column

1. Click the header of the column you want to format and in the dialog box select **Format > Options > Conditional formatting**.
2. To define the condition, click **Add another condition** and do the following:
   a. From the **Select condition** list, select a conditional operator.
   b. Enter a value to test against the conditional operator.

      Any positive number you input cannot be smaller than 10^{-5} or larger than 10^{21}.

   c. Select an icon and icon color or select to disable the icon.
   d. Select a text color or click the toggle to disable text formatting.
   e. Select a background color or click the toggle to disable background formatting.
   f. Optional. To define another condition, click **Add another condition** and repeat steps a to e.

> **Note**
> If a field evaluates to true for more than one condition, the first defined condition takes precedence and that formatting is applied.

3. To save the interpretation, on the right-hand side of the page, do one of the following:
   - Click **Save**.
   - Click **More Actions** and select **Save As**.

   When you save, you can edit the title and optionally enter a summary for the visualization.
4. Optional. To change or remove the formatting applied to a column, click the column header, in the dialog box select **Options > Conditional formatting**, and then update or delete  the condition.

Data formatting options

Numeric, text, date, and datetime fields support a number of different formatting options that are useful in different situations.

Numeric data formats

Supported formats

Numeric fields support the following formatting options:

- number
- currency
- financial
- percent

Rounding options

Any format can be displayed as the exact number, rounded to the nearest whole number, or rounded to two decimal places.

> **Note**
> Rounding is to the nearest decimal.

> **Note**
> The rounding settings you choose here are used in visualizations based on this data.

Text data formats

Text data fields support plain text or a subset of HTML elements.

Supported HTML elements

- bold <b>
- italic <i>
- underline <u>
- ordered list
- unordered list
- anchor <a>
- image <img>

Image tag src attributes may reference images hosted on external domains.

> **Note**
> If you click an HTML link in the table, the page opens within the Table View. Use the browser back button to navigate back to the table.

When to use HTML formatting

HTML formatting is useful for:

- free form text questionnaire responses that include HTML formatting
- `.csv` files that include HTML formatting in specific fields
- records imported from Analytics, Analytics Exchange, or Add-In for Excel that include HTML formatting

Date and datetime data formats

Date and datetime fields support a number of common date formats that are used in different locales and by different systems.

Format column data

1. Click the header of the column you want to format and in the dialog box select **Options**.
2. From the list of display format options, select the format you want to apply to the data in the column.

   Result – The window closes and the formatting is applied to the table.
3. To save the interpretation, on the right-hand side of the page, do one of the following:
   - Click **Save**.
   - Click **More Actions** and select **Save As**.

   When you save, you can edit the title and optionally enter a summary for the visualization.

Filtering table data

Use filters to define precise data sets that are displayed in the table you are working with. As you work with filters, any associated visualizations or metrics update to reflect the filtered data.

Combining filters

You can apply more than one filter to a table to display a specific subset of records. When you add more than one filter, the table displays records that match all filters that you apply. In other words, the filters are joined using AND logic to decide which records to include.

> **Note**
> If you add more than one filter for the same field, you can combine the filters using OR logic. Only filters applied to the same field support OR logic.

Example

You are working with a table of customer data and you want to only look at customers in New York with a credit limit exceeding $50,000. To filter the table so that only these customers are displayed, you create the following filters:

- STATE = NY
- LIMIT > 50000

The filters work together and include only those records where the state is New York and the limit is greater than 50000.

Create a filter

1. To open the filter dialog box, on the right-hand side of the screen, click **Filters**  , click **Add Filter**, and then select the column to filter.

> **Tip**
> You can also access the filter dialog box by clicking the heading of the column you want to filter.

2. In the **Filter** section, do one of the following:
   - From the **Select Condition** list, select a conditional operator to use and then enter the value to test against.

   - To include all records with values equal to a specific value or set of values, select one or more values below the **Filter** section.

     The list displays the first 100 values in sorted order and shows their frequencies in parenthesis. To restrict the list, or to view values beyond the first 100, enter a term in the **Quick Search** field. The search is progressive and case-insensitive.
3. To apply the filter to the table, click **Apply Filter**.
4. **Optional** – To add another filter, click **Add Filter** and repeat steps 2 and 3.
5. **Optional** – To change a filter that you have defined, do any of the following:
   - To change the filter, update the conditional operator and test value or the logical operator and click **Apply Filters**.
   - To temporarily disable a filter, click the toggle .
   - To delete a filter, click ×.

Export data to file

Export interpretation data to a comma-separated values (`.csv`) file so you can open the data in other applications that you use to view data and produce reports.

> **Note**
> You can export up to 100,000 records with up to 30 fields of approximately 15 character length at one time.

1. In the Analysis App window, click **Open** beside the table that you want to export data from.
2. In the table, prepare the data by formatting, filtering, sorting, or reordering columns as necessary.
3. In the top right-hand corner of the table, click **More Actions** and select **Export to CSV**.
4. In the **Save File** dialog box, choose a location to save the file, rename the file if required, and click **Save**.

   Result - the data that is currently showing in the table is exported to a `.csv` file in the location that you specified.

Exporting interpretations to HighBond Results

You can export an entire interpretation from the Analysis App window to Results in HighBond.

In Results, the interpretation appears exactly as it appears in the Analysis App window. All charts, and any data customizations in the **Table View**, such as filtering, sorting, and highlighting, are preserved.

By default, the data in the source table associated with the interpretation is also exported. All the data in the source table is exported, regardless of any data customizations in the **Table View**.

Exporting an interpretation without the source data

You have the option of exporting an interpretation without exporting the source table data. This feature is useful if you have already exported an interpretation with the source table data to Results and you only want to update the interpretation, or export additional interpretations for the same source table data.

If you want to export only the source table data without an interpretation, see Exporting exceptions to Results.

Misaligned data

If you are round-tripping data between Results and Analytics, and data ends up misaligned in Results, you probably have mismatched field names.

For more information, see "Field name considerations when round-tripping Results data" on page 692.

Export an interpretation to Results

1. Create a new interpretation, or open an existing interpretation, in the Analysis App window.
2. In the Visualizer, customize the data in the **Table View**, if required.

> **Note**
> You can also customize the data once it is in Results.

3. In the upper right corner, click the **More Actions** drop-down list ••• and select **Share to ACLGRC**.

   If you have created a new interpretation, you need to click **Save** first.

4. In the **Share Interpretation in ACLGRC** window, navigate to the appropriate data analytic in Results, and click **Share**.
5. If an interpretation with the same name already exists in the data analytic, do one of the following:
   - Click **Replace** to overwrite it.
   - Click **Cancel** if you want to keep it and rename the interpretation you are exporting.

     If you clicked **Cancel**, click the **More Actions** drop-down list ••• and select **Save As** to save the interpretation with a different name. Repeat step 3 and step 4.
6. Do one of the following:
   - Click **Continue** to export the interpretation and the source table data, and click **OK** when the data export is complete.

     > **Note**
     >
     > If source table data already exists in the target data analytic in Results, the exported data is appended to the existing table.
     >
     > Regardless of their order in a table, exported fields are appended to existing fields if they have an identical field name – that is, an identical physical name in the table layout of the source Analytics table.
     >
     > Exported fields that do not match the name of any existing field are added as additional columns to the table in Results.
     >
     > The display name of fields in the **Table View** in the Visualizer is not considered when fields are appended.

   - Click **Do not upload data** to export only the interpretation without the source table data.

Visualizing table data in charts

Visualizations are charts and statistics that you can use to present and communicate table data.

Chart types

Select a chart that can support the number of data dimensions, or variables, you want to use in the chart. Depending on the type of chart, you can display two, three, or four data dimensions.

Type	Available charts	Description
"Bar chart" on page 2675 Compares two or three variables using a single value. A bar chart is a graph with rectangular bars and each bar's length is proportional to the value it represents.	 Stacked Column Chart	A single second dimension column is divided by third dimension values.
	 Stacked Bar Chart	A single second dimension bar is divided by third dimension values.
	 Grouped Column Chart	Third dimension values are displayed as adjacent columns within the second dimension value.
	 Grouped Bar Chart	Third dimension values are displayed as adjacent bars within the second dimension value.
	 100% Stacked Column Chart	A single dimension column is divided by the percentage each third dimension value contributes to the total.

Type	Available charts	Description
	 100% Stacked Bar Chart	A single dimension bar is divided by the percentage each third dimension value contributes to the total.
"Line chart" on page 2694 Shows trends or changes over time by displaying a series of data points connected by straight line segments. You can display a single data field as multiple lines based on different categories through Color by field.	 Line Chart	Plots changes over short or long periods of time and to assist in predictive data analysis. If small, frequent changes exist in the series, line charts are more effective than bar charts at visualizing the change over time.
"Area chart" on page 2685 Compares historical trends or changes by showing the proportion of the total that each category represents at any given point in time. Area charts communicate overall trends rather than individual values when comparing multiple data series.	 Stacked Area Chart	Shows the proportion of the total that each category represents at any given point in time.
	 Standard Area Chart	Shows a quantitative progression over time.
	 100% Stacked Area Chart	Shows how the constituent parts of a whole have changed over time. The y-axis expands to 100% and each colored area represents one part of the whole, with each part stacked vertically. The height of each colored stack represents the percentage proportion of that category at a given point in time.
"Pie chart" on page 2681 Shows categories as a proportion or a percentage of the whole. Use pie charts to show the composition of categorical data with each segment proportional to the quantity it represents.	 Pie Chart	Effective when comparing parts of a whole for a static period. Does not show change over time.

Type	Available charts	Description
"Bubble chart" on page 2690 Communicates the raw count, frequency, or proportion of a variable. Bubble size reflects quantity, bubble color reflects category, and the x and y axes both display independent values.	 Bubble Chart	Effective when you are working with three data series that each contain a set of values. You can add an optional fourth variable for categorization. The bubble size is effective for emphasizing specific values and comparisons between categories.
"Heat map chart" on page 2698 Displays individual values in a matrix and represents value as colors. Data points are defined by an x and y axis intersection and a third value that determines the data point's color.	 Heat Map	Compares variables across a large number of categories and sorts complex data by color intensity.
"Summary table" on page 2704 Groups records on unique values in one or more key fields and then performs a count for the number of matching records.	 Summary Table	Drill into the summary by selecting a column and producing a crosstab. Once you define the summary, you can also select numeric fields to subtotal.
"Treemap chart" on page 2707 Displays hierarchical, tree-structured data as a set of nested rectangles.	 Treemap	Each group is given a rectangle, which is then tiled with smaller rectangles representing sub-groups. Size and color are used to show separate numeric dimensions of the data.
"Statistics" on page 2702 Shows statistical information in a visually attractive format for one or more fields in a table.	 Statistics	Identifies trends or irregularities in a table.

Create a chart

> **Note**
> Custom formatting applied in the **Table View** does not persist in charts or column overview visualizations. Visualizations display data in the default format.

1. From the **Table View**, at the top of the table, click **Add Visualization** + and select the type of chart to create.
2. Replace **Untitled Visualization** with a descriptive title for the chart.

3. On the right-hand side, click **Configure**  and in the **Configure Visualization** panel, click **Data**, specify the data dimensions to include in the chart and click **Apply**.

   For more information about the defining the data dimensions for the chart type you are working with, see the appropriate chart link in "Chart types" on page 2665.

4. On the **Configure Visualization** panel, click **Display** and select the display options for the chart.

   For more information about the defining the display options for the chart type you are working with, see the appropriate chart link in "Chart types" on page 2665.

5. To save the interpretation with the visualization, on the right-hand side of the page, do one of the following:
   - Click **Save**.
   - Click **More Actions** and select **Save As**.

   When you save, you can edit the title and optionally enter a summary for the visualization.

> **Tip**
> If you want to save an image of your interpretation as a local file in `.png` or `.jpg` format, use an image capture tool or screen capture tool to create an image. For more information, see the help for the screen capture tool on your operating system.

Delete a chart

1. From the **Table View**, at the top of the table, select the tab for the chart.

2. On the right-hand side, click **Configure**  and in the **Configure Visualization** panel, click **Data**, and then click **Delete chart**.

Data visualization best practices

Data visualizations are essential to helping people understand the story within the data. Placing the data in a visual context helps patterns, trends, and correlations emerge that might otherwise go unnoticed. To make sure your data visualizations tell compelling stories, follow these best practices.

Know your audience

Before you start designing a data visualization, consider who the primary audience is for the visual representation of the data.

Make sure the visualization answers the questions that are most important to the primary audience. Resist the temptation to create visualizations that meet the needs of any and all potential audiences as this may make the message for your intended audience less clear.

To design for your audience, ask yourself these key questions:

- who is my primary audience?
- how will my audience view this visualization?
- what actions do I want the audience to take based on this data?

Provide context

Data trends and patterns are best demonstrated in the context of larger goals and metrics. By presenting your data visualizations within context, a better story emerges from your data and stakeholders can draw clearer conclusions:

- use color to imply meaning
- compare data to metrics or goals
- ensure data periods are clearly indicated

Example: associating colors with meaning

You need to provide a visualization for customer satisfaction ratings by regional division in your company. To visualize this data, you choose a grouped bar chart and then use color to associate the satisfaction ratings with meaning:

- **Satisfied** - green
- **Neutral** - yellow
- **Dissatisfied** - red

This color to word association makes it easy for your audience to understand the overall meaning of the data without having to study the details closely.

Keep it simple and clear

Your audience has a short attention span. If your visualization cannot be clearly understood within 10 to 15 seconds, your audience may miss the point. Use the following tips to simplify your visualizations and improve clarity:

- use white space to make labels and chart components distinct
- write descriptive labels and headings so that nothing is ambiguous
- align chart colors with the meaning you are trying to convey
- keep text short and simple so it is easy to read
- tell one story at a time to avoid confusing your audience

Understand your data

There are three types of data you may encounter when building visualizations:

- **categorical** – data that belongs together logically but has no intrinsic ordering (department: sales, human resources, IT, etc.)
- **ordinal** – data that belongs together logically and has an intrinsic ordering (education: high school, some college, college graduate, etc.)
- **quantitative** – data that defines a quantity, or how much, of something there is (transaction amount: $400, $100, $175, etc.)

What kind of data are you trying to visualize? Knowing the data you are working with makes choosing the right chart type and communicating meaning an easier proposition.

Choose the right chart type

Once you understand your audience and your data, it is time to select the chart type that best expresses the story in the data:

Chart type	Description	Suitable data type(s)
"Bar chart" on page 2675 	Compares quantities of categorical data	Categorical, Quantitative
"Line chart" on page 2694 	Shows change over time	Ordinal, Quantitative
"Area chart" on page 2685 	Shows how categories contribute to a cumulative total over time	Ordinal, Categorical, Quantitative
"Pie chart" on page 2681 	Shows the parts of a whole	Categorical
"Bubble chart" on page 2690 	Shows correlations between three or four variables **Tip** If no correlation exists, the points appear randomly scattered. If a strong correlation exists, the points concentrate near a straight line.	Quantitative, Ordinal, Categorical
"Heat map chart" on page 2698	Compares variables across a large number of categories and sort data by color intensity	Categorical, Quantitative

Chart type	Description	Suitable data type(s)
		

Next steps... create a chart

For information about creating any of the charts available to you, see "Visualizing table data in charts" on page 2665.

Chart display options

Use chart display options to customize the appearance of your chart.

Setting	Applicable charts	Description
General display settings		
Legend	All charts	Shows or hides the legend from the chart.
Rotate Horizontal	Bar chart	Changes the orientation of the chart from vertical to horizontal.
Round edges	Line chart	Smooths the lines on the chart. Smoothing lines gives a general impression of data trends, which may be appropriate for some presentations. If representing data absolutely accurately is important, avoid using this option.
Percentages	Pie chart	Shows or hides the labels from pie segments.
Values	Pie chart	Displays the values of each slice of the pie chart.
Donut	Pie chart	Displays the chart with a hole in the middle.
Donut hole ratio	Pie chart	Increases or decreases the white space inside the donut hole.
Bottom margin (in pixels)	Pie chart	Adjusts the number to increase or decrease the white space at the bottom of the chart.
Left margin (in pixels)	Pie chart	Adjusts the number to increase or decrease the white space at the left of the chart.
X-Axis display settings		
Show label	◦ Bar chart ◦ Stacked area chart ◦ Bubble chart ◦ Line chart	Shows or hide the X-axis label.
Margin (in pixels)	◦ Bar chart ◦ Stacked area chart ◦ Bubble chart ◦ Line chart	Adjusts the number to increase or decrease the white space at the bottom of the chart.
Tick Mark	◦ Bar chart	Adjusts the number from -90 to 90 to rotate the data marker labels left or

Setting	Applicable charts	Description
Rotation (in degrees)	◦ Stacked area chart ◦ Bubble chart ◦ Line chart ◦ Heat map chart	right. A setting of "0" positions the labels perfectly level.
Stagger Tick Marks	Bar chart	Staggers the data marker labels. The staggered labels are positioned level regardless of the rotation setting.
Y-Axis display settings		
Show label	◦ Bar chart ◦ Stacked area chart ◦ Bubble chart ◦ Line chart	Shows or hide the Y-axis label.
Margin (in pixels)	◦ Bar chart ◦ Stacked area chart ◦ Bubble chart ◦ Line chart	Adjusts the number to increase or decrease the white space to the left of the chart.
Axis Label Distance (in pixels)	◦ Bar chart ◦ Stacked area chart ◦ Bubble chart ◦ Line chart	Adjusts the number to increase or decrease the distance between the Y-axis label and the Y-axis. You can enter a negative number, such as -100, to move the label to the right of the axis and into the chart area.
Data display settings		
Sort by	◦ Bar chart ◦ Stacked area chart ◦ Bubble chart ◦ Line chart	Sort visualization data using the aggregation type in the x-axis according to one of the following orders: ◦ **Ascending** - displays values from lowest to highest ◦ **Descending** - displays values from highest to lowest ◦ **Default** - displays values alphabetically along the x-axis
Colors	All charts	Specifies a color scheme for data segments if you have selected a **Color by** field or a pie chart.

Bar chart

A bar chart is a graph with rectangular bars. Each bar's length is proportional to the value it represents. Use bar charts to compare two or three variables using a single value.

When do you use it?

Basic bar charts

Bar charts are useful for comparing categories: the x-axis represents the category and the y-axis represents the value to compare.

> **Tip**
> You can also use a basic bar chart to create a histogram that displays the distribution of x-axis data. Histograms display the frequency of each unique value in the selected x-axis field as a bar that is proportional to the value.

Complex bar charts

Bar charts can represent more complex categories using grouped or stacked bars. Grouped or stacked bar charts use a third variable to sub-divide the comparison category:

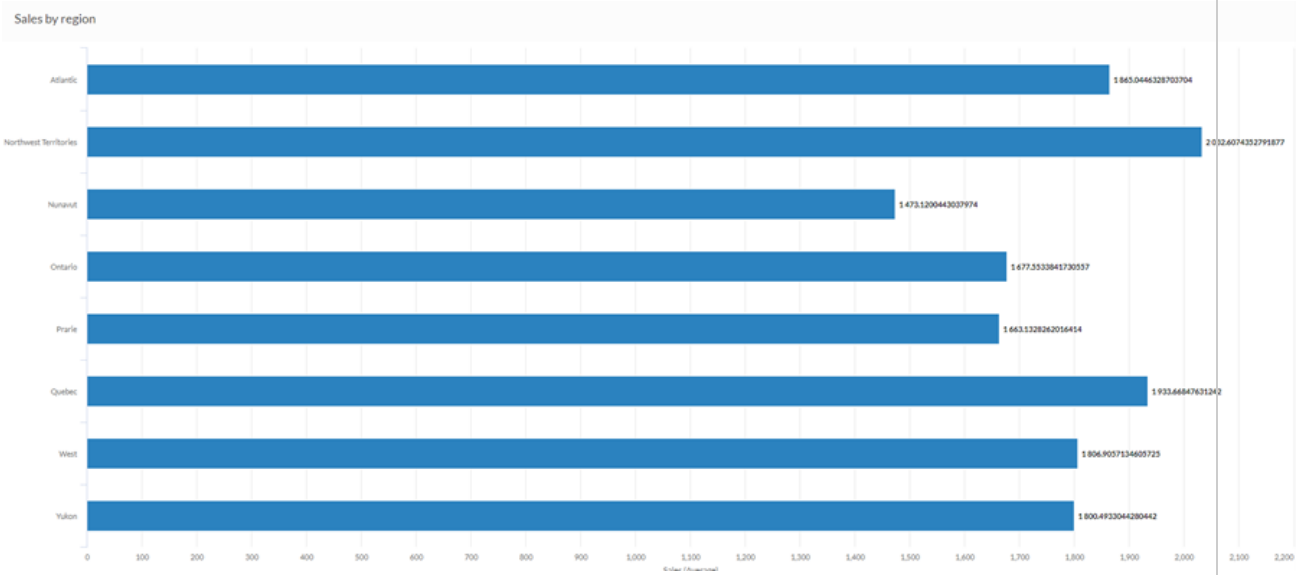
- **grouped** – third dimension values are displayed as adjacent bars within the second dimension value
- **stacked** – a single second dimension bar is divided by the third dimension values
- **100% stacked** – a single dimension bar is divided by the percentage each third dimension value contributes to the total

> **Note**
> Select the grouped vs stacked view setting on the upper left-hand corner of the chart.

Examples

Bar chart

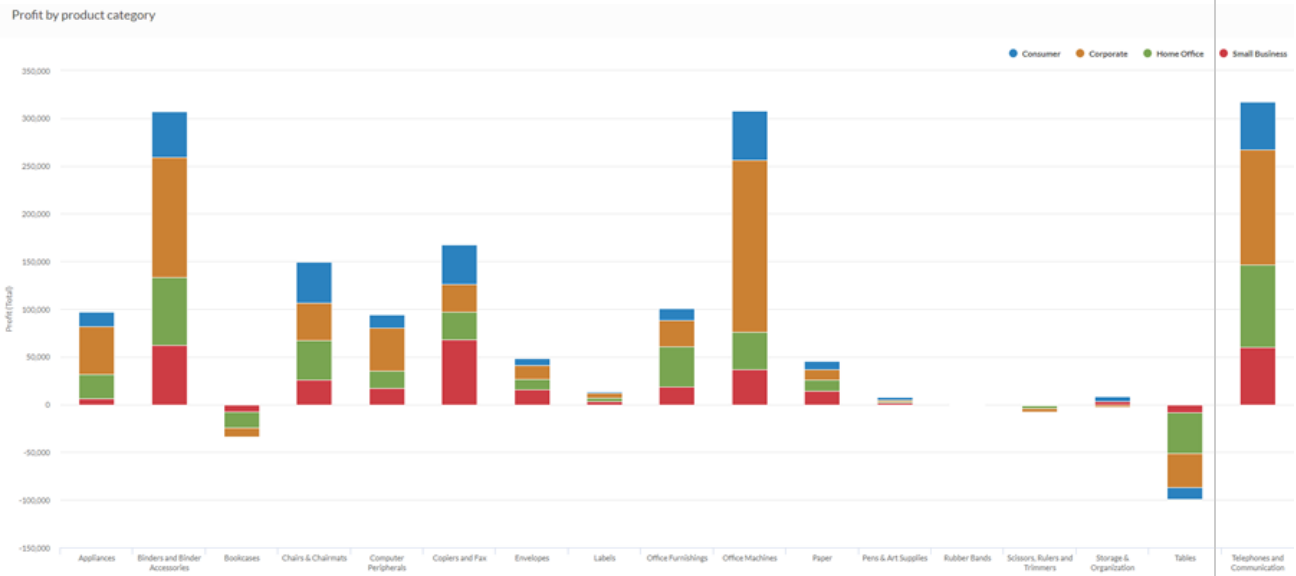
Your company is tracking sales data and needs to determine which regions are enjoying the highest sales numbers. Using a bar chart, you visualize the sales numbers and clearly demonstrate how each region is performing:



Stacked column chart

Your company is tracking sales data and needs to determine which product categories are returning a negative profit.

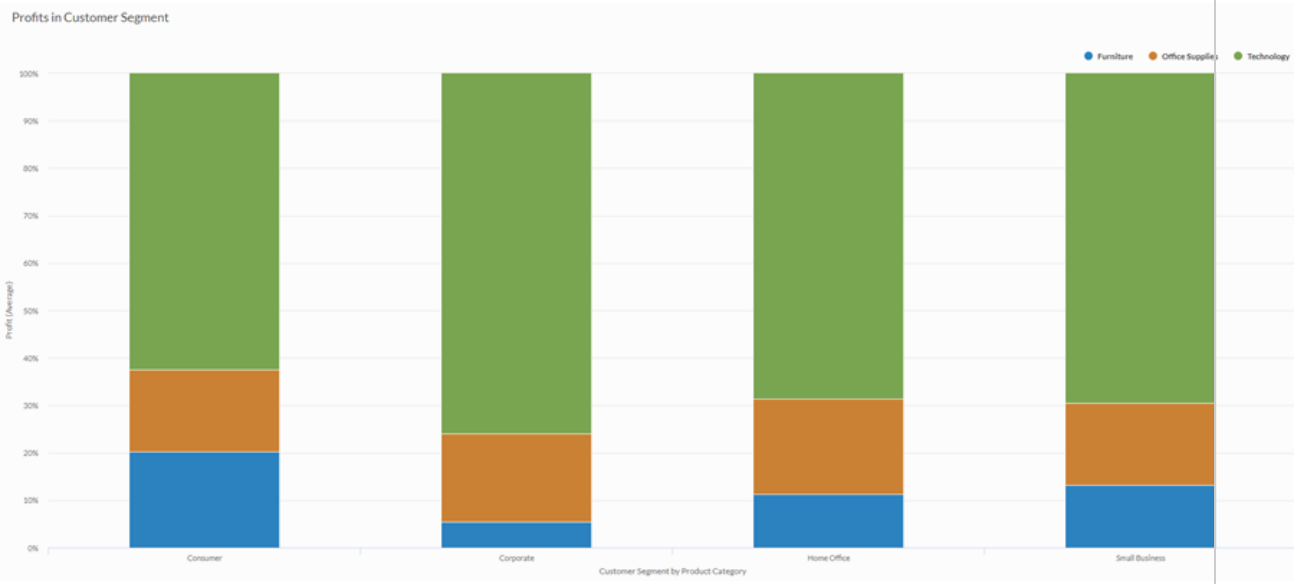
Using a stacked column chart, you visualize the sales numbers and clearly demonstrate which product categories are performing well, and which are not. Stacking the third dimension also helps you determine which segment of each category is most responsible for the overall totals:



100% stacked column chart

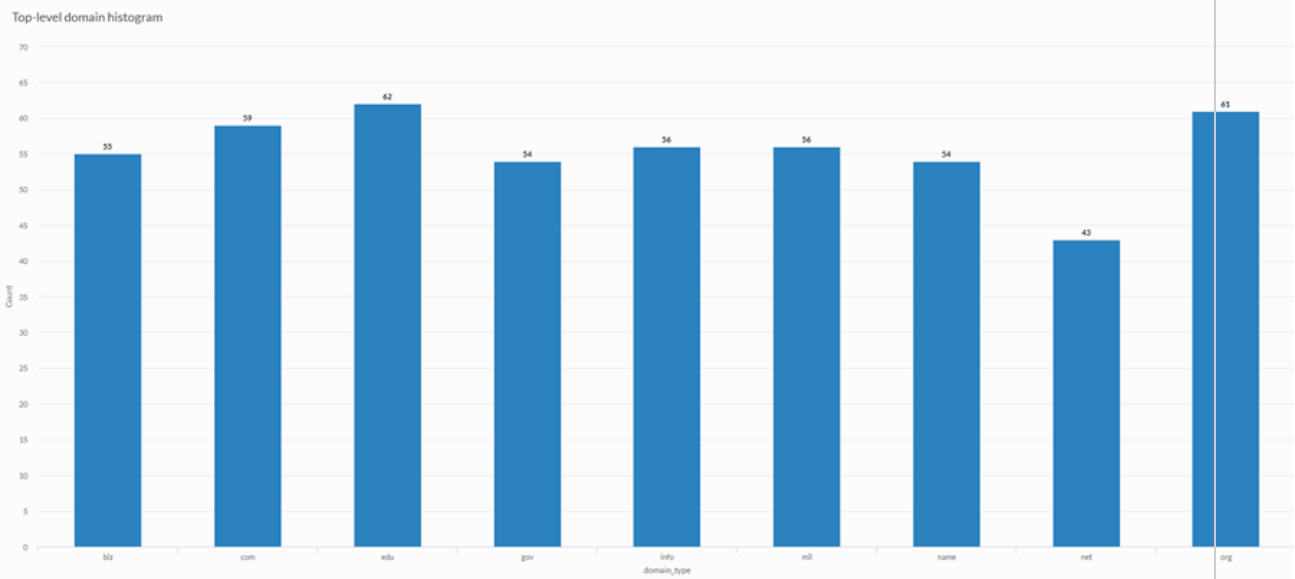
You are performing an analysis of profits by customer segment. You want to visualize the proportion of each product category within the average profit of each customer segment.

To do this, you use a 100% stacked column chart that displays the percentage each product category contributes to the customer segment value:



Histogram

You are performing an analysis of customer web sites and you want to display the frequency of top-level domains used in the collection of sites. To do this, you use a histogram that displays the count of each unique top-level domain value in your table:



Data configuration settings

On the **Configure** panel, click **Data** and configure the following settings:

Setting	Supported data types	Description
X-Axis	◦ character ◦ numeric ◦ datetime	The field to use as the basis for the chart's horizontal scale. One bar is created for each unique value in the field, or each unique combination of values if you also specify a **Color by** field.
Y-Axis	numeric	The aggregate value represented by the chart's vertical axis. You can select a count of the x-axis field or one of several aggregate values for a different numeric column in the table: ◦ average ◦ sum ◦ min ◦ max

Setting	Supported data types	Description
		Tip Use the **Count** option to create a histogram that displays the distribution of data for the x-axis. **Tip** You can control decimals and rounding on numeric data by changing the format of this field in the table view. For help doing this, see "Data formatting options" on page 2658.
Color by optional	character	The field represented by the third data dimension to the chart. Adding a third data dimension can sub-divide x-axis categories. You can think of the **Color by** field as a break field.
Format options	numeric	Select an option in this field to apply formatting such as decimals and rounding to the Y-axis values in the chart. For help doing this, see "Data formatting options" on page 2658. Format options is available only for the **Average** aggregate option. For all other aggregate options, format options set in Table View is applied.

Chart display settings

On the **Configure**  panel, click **Display** and configure the following settings:

Setting	Description
Options	
Show Legend	Show or hide the legend at the top of the chart.
Show Values	Show or hide the data point values.
Rotate Horizontal	Display the bar chart with the x-axis on the left-hand side of the chart and the y-axis running horizontally along the bottom of the chart.
Chart Type	How to display the third dimension of the chart (**Color by**): ◦ **Grouped** - use to compare the value of each segment within each category ◦ **Stacked** - use to compare the value each segment contributes to the total of the overall category while comparing category totals against each other ◦ **100% Stacked** - use to compare the percentage each segment contributes to the total of each category
X-Axis	

Setting	Description
Show Label	Show or hide the label for the x-axis.
Y-Axis	
Show Label	Show or hide the label for the left y-axis.
Min	The minimum value to use for the left y-axis. By default, the chart uses the lowest value of the left y-axis data to determine the minimum.
Max	The maximum value to use for the left y-axis. By default, the chart uses the highest value of the left y-axis data to determine the minimum.
Other settings	
Sort by	Sorts the data categories by ascending or descending total value. By default, the chart is sorted alphabetically.
Colors	The colors assigned to each series in the **Color by** dimension.

Pie chart

Pie charts show categories as a proportion or a percentage of the whole. Use pie charts to show the composition of categorical data with each segment proportional to the quantity it represents.

When do you use it?

Pie charts are effective when comparing parts of a whole for a static period. They do not show change over time.

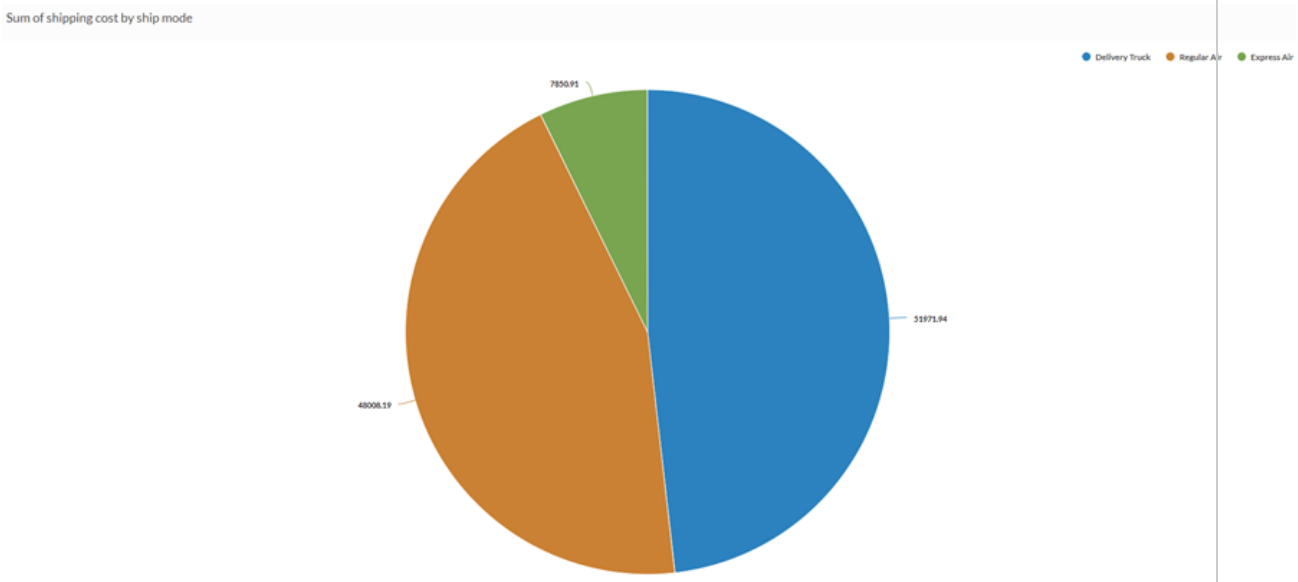
Consider using a pie chart if you are working with:

- **one data set** – pie charts are effective for categorizing and comparing one data set
- **positive values** – pie charts cannot display zeros and can be confusing when working with negative values
- **seven or fewer categories** – it becomes increasingly difficult to perceive the relative size of each segment when working with more than seven categories

Examples

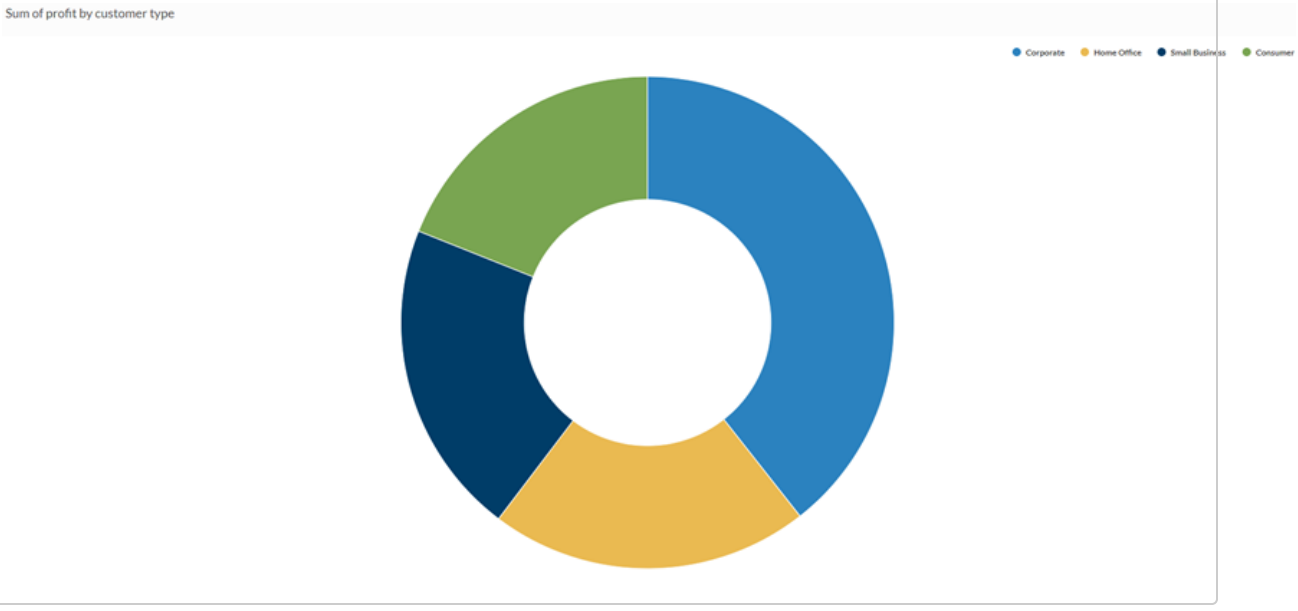
Standard pie chart

You need to determine which shipping methods represent the greatest total proportion of the total amount your company pays in shipping costs. Using a pie chart, you demonstrate the percentage of total cost that each method represents:



Donut chart

You need to determine if there is a relationship between customer type and profit. Using the pie chart, you display the sum of profit for each category of customer to show which customer types represent the most important segment of your sales:



Data configuration settings

On the **Configure**  panel, click **Data** and configure the following settings:

Setting	Supported data types	Description
Categories displayed	◦ character ◦ numeric ◦ datetime	The field represented by pie chart sections. One section is created for each unique value in the field.
Value	numeric	The aggregate value that serves as the basis for the chart's values. The total of all values in the field for each category determines the size of the pie chart sections. **Tip** You can control decimals and rounding on numeric data by changing the format of this field in the table view. For help doing this, see "Data formatting options" on page 2658.
Format options	numeric	Select an option in this field to apply formatting such as decimals and rounding to the Y-axis values in the chart. For help doing this, see "Data formatting options" on page 2658. Format options is available only for the **Average** aggregate option. For all other aggregate options, format options set in Table View is applied.

Chart display settings

On the **Configure**  panel, click **Display** and configure the following settings:

Setting	Description
Options	
Show Legend	Show or hide the legend at the top of the chart.
Show Values	Show or hide the data point values.
Show Percentages	Show or hide the percentage that each proportion represents of the whole.
Donut	Display the pie chart as a donut chart rather than a traditional pie chart.

Setting	Description
Ratio	Controls the size of the donut hole in relation to the chart as a whole. Specify a larger value to increase the size of the hole. The maximum value is 0.7.
Other settings	
Colors	The colors assigned to each category.

Area chart

Area charts compare historical trends or changes by showing the proportion of the total that each category represents at any given point in time. They communicate overall trends rather than individual values when comparing multiple data series.

When do you use it?

Use area charts to show how each category contributes to a cumulative total over time.

Few charts are as effective at representing time-series relationships as area charts. The area component of the chart represents the volume or proportion of the whole in the space between the axis and the data point.

> **Tip**
> A time series is a series of data points listed in time order, typically at equal intervals.

Available types of area charts

The following types of area charts are available:

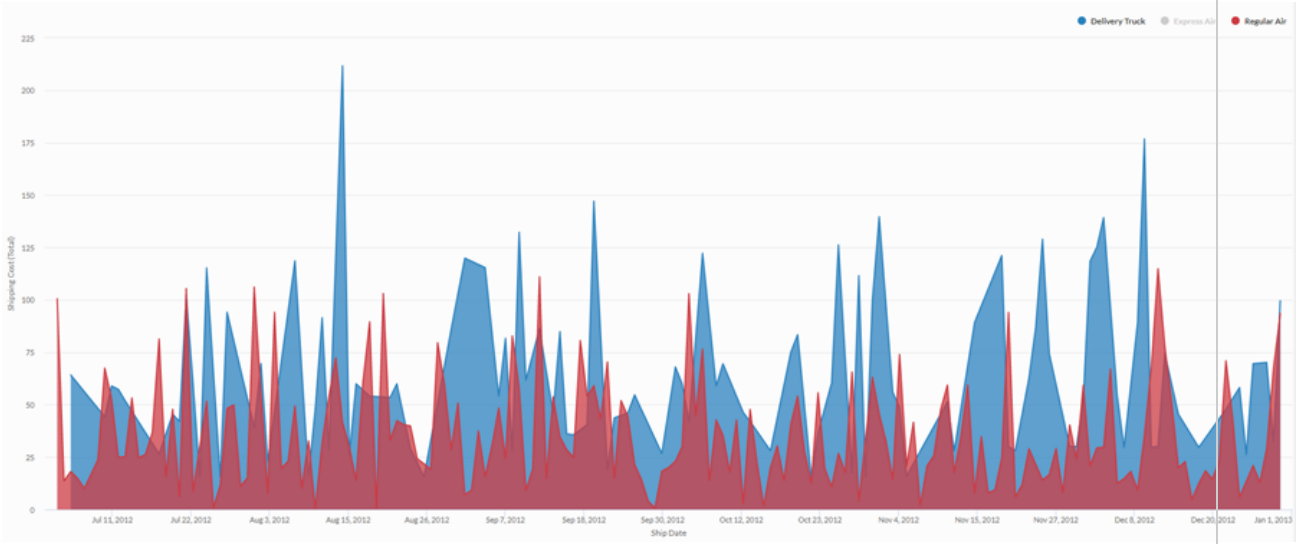
- **Standard** – shows a quantitative progression over time
- **Stacked** – shows the proportion of the total that each category represents at any given point in time
- **100% Stacked** – shows how the constituent parts of a whole have changed over time

  The y-axis expands to 100% and each colored area represents one part of the whole, with each part stacked vertically. The height of each colored stack represents the percentage proportion of that category at a given point in time.

Examples

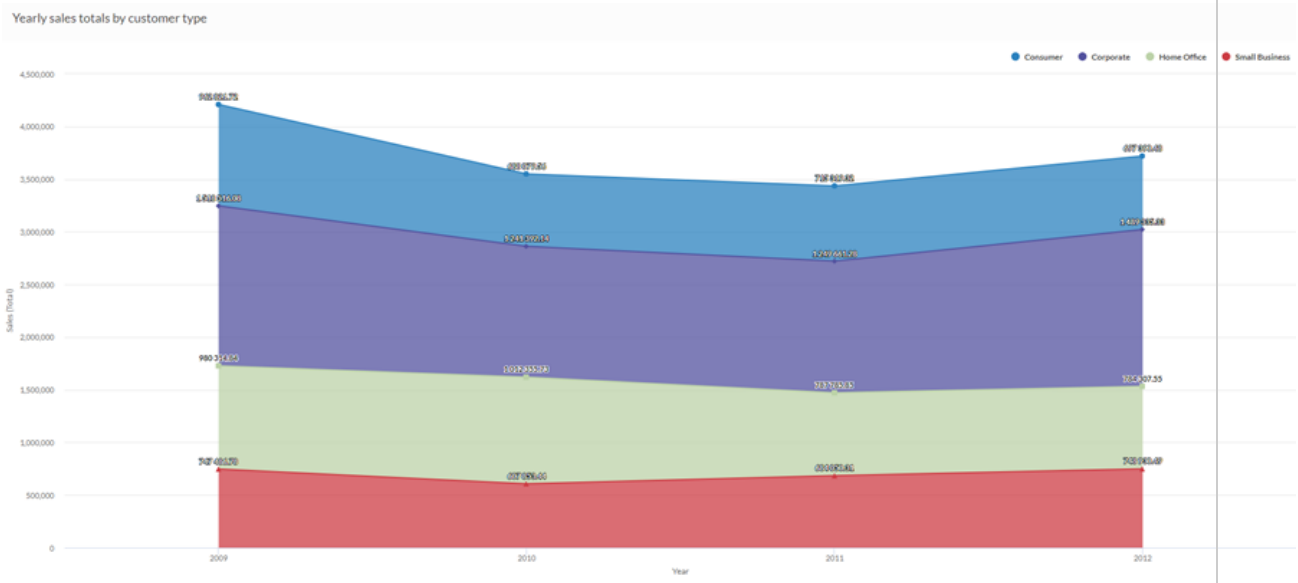
Standard area chart

Using aggregated sales data from a four-year period, you need to communicate the trend in shipping costs for each shipping method your company uses. To do this, you create a standard area chart that shows the trend in costs, broken down by shipping method, over the period:



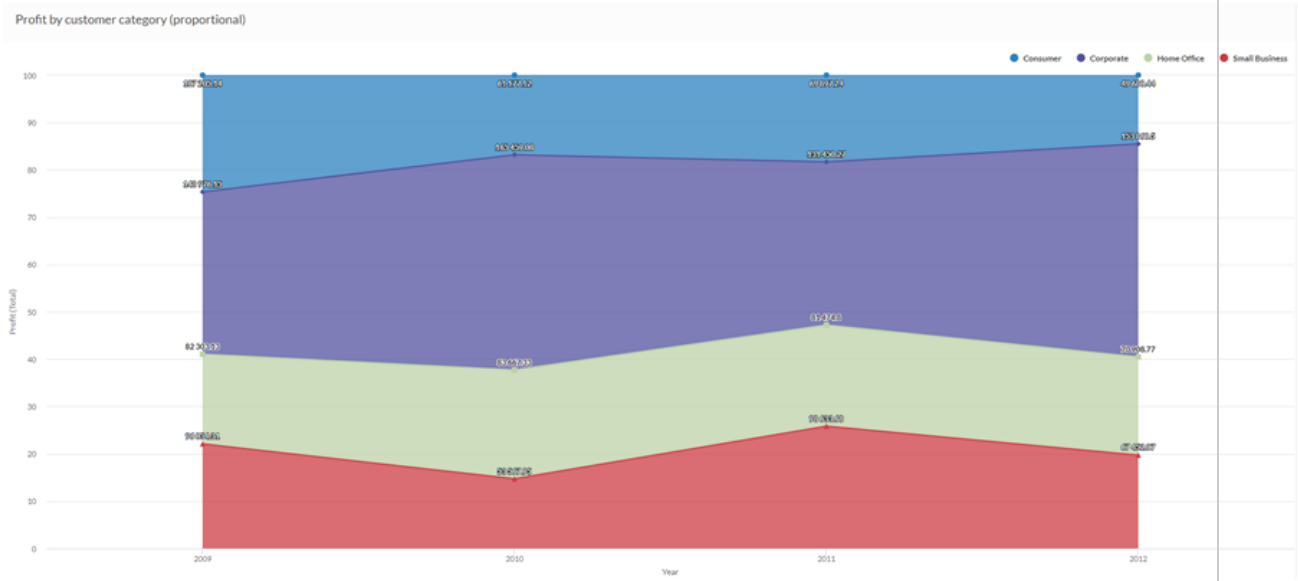
Stacked area chart

Using aggregated sales data from a four-year period, you need to communicate the proportional trend in overall sales, broken down by customer segments over time. To do this, you create a stacked area chart that show the trend over the period:



100% stacked area chart

Using aggregated sales data from a four-year period, you need to communicate how sales to different customer segments have changed over time. To do this, you create a 100% stacked area chart that shows how each segment has performed over time in relation to the total:



Data configuration settings

On the **Configure** panel, click **Data** and configure the following settings:

Setting	Supported data types	Description
X-Axis	◦ character ◦ datetime ◦ numeric	The field to use as the basis for the chart's horizontal scale.
Y-Axis	numeric	The aggregate value represented by the chart's vertical axis. You can select a count of the x-axis field or one of several aggregate values for a different numeric column in the table: ◦ average ◦ sum ◦ min ◦ max The position of data points on the vertical scale determines the height of each line. The height of a line is interpolated or gapped if a data point is missing.

Setting	Supported data types	Description
		Tip You can control decimals and rounding on numeric data by changing the format of this field in the table view. For help doing this, see "Data formatting options" on page 2658.
Color by optional	character	The field represented by the third data dimension to the chart. Adding a third data dimension creates the categories represented by stacked areas. A separate stacked area is created for each unique value in the field.
Format options	numeric	Select an option in this field to apply formatting such as decimals and rounding to the Y-axis values in the chart. For help doing this, see "Data formatting options" on page 2658. Format options is available only for the **Average** aggregate option. For all other aggregate options, format options set in Table View is applied.

Chart display settings

On the **Configure**  panel, click **Display** and configure the following settings:

Setting	Description
Options	
Show Legend	Show or hide the legend at the top of the chart.
Show Values	Show or hide the data point values.
Interpolate	Handle missing data points by connecting the line using the available data points, but do not plot the missing data point on the x-axis. If disabled, the line is not connected across missing data points.
Chart Type	The type of area chart to use: ◦ Standard ◦ Stacked ◦ 100% Stacked
X-Axis	
Show Label	Show or hide the label for the x-axis.
Y-Axis	
Show Label	Show or hide the label for the left y-axis.

Setting	Description
Min	The minimum value to use for the left y-axis. By default, the chart uses the lowest value of the left y-axis data to determine the minimum.
Max	The maximum value to use for the left y-axis. By default, the chart uses the highest value of the left y-axis data to determine the minimum.
Other settings	
Colors	The colors assigned to each series in the **Color by** dimension.

Bubble chart

Bubble charts communicate the raw count, frequency, or proportion of a variable. Bubble size reflects quantity, bubble color reflects category, and the x and y axes both display independent values.

When do you use it?

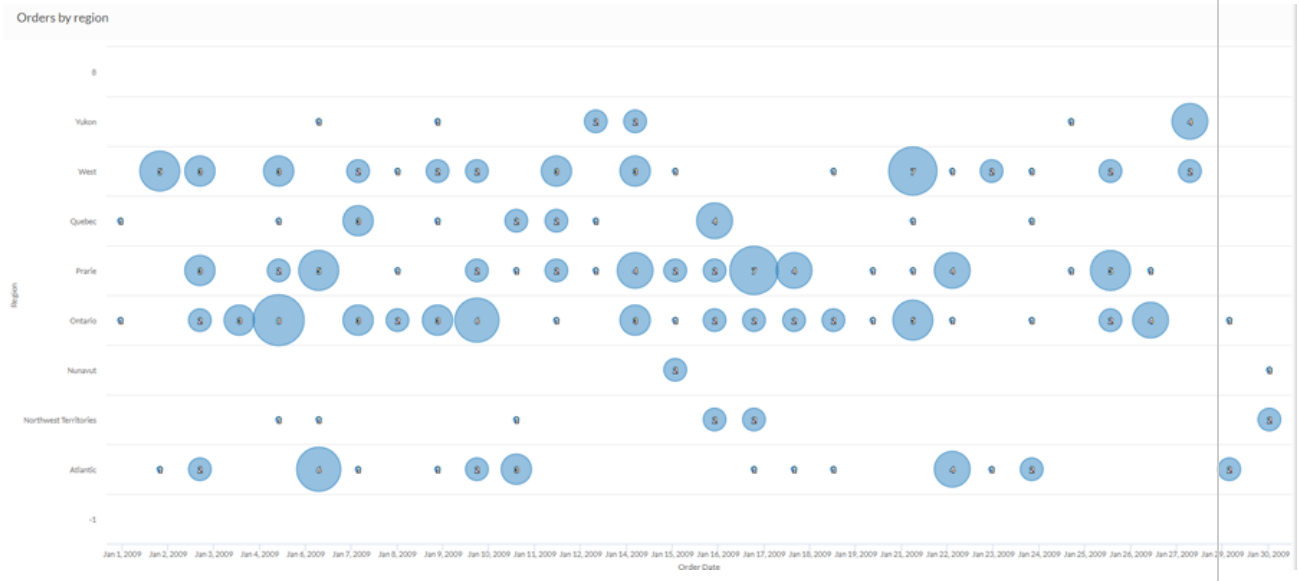
Use bubble charts when you are working with three data series that each contain a set of values. You can add an optional fourth variable for categorization. The bubble size is effective for emphasizing specific values and comparisons between categories.

When designing bubble charts, use clear and visible labels as these charts require clear labeling to help interpret the visualization.

Examples

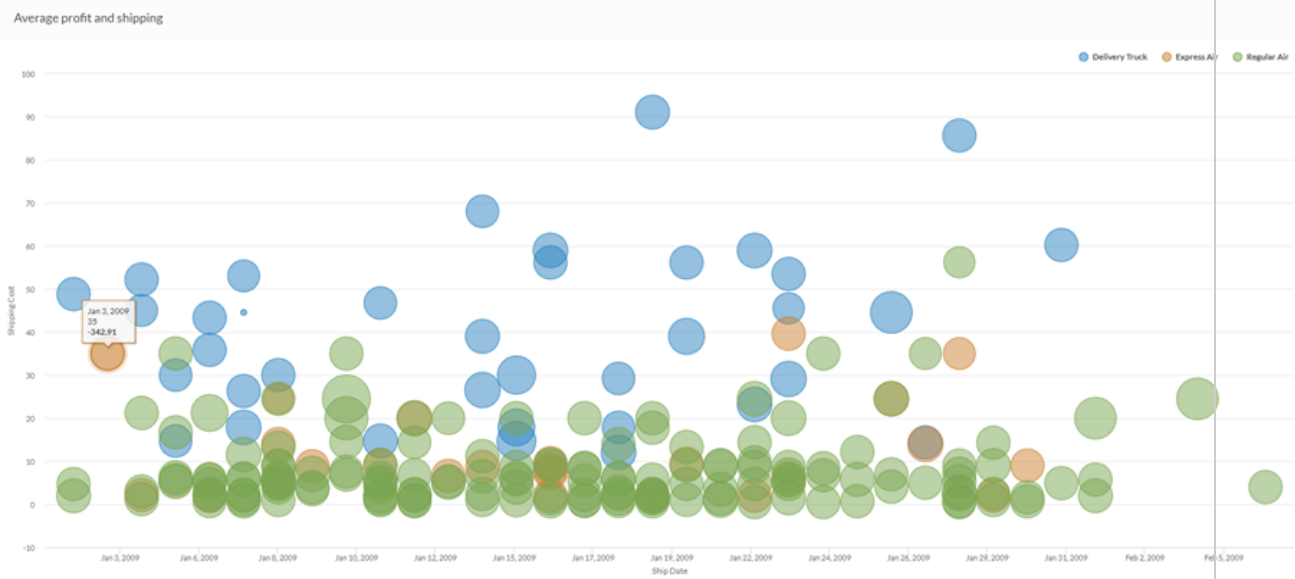
Three dimensional bubble chart

Using sales data from a single month, you want to identify the proportion of orders placed from each region over the period. Using a simple bubble chart, you plot the count of orders by region over the month:



Four dimensional bubble chart

Using sales data from a single month, you want to identify any relationships between shipping and profit. Using a complex bubble chart, you plot the average profit as it relates to shipping cost and shipping method over the month:



Data configuration settings

On the **Configure** panel, click **Data** and configure the following settings:

Setting	Supported data types	Description
X-Axis	◦ character ◦ datetime ◦ numeric	The field to use as the basis for the chart's horizontal scale.
Y-Axis	◦ character ◦ numeric ◦ datetime	The field to use as the basis for the chart's vertical scale.
Bubble size	numeric	The numeric field to use to determine the size of individual data points. An increase in size indicates an increase in quantity or value. You can select a count of the x-axis field or one of several aggregate values for a different numeric column in the table: ◦ average ◦ sum

Setting	Supported data types	Description
		◦ min ◦ max **Tip** You can control decimals and rounding on numeric data by changing the format of this field in the table view. For help doing this, see "Data formatting options" on page 2658.
Color by optional	character	The field represented by the fourth data dimension to the chart. Adding a fourth data dimension identifies the data points by category. You can think of the **Color by** field as a break field.
Format options	numeric	Select an option in this field to apply formatting such as decimals and rounding to the Y-axis values in the chart. For help doing this, see "Data formatting options" on page 2658. Format options is available only for the **Average** aggregate option. For all other aggregate options, format options set in Table View is applied.

Chart display settings

On the **Configure**  panel, click **Display** and configure the following settings:

Setting	Description
Options	
Show Legend	Show or hide the legend at the top of the chart.
Show Values	Show or hide the data point values.
X-Axis	
Show Label	Show or hide the label for the x-axis.
Y-Axis	
Show Label	Show or hide the label for the left y-axis.
Min	The minimum value to use for the left y-axis. By default, the chart uses the lowest value of the left y-axis data to determine the minimum.
Max	The maximum value to use for the left y-axis. By default, the chart uses the highest value of the left y-axis data to determine the minimum.

Setting	Description
Other settings	
Colors	The colors assigned to each series in the **Color by** dimension.

Line chart

Line charts show trends or changes over time by displaying a series of data points connected by straight line segments. You can display a single data field as multiple lines based on different categories through Color by field.

When do you use it?

Use line charts to track changes over short or long periods of time and to assist in predictive data analysis. If small, frequent changes exist in the series, line charts are more effective than bar charts at visualizing the change over time.

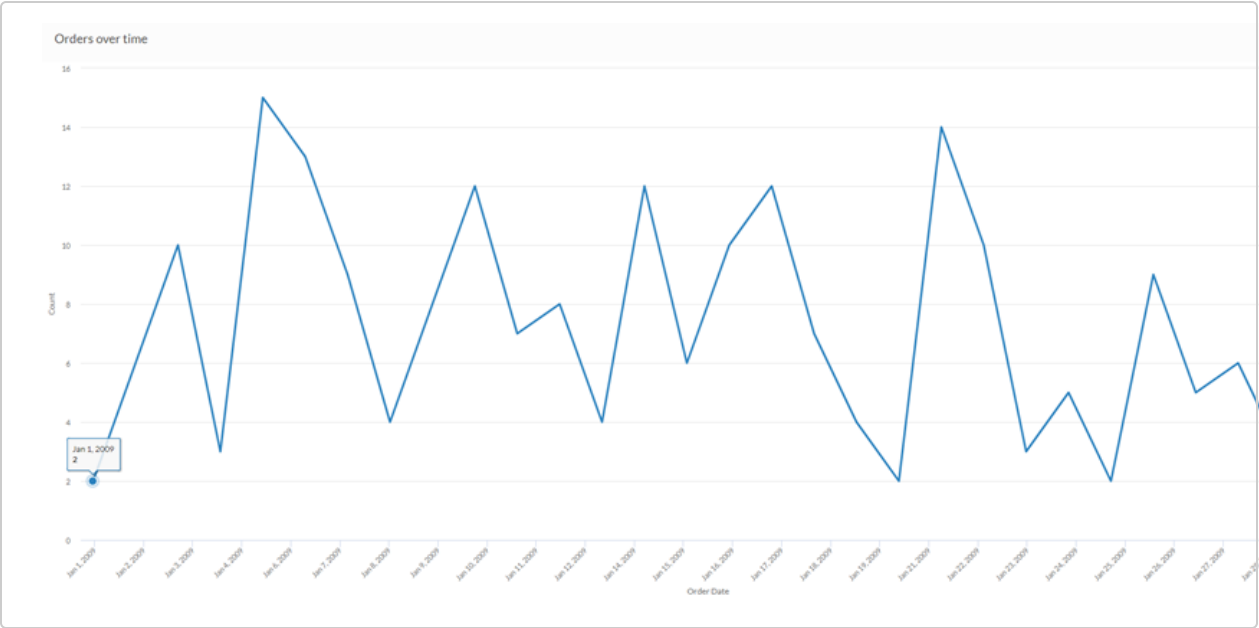
Line charts are also useful for comparing changes over the same period of time for multiple groups or categories.

> **Tip**
> Use independent variables such as time or dates on your line chart's x-axis and dependent numeric variables on the y-axis.

Examples

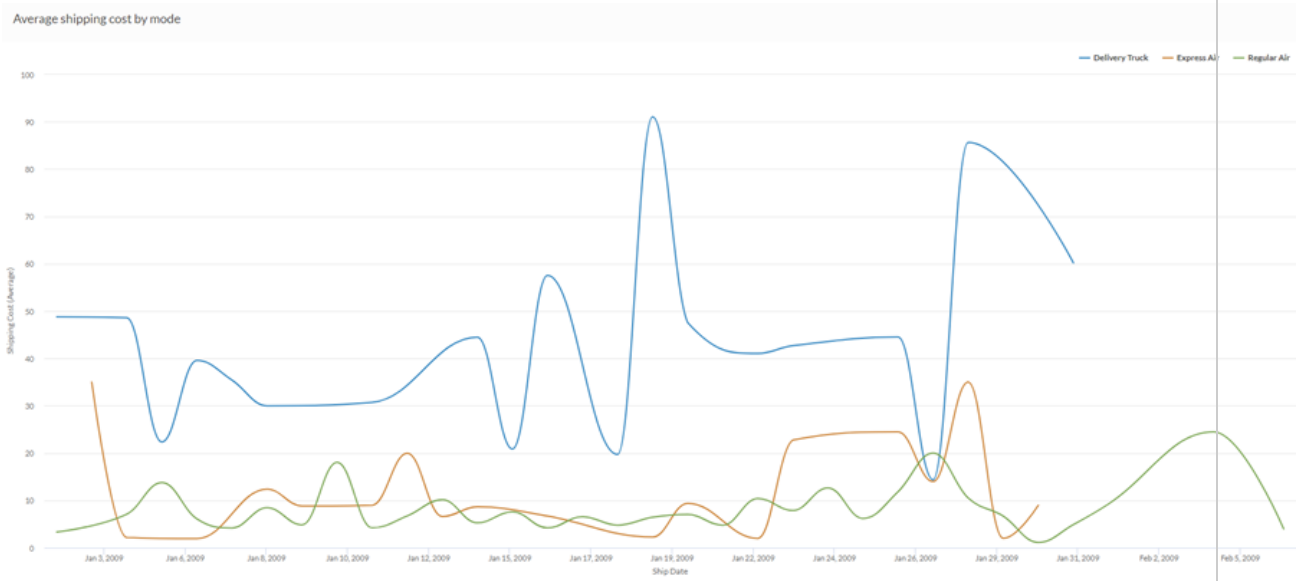
Simple line chart

Using last month's sales data, you want to show the trend of orders placed over time. To do this, you use a line chart that plots the count of orders:



Curved multiseries line chart

Using last month's sales data, you want to show the trend of average shipping cost for each shipping method your company employs. To do this, you use a line chart that groups each series by shipping method:



Data configuration settings

On the **Configure**  panel, click **Data** and configure the following settings:

Setting	Supported data types	Description
X-Axis	◦ character ◦ numeric ◦ datetime	The field to use as the basis for the chart's horizontal scale. Using a character field as the basis for the x-axis allows you to display a line chart that shows X number of comparisons for a given Y value aggregate. This is especially useful when combined with the **Color by** option, which allows you to display separate lines, differentiated by color, for each selected field. Character fields along the x-axis are sorted in alphabetical order by default.
Y-Axis	numeric	The aggregate value represented by the chart's vertical axis. You can select a count of the x-axis field or one of several aggregate values for a different numeric column in the table: ◦ average ◦ sum ◦ min ◦ max The position of data points on the vertical scale determines the height of each line. The height of a line is interpolated or gapped if a data point is missing. **Tip** You can control decimals and rounding on numeric data by changing the format of this field in the table view. For help doing this, see "Data formatting options" on page 2658.
Color by (optional setting)	character	The field represented by the third data dimension to the chart. Adding a third data dimension creates the categories represented by lines. A separate line is created for each unique value in the field.
Format options	numeric	Select an option in this field to apply formatting such as decimals and rounding to the Y-axis values in the chart. For help doing this, see "Data formatting options" on page 2658. Format options is available only for the **Average** aggregate option. For all other aggregate options, format options set in Table View is applied.

Chart display settings

On the **Configure**  panel, click **Display** and configure the following settings:

Setting	Description
Options	
Show Legend	Show or hide the legend at the top of the chart.
Show Values	Show or hide the data point values.
Round Edges	Smooths out the transitions between data points to create a curved line chart.
Interpolate	Handle missing data points by connecting the line using the available data points, but do not plot the missing data point on the x-axis. If disabled, the line is not connected across missing data points.
X-Axis	
Show Label	Show or hide the label for the x-axis.
Y-Axis	
Show Label	Show or hide the label for the left y-axis.
Min	The minimum value to use for the left y-axis. By default, the chart uses the lowest value of the left y-axis data to determine the minimum.
Max	The maximum value to use for the left y-axis. By default, the chart uses the highest value of the left y-axis data to determine the minimum.
Other settings	
Colors	The colors assigned to each series in the **Color by** dimension.

Heat map chart

A heat map is a graphical representation of data where the individual values contained in a matrix are represented as colors. Data points are defined by an x and y axis intersection and a third value that determines the data point's color.

When do you use it?

Use heat maps to compare variables across a large number of categories and to sort complex data by color intensity.

How does it work?

Data values

Data values appear as boxes on the heat map. The size and color of each box are determined by the data for that item:

- **size** – determined by the concentration of x and y axis categories and is not configurable
- **color** – determined by the calculated value specified in the **Color by** setting

Gradients vs stepped colors

By default, the lowest values are colored green, the mid-range values are colored yellow, and the highest values are colored red. Colors are fully customizable.

Data points are colored using one of the following display setting coloring schemes:

- **color gradient** – continuous change from green, through yellow, to red following individual data values
- **stepped color scale** – data values grouped into equidistant categories and displayed using discrete colors

  You can customize the number of steps used to group the data values in chart the display settings.

Examples

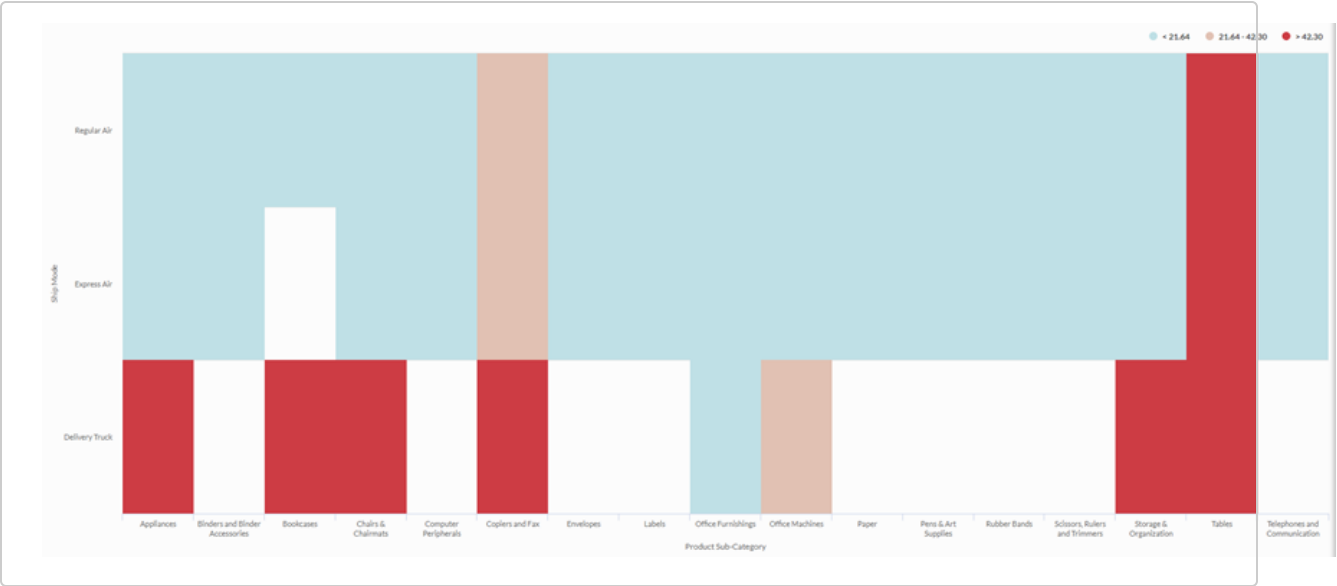
Heat Map chart

Using your company's sales data, you want to show average profit across a large number of categories in two dimensions: region and customer segment. To do this, you create a heat map chart. The average profit value is sorted by color, from blue (low) to red (high):

Stepped Heat map chart

You need to examine the relationship between shipping method and product category for your company. You are looking for a simple analysis that groups average shipping cost into three bins: low, medium, and high.

To visualize this relationship, you create a stepped heat map with three steps. Values are grouped into the corresponding bins based on the step thresholds:



Data configuration settings

On the **Configure** panel, click **Data** and configure the following settings:

Setting	Supported data types	Description
X-Axis	◦ character ◦ numeric ◦ datetime	The field to use as the category on the chart's horizontal scale.
Y-Axis	◦ character ◦ numeric ◦ datetime	The field to use as the category on the chart's vertical scale.
Color by	numeric	The field that determines the color of the box at the intersection of the X and Y axis values: ◦ **Count** - calculates the count of records that match each intersection of X and Y axis values ◦ **Average, Sum, Min, Max** - calculates the selected statistical value for the numeric field selected at each intersection of X and Y values **Tip** You can control decimals and rounding on numeric data by changing the format of this field in the table view. For help doing this, see "Data formatting options" on page 2658.
Format options	numeric	Select an option in this field to apply formatting such as decimals and rounding to the Y-axis values in the chart. For help doing this, see "Data formatting options" on page 2658.

Setting	Supported data types	Description
		Format options is available only for the **Average** aggregate option. For all other aggregate options, format options set in Table View is applied.

Chart display settings

On the **Configure**  panel, click **Display** and configure the following settings:

Setting	Description
Options	
Show Legend	Show or hide the legend at the top of the chart.
Show Values	Show or hide the data point values.
X-Axis	
Show Label	Show or hide the label for the x-axis.
Y-Axis	
Show Label	Show or hide the label for the left y-axis.
Other settings	
Color	The starting, middle, and ending values for the heatmap range. You can specify a color and numeric boundary for each place on the scale.
Stepped Colors	Display the categories as discrete buckets rather than a constant scale. You can specify between 2 and 20 steps.

Statistics

Statistics provide an overview of the data in a table that you can use to identify trends or irregularities.

Data configuration settings

When you configure statistics, you select one or more table fields to use. If any filters are applied to the selected fields, the calculations reflect these filters.

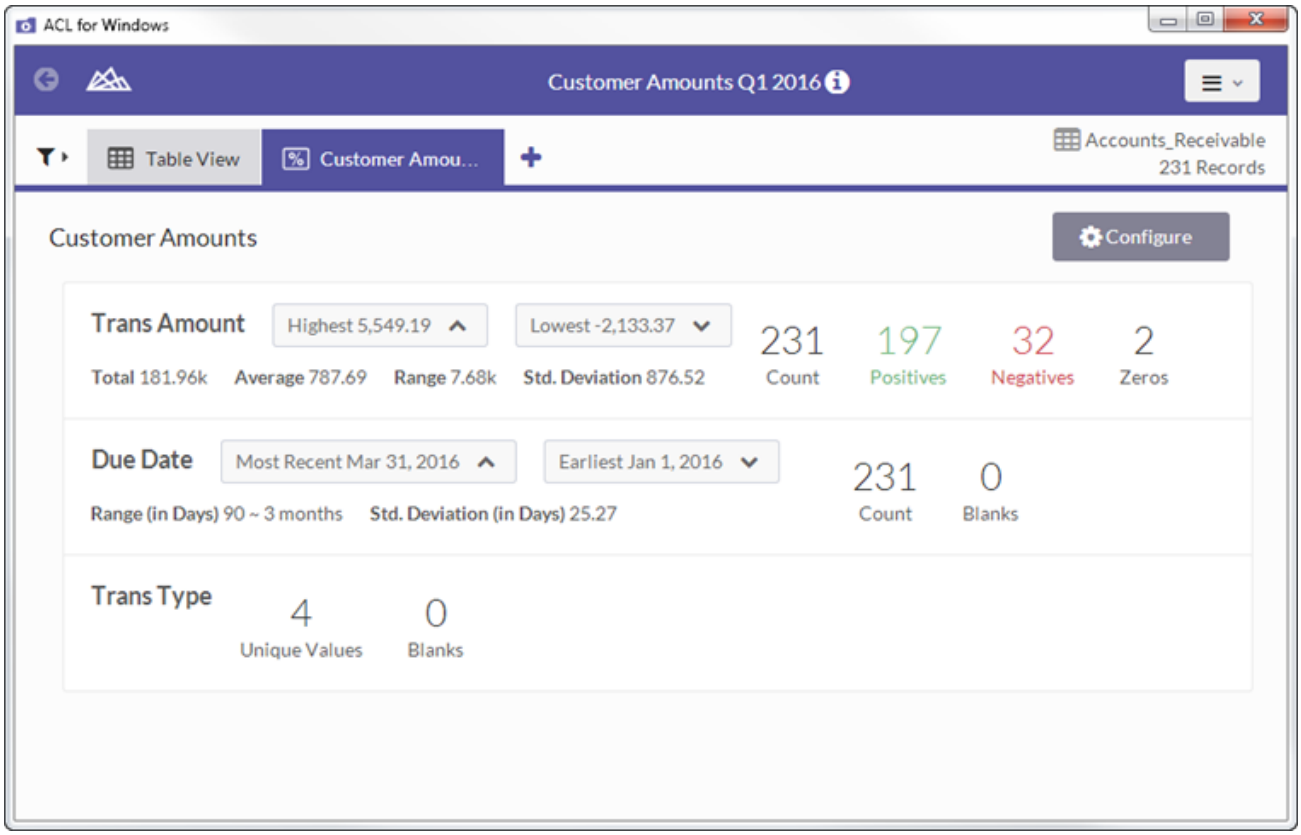
Calculation terms per data type

Data type of field	Term	Definition
Numeric	Count	The total number of records in the current table view
	Positives	The number of positive values in the field, displayed in green
	Negatives	The number of negative values in the field, displayed in red
	Zeros	The number of zero values in the field
	Total	The total of the values in the field
	Average	The average of the values in the field
	Range	The difference between the highest and lowest values in the field
	Standard Deviation	The standard deviation from the average value
	Highest	The five highest values in the field
	Lowest	The five lowest values in the field
Datetime	Range (in days)	The number of days between the most recent date and the earliest date in the field
	Standard deviation (in days)	The standard deviation, measured in days, from the average date
	Count	The number of non-blank dates in the field
	Blanks	The number of blank dates in the field

Data type of field	Term	Definition
	Most Recent	The five most recent dates in the field
	Earliest	The five earliest dates in the field
Character	Unique Values	The number of unique values in the field. One or more blank values count as 1 unique value
	Blanks	The number of blank values in the field

Example

Statistics for numeric, datetime, and character fields



Summary table

Summary tables group records on unique values in one or more key fields and then perform a count for the number of matching records. You can further drill into the summary by selecting a column and producing a crosstab. Once you define the summary, you can also select numeric fields to subtotal.

> **Tip**
> Summary tables most effective when you are comparing a limited number of categories.

When do you use it?

Frequency summaries

Select row key fields only and summarize tables to view the frequency of values across each unique combination of key fields.

Example

Vendor State	Vendor Category	Count
TX	A	3
TX	B	15
CA	A	47
NY	F	2

Cross-tabular summaries

Select row key fields as well as a column and create a crosstab that summarizes the relationship between two categorical variables by depicting the number of times each of the possible category combinations occurred in the data.

Example

Class rank	Performance		
	<= 60% GPA	60% to 79% GPA	>= 80% GPA
Freshman	32	60	15
Sophomore	25	70	23
Junior	40	60	10
Senior	15	45	30

Subtotals

Include values in the summary to find the sum of a numeric field for each of the possible category combinations in your data.

Data configuration settings

Setting	Data types supported	Description
Column	All	The field used to include a second categorical variable for cross-tabular results. You can add one column to the summary table and create a crosstab, or leave the field blank to perform a simple one-dimensional summary.
Rows	All	The fields used to group records. You can add between 0 and *n* rows and each record in the summary table represents a unique combination of the values from the selected fields. **Note** Summary table sorting depends on the order in which you select rows. The first row selection is sorted alphabetically or numerically by the values in the field you select. Within each subsequent selection, the row is sorted inside the preceding row using the same logic.
Values	Numeric	The calculation to perform on the summary table data: ◦ **Count** - the count of records within each group or crosstab cell in the summary table ◦ ***numericField*** - the sum of the selected numeric field for each group or crosstab cell in the summary table You can add between 0 and n value fields to the summary table.

Example

You are performing an analysis of inventory data and you need to review information about unit costs and quantities for different vendors in your vendor list. You are also interested in seeing how these numbers break down across product class and vendor city.

To capture this information in a single view, you create a summary table with the following settings:

- **Column** – Product Class
- **Rows** – Vendor City and Vendor Name
- **Values** – Unit Cost and Quantity on Hand

The table summarizes the vendor data into the crosstab you defined and performs sum calculations for unit cost and quantity values. Rows are sorted first by Vendor City and then by Vendor Name inside each city:

Vendor City	Vendor Name	Product Class				
		03		09		
		Sum of Unit Cost	Sum of Quantity On Hand	Sum of Unit Cost	Sum of Quantity On Hand	
Ann Arbor	Arizona Industries	0	0	2.88	40	
Austin	DIDA Limited	0	0	-6.8	408	
Austin	Global Trade Hardware	1.22	587	0	0	
Austin	Liberty Trading	0	0	0.63	112	
Austin	Miller Lights	0.73	1,478	0	0	
Baton Rouge	Harris Projects	0	0	9	47	
Bay Minette	Larson Supplies	0	0	3	212	
Bellevue	MGMT Mfg.	0	0	21.4	43	
Boise	O'Conner And Daughters	12.5	248	0	0	
Charleston	US Mfg. Corp	0	0	173.8	147	
Charlotte	Lilydale Hardware	4.98	624	0	0	
Chicago	Meridian Industries	4.12	536	0	0	
Des Moines	NOVATECH Wholesale	11.53	700	0	0	
Des Moines	Steel Case Manufacturing	0	0	8.08	200	
Englewood	Adams & Meddick	0	0	0.43	300	
Farmington Hills	Carr International	0	0	4.82	714	
Gibsland	Herbie's Hardware	41.23	600	0	0	

Treemap chart

Treemaps display hierarchical, tree-structured data as a set of nested rectangles. Each group is given a rectangle, which is then tiled with smaller rectangles representing sub-groups. Size and color are used to show separate numeric dimensions of the data.

When do you use it?

Use treemaps when working with large amounts of data that is hierarchically structured. When color and size are correlated, treemaps can help identify patterns that would otherwise be difficult to see.

Treemaps are also effective at legibly displaying large volumes of information in a single screen. Viewers can then drill into a specific category to explore further.

> **Note**
> Treemaps support up to two levels of grouping at this time.

Examples

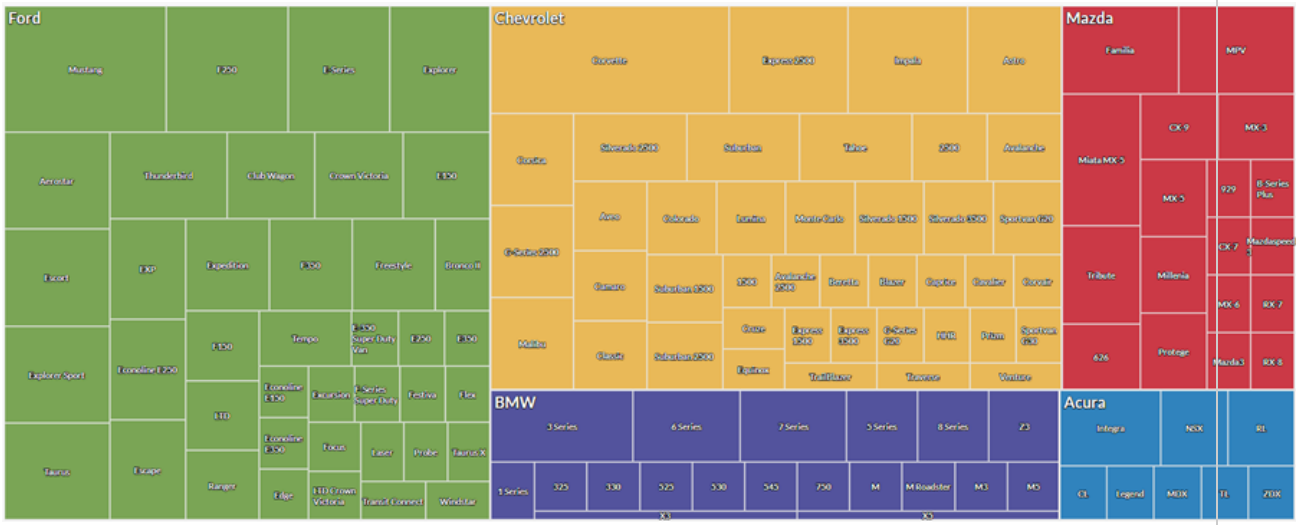
Car makes and models

You have a table that contains an inventory of cars. You want to visualize the count of car models within each make to get an overview of the inventory.

To visualize this data, you use a treemap and:

- group the data by make, and then by model
- set **Size by** to the count of vehicles

Based on the results, you can see how the inventory is distributed across makes and models:



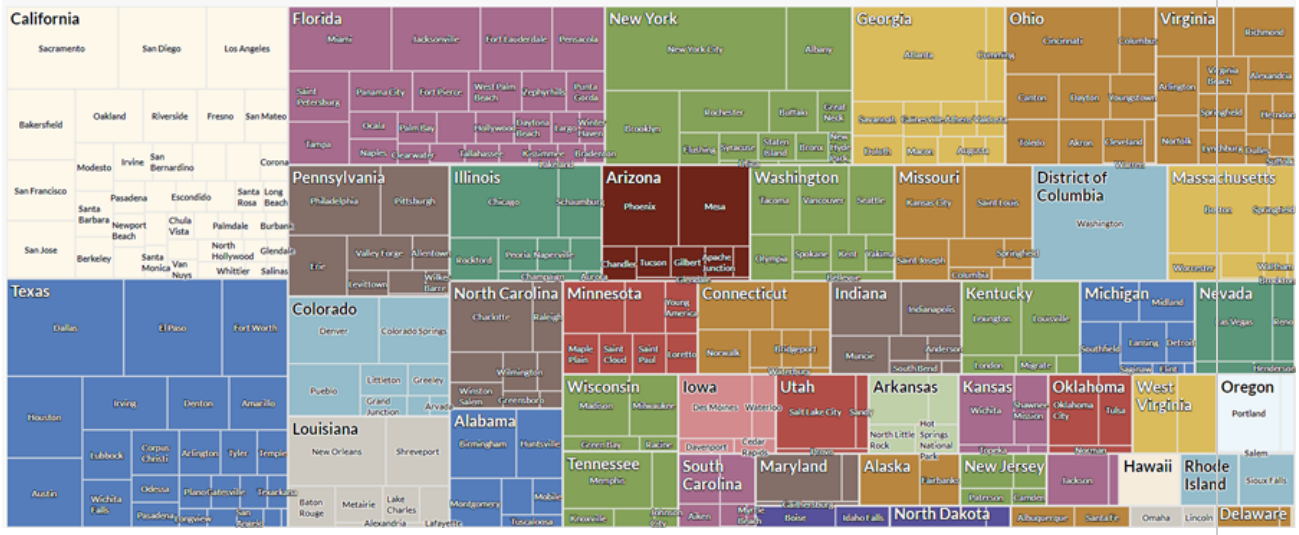
Transaction totals by state and city

You have a table that contains transactions across multiple states and cities in the United States of America. As part of your analysis, you want to visualize the **total transaction amount** by state and by city within each state.

To visualize this data, you use a treemap and:

- group the data by state, and then by city
- set **Size by** to the sum of transactions

Based on the results, you can see a pattern start to emerge for the aggregate transaction amount within these groups:



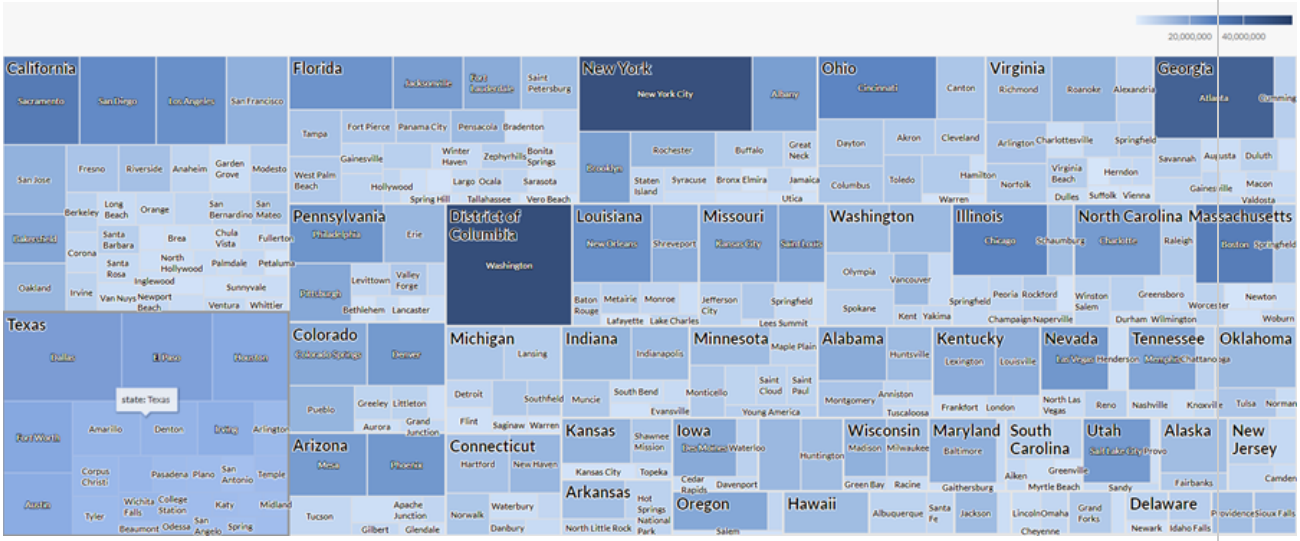
Transaction counts and totals by state and city

You have a table that contains transactions across multiple states and cities in the United States of America. As part of your analysis, you want to visualize the **count of transactions** and the **total transaction amount** by state and by city within each state.

To visualize this data, you use a treemap. You group the data by state, and then by city. You also use the following additional settings:

- **Size by** - count
- **Color by** - sum of transactions

Based on the results, you can determine how transactions are distributed across different city and state combinations, and see a pattern start to emerge for the aggregate transaction amount within these groups:



Data configuration settings

On the **Configure** panel, click **Data** and configure the following settings:

Setting	Supported data types	Description
Group	◦ character ◦ numeric ◦ datetime	The fields to use as categories. The second group you select is nested within the first group. Groups are displayed as rectangles. You can select a maximum of two groups.
Size by	numeric	The aggregate value that determines the size of each group. You can select

Setting	Supported data types	Description
		a count of records or one of several aggregate values for a numeric column in the table: ◦ average ◦ sum ◦ min ◦ max **Tip** You can control decimals and rounding on numeric data by changing the format of this field in the table view. For help doing this, see "Data formatting options" on page 2658.
Color by optional	numeric	The aggregate value that determines the color intensity, or scale, of each group. You can select a count of records or one of several aggregate values for a numeric column in the table: ◦ average ◦ sum ◦ min ◦ max **Tip** You can control decimals and rounding on numeric data by changing the format of this field in the table view. For help doing this, see "Data formatting options" on page 2658.
Format options	numeric	Select an option in this field to apply formatting such as decimals and rounding to the Y-axis values in the chart. For help doing this, see "Data formatting options" on page 2658. Format options is available only for the **Average** aggregate option. For all other aggregate options, format options set in Table View is applied.

Chart display settings

On the **Configure**  panel, click **Display** and configure the following settings:

Setting	Description
Options	
Show Legend	Show or hide the legend at the top of the chart.
Group Labels	
Show First Group	Include labels for values in the first group.

Setting	Description
Show Second Group	Include labels for values in the second group.
Other settings	
Colors	The colors assigned to: ◦ each top level group if no **Color by** selection is made ◦ the range of colors if a **Color by** selection is made Select the starting, middle, and ending values for the range. You can specify a color and numeric boundary for each place on the scale. Use **Stepped Colors** to display discrete buckets for the **Color by** field rather than a constant scale. You can specify between 2 and 20 steps.

Working with Analytics Exchange

> **Note**
> Galvanize will end support for Analytics Exchange on January 1, 2023. Learn more or upgrade to Robots.

You can connect to AX Server from Analytics. The component of AX Server you are connecting to is a Windows service called AX Connector.

To connect, you need to use a server profile, which includes connection information for the server, such as the hostname or IP address for the computer where AX Server is located, and the username and password required to access the server. If Analytics and AX Client are both installed on your computer, the required server profile is automatically created.

You can create as many server profiles as you require, but you can only connect to one server at a time. When a connection is made to a server, that server becomes the active server, and any server tables or local tables that are already open in Analytics are closed.

Analytics automatically connects to AX Server when you:

- Open a table linked to a file on the server.
- Link a table to a file on the server.
- Define a new table from a server profile using the **Data Definition Wizard**.

Analytics automatically disconnects from AX Server when:

- You close and exit Analytics.
- Communications between Analytics and AX Server fail for any reason.
- The connection to AX Server is inactive for a period of time. The server timeout value is configured on the server.

If you want to access database tables using Analytics server tables, you also need to create a database profile, which includes the information required to connect to the database, such as the database type and the user account to use to access the database. Connections to Oracle, SQL Server, and IBM DB2 are supported on Windows using database profiles.

Guidelines for working with server tables

> **Note**
> Galvanize will end support for Analytics Exchange on January 1, 2023. Learn more or upgrade to Robots.

- Analytics projects that contain server tables can be shared by multiple users if all users configure Analytics with the same server profile and database profile information.
- Joins and relations that include more that one Analytics table can only be performed if the Analytics tables are located on the same computer. You cannot join or relate local Analytics tables with server tables.

Enabling server connections

> **Note**
> Galvanize will end support for Analytics Exchange on January 1, 2023. Learn more or upgrade to Robots.

Analytics has a preference setting for turning on or off support for connecting to Galvanize server products. This setting is turned on by default. If it has been turned off, and you require access to this functionality, you can turn it back on by completing the following procedure.

The preference setting controls whether the menu items in the **Server** menu are enabled or not. You need to restart Analytics in order to make this change, so you should save and close any open Analytics projects before you begin.

To enable connections to a server:

1. Select **Tools > Options**.
2. Click the **Interface** tab.
3. Select the **Enable Server integration** checkbox.
4. Click **OK**.
   Analytics displays a message box indicating that you must restart the application before the changes will take effect.
5. Click **OK**.
6. Select **File > Exit** to close Analytics.

   The next time you start Analytics the menu items in the **Server** menu will be enabled.

Server profiles

> **Note**
> Galvanize will end support for Analytics Exchange on January 1, 2023. Learn more or upgrade to Robots.

Server profiles store the information required for Analytics to connect to a server. You need to create a separate server profile for each server that you want to connect to.

You can also create more than one profile for a server, if you need to use different user accounts to access the server for particular data sources.

The Analytics Exchange server profile

If your company is an Analytics Exchange customer, and if Analytics and AX Client are both installed on your computer, the server profile required to connect from Analytics to AX Server is created automatically. After you use AX Client to log in to AX Server for the first time, a profile called "Analytics Exchange" is automatically created in the list of server profiles in Analytics.

Sharing projects that use server profiles

If you want to share Analytics projects that include server tables that reference particular server profiles, each user must create a server profile with the same name and settings.

When you create an Analytics project with tables that are linked to server data, these tables are also linked to the server profile that was used to make the connection. Each user who wants to use this Analytics project must create a server profile with an identical name and settings. The server profile is stored in the Windows registry on each user's computer.

The **Prefix** path specified in the server profile determines the default location where files are created on the server. Each user that specifies a particular **Prefix** location must have write access to that folder.

Create a server profile

Before you create a server profile, you need to get the required server settings from your system administrator.

If more than one user is sharing an Analytics project, you must make sure each of them uses the same profile name, and has the same host and port settings, for each server profile used in the project. Each user can specify their own login credentials in the server profile, but they must have the appropriate security rights assigned.

Show me how

1. Select **Server** > **Server Profiles**.
2. Click **New**.
3. In **Add New Profile**, complete the following steps:
   a. Enter a descriptive name in **Profile Name**.

      The profile name can have up to 50 alphanumeric characters, including spaces. If spaces are used, the profile name must be enclosed in quotation marks if you reference the profile name in an Analytics command.

   b. If you want to copy the settings from an existing profile to the new profile, select the appropriate profile from the **Copy Settings From** drop-down list.
   c. Click **OK**.
4. Enter the connection details for the server by completing the following fields:
   - **Use Integrated Windows Authentication** – If your company is an Analytics Exchange customer, and AX Server is configured to use Integrated Windows Authentication, select this checkbox to use your Windows account information to automatically log in.
   - **User ID** – The user ID required to log on to the server. This text box is disabled if **Use Integrated Windows Authentication** is selected.
   - **Password** – **(Optional)** The password required to log on to the server. This text box is disabled if **Use Integrated Windows Authentication** is selected.

     > **Note**
     > You are not required to save your password with the profile. If you choose not to save your password, Analytics prompts you for it the first time you use the server profile each session. Analytics retains the password for the remainder of the session only.

   - **Prefix** – The file path for your working directory on the server. For example: `C:\ACL\MonthEndAnalysis`
   - **Host Name** – The hostname or IP address of the server to connect to. If you selected the **Use Integrated Windows Authentication** checkbox, you must specify the full hostname for the server. For example: ax.example.com
   - **Port** – The port to use to connect to the server. The default value is 10000.
5. If necessary, select either of the following options:
   - **Encryption** – Encrypts network communications between Analytics and the server using a secure 32-bit encryption algorithm. This option is selected by default, but encrypting all data may noticeably slow down communications. Deselect this option if encrypted communication is not required.
   - **Compression** – Compresses data sent between Analytics and the server. Compressing data speeds up transmission time, even if the data is encrypted. Depending on the type of data being transferred, compression may be as high as 10:1. However a compression ratio of 3:1 is characteristic of most data.
6. Click **Save**.
7. If you entered a password, click **OK** to save the password with the profile.

8. Click **Close**.

   After you complete these steps you can connect to the server. If you are connecting to AX Server, you must create database profiles for any SQL Server, Oracle, or IBM DB2 databases you want to connect to using the server profile you have configured.

Modify a server profile

You can modify an existing server profile whenever settings such as your password or the location of the server change.

Show me how

1. Select **Server > Server Profiles**.
2. Select the server profile you want to modify from the **Profile Name** drop-down list.
3. Update the information in any of the following fields, as necessary:
   - **Use Integrated Windows Authentication** – If your company is an Analytics Exchange customer, and AX Server is configured to use Integrated Windows Authentication, select this checkbox to use your Windows account information to automatically log in.
   - **User ID** – The user ID required to log on to the server. This text box is disabled if **Use Integrated Windows Authentication** is selected.
   - **Password – (Optional)** The password required to log on to the server. This text box is disabled if **Use Integrated Windows Authentication** is selected.

     > **Note**
     > You are not required to save your password with the profile. If you choose not to save your password, Analytics prompts you for it the first time you use the server profile each session. Analytics retains the password for the remainder of the session only.

   - **Prefix** – The file path for your working directory on the server.
   - **Host Name** – The hostname or IP address of the server to connect to. If you selected the **Use Integrated Windows Authentication** checkbox, you must specify the full hostname for the server. For example: ax.example.com
   - **Port** – The port to use to connect to the server. The default value is 10000.
   - **Encryption** – Encrypts network communications between Analytics and the server using a secure 32-bit encryption algorithm.
   - **Compression** – Compresses data sent between Analytics and the server.
4. Click **Save**.
5. If you entered a password, click **OK** to save the password with the profile.
6. Click **Close**.

Delete a server profile

You can delete server profiles you no longer need. However, before you delete a server profile, you should ensure that none of the database profiles associated with the server profile are being used. Any database profiles that have the server profile you are deleting specified in the **Server Profile** drop-down list are also deleted.

If you want to retain a database profile linked to a server profile you are deleting, you need to link the database profile to a different server profile in the **Database Profiles** dialog box.

Show me how

1. Select **Server > Server Profiles**.
2. Select the Server Profile you want to delete from the **Profile Name** drop-down list.
3. Click **Delete**.
4. Click **OK** in the confirmation dialog box.

   The server profile is deleted along with any associated database profiles.

Database profiles

> **Note**
> Galvanize will end support for Analytics Exchange on January 1, 2023. Learn more or upgrade to Robots.

Database profiles are used to connect directly from AX Server to databases running on Microsoft SQL Server, Oracle, or IBM DB2 database servers. Using a database profile to connect to a database allows you to access database tables using the **Data Definition Wizard** in Analytics, or retrieve server data from Analytics Exchange analytics.

Creating database profiles

> **Note**
> Galvanize will end support for Analytics Exchange on January 1, 2023. Learn more or upgrade to Robots.

Database profiles are used to connect AX Server to an Oracle, IBM DB2, or Microsoft SQL Server database. You need to create the server profile that connects Analytics to AX Server before you can create an associated database profile.

The database profile does not need to connect to a database located on the same server as AX Server, but it does need to be accessible from the server where AX Server is installed. For example, when connecting to Microsoft SQL Server, the database engine can be located on any accessible server on the network, but the ODBC connection used by the database profile must be configured on the server where AX Server is installed.

You can create database profiles for any Oracle, DB2, or SQL Server database you want to directly access using AX Server. Direct database access means that each time you open a server table in Analytics a query is run against the database and current data is transferred between AX Server and Analytics, and is then displayed in the active Analytics view for the table. This differs from most other types of data access in Analytics where data is copied into an intermediate data file (.fil) before being displayed in Analytics, and the data file is not updated unless you explicitly refresh the contents of the file.

> **Note**
> Contact your systems administrator or database administrator if you are unsure of the settings you need to enter, or if you are unable to connect to the data source.

To create a database profile:

1. Select **Server > Database Profiles**.
2. Click **New**.
3. In **Add New Profile**, enter a name in **Profile Name**, and optionally select an existing profile in the **Copy Settings From** drop-down list to pre-populate the **Database Profiles** dialog box with settings from the existing profile, and click **OK**.
4. In the **Server Profile** drop-down list, select the server profile you want to associate with the database profile.
5. Select the appropriate database (Oracle, DB2, or SQLServer) from the **Type** drop-down list.
6. Enter the user ID required to access the database in **User ID**.
7. **(Optional)** Enter the password required to access the database in **Password**.
8. Enter the name of the database to connect to in **Service Name**.
9. Click **Save**. If you entered a password, Analytics displays a message box asking you to confirm whether you want to save the password with the profile.
    - To save the password with the profile, click **OK**.
    - To return to the **Profile** dialog box and remove the password, click **Cancel**.

> **Note**
> If you chose not to save your password with the profile, you must enter it each time Analytics attempts to establish a connection to the data source.

10. Click **Close**.

Modifying database profiles

> **Note**
> Galvanize will end support for Analytics Exchange on January 1, 2023. Learn more or upgrade to Robots.

You can modify an existing database profile whenever your settings, such as your password or the name of the database, change.

To modify a database profile:

1. Select **Server > Database Profiles**.
2. Select the database profile you want to modify from the **Profile Name** drop-down list.
3. If the server you want to connect has changed, select the correct server profile from the **Server Profile** drop-down list.
4. Update the information in any of the following fields, as necessary:
   - **Type** – Specify the type of database to connect to.
   - **User ID** – The user account to use to access the database.
   - **Password** – **(Optional)** The password required to access the database.
   - **Service Name** – The name of the database to connect to.
5. Click **Save**. If you entered a password, Analytics displays a message box asking you to confirm whether you want to save the password with the profile. Click **OK** to save the password with the profile.

   > **Note**
   > If you do not save your password with the profile, you will be prompted to enter it each time Analytics attempts to establish a connection to the database.

6. Click **Close**.

Deleting database profiles

> **Note**
> Galvanize will end support for Analytics Exchange on January 1, 2023. Learn more or upgrade to Robots.

You can delete database profiles you no longer need. However, before you delete a database profile, you should ensure that it is no longer being used by tables in any of the projects on the workstation or server.

When you attempt to delete a database profile, Analytics displays a message indicating that deleting the profile will affect tables defined with the profile. The table and the database profile used to create it are linked because the name of the database profile is stored in the table layout. If this profile cannot be found, Analytics prompts the user to select a new profile. If the user selects a profile with more restricted access rights the Analytics table may not display the same results as when the original profile was used.

To delete a database profile:

1. Select **Server > Database Profiles**.
2. Select the database profile you want to delete from the **Profile Name** drop-down list.
3. Click **Delete**.
4. Click **Yes** in the confirmation dialog box.

Verifying database profiles

> **Note**
> Galvanize will end support for Analytics Exchange on January 1, 2023. Learn more or upgrade to Robots.

You can use the **Verify** command to test the connection to the database. Analytics attempts to connect to the database using the settings you have entered for the database profile and displays a message box indicating whether the attempt succeeded or failed. If the connection failed, check to ensure that you have entered all required settings correctly, and then check with your systems administrator to ensure that you have the necessary network and database security rights to access the database.

To verify a database profile:

1. Select **Server > Database Profiles**.
2. Select the database profile you want to verify from the **Profile Name** drop-down list.
3. Click **Verify**.
4. If the **Server Profile Password** dialog box is displayed, enter the password for your server profile and click **OK**.
5. If the **Database Profile Password** dialog box is displayed, enter the password for your database profile and click **OK**.
6. An **ACL Analytics** dialog box is displayed with a message indicating whether the connection to the database succeeded or failed. Click **OK** to close the dialog box.
7. Click **Close**.

Export database profiles

> **Note**
> Galvanize will end support for Analytics Exchange on January 1, 2023. Learn more or upgrade to Robots.

You can export database profiles to a location on the local workstation, or directly to a server. Exported database profiles must be saved with a .dbp file extension.

1. Select **Server > Database Profiles**.
2. Select the database profile you want to export from the **Profile Name** drop-down list.
3. Click **Export**.
4. In **Select File Location**, select one the following options:
   - **Client** – Select this option to export the database profile to a local folder on you computer.
   - **Server** – Select this option, and choose the appropriate server profile from the drop-down list, to export the database profile to a folder on a server. The default folder to export to is the **Prefix** folder specified for the server profile.
5. In **Export Database Profile**, select the directory to export the profile to, enter the filename for the database profile you are exporting with a .DBP file extension (for example, MyProfile.DBP), and then click **Save**.
6. Click **OK** in the confirmation dialog box.

Running analytic scripts that use a database profile

> **Note**
> Galvanize will end support for Analytics Exchange on January 1, 2023. Learn more or upgrade to Robots.

To run analytic scripts on AX Server that connect directly to a database server using a database profile, you must first export the database profile from Analytics and then import the file into the **Related Files** folder of the associated analysis app on AX Server.

Prerequisites

- On AX Server, configure direct database access for the specific database server you are using. For more information, see Direct database access for AX Connector
- In Analytics, create and verify a database server profile. For more information, see:
  - "Creating database profiles" on page 2720
  - "Verifying database profiles" on page 2724

Export the database server profile from Analytics

1. In Analytics, select **Server > Database Profiles**.
2. From the **Profile Name** list, select the correct database profile and click **Export**.
3. In the Select File Location dialog box, select **Client** and click **OK**.
4. In the **Export Database Profile** dialog box, select the directory to export the profile to, enter the filename for the database profile you are exporting, and then click **Save**.
5. In the confirmation dialog box, click **OK**.

Import the database server profile (.DBP file) into AX Server

1. In AX Client, in the **Server Explorer**, open the folder that contains the analytic script, right-click the **Related Files** sub-folder, and then click **Import**.
2. In the **Select the Files to Import** dialog box, navigate to the exported `.DBP` file on your local

computer and select **Open**.
3. Click **Import**.

Declare the //FILE tag for the imported database profile

In the analytic header of the script that connects to the database, include the `//FILE` tag in the following format:

```
//FILE databaseProfileFileName.dbp
```

What happens next?

When you run the analytic script on AX Server, the script can now use the database profile information stored in the **Related Files** folder to connect to your database. If the database profile information changes in the future, you must complete this procedure again and overwrite the `.DBP` file.

Connecting to AX Server

> **Note**
>
> Galvanize will end support for Analytics Exchange on January 1, 2023. Learn more or upgrade to Robots.

In most cases, Analytics connects to AX Server as required. For example, when you open a server table, Analytics uses the server table and database profile used to define the table to connect to AX Server and open the table. You can also connect to AX Server manually whenever necessary.

To connect to AX Server:

1. Select **Server > Connect**.
2. In the **Server Profiles** dialog box, select the server you want to connect to from the **Profile Name** drop-down list.
3. Click **Connect**.
4. If you did not save a password with your profile, Analytics prompts you to enter a password. If required, enter your password in the **Server Profile Password** dialog box and click **OK**.

   While the connection is being established, the Server Activity dialog is displayed listing the progress of you connection and any errors encountered. Once the connection is established, the **Server Activity** log closes, and Analytics displays the connection status in the lower left corner of the **Server Profiles** dialog box.

Modifying Analytics server table queries

> **Note**
> Galvanize will end support for Analytics Exchange on January 1, 2023. Learn more or upgrade to Robots.

You can modify the WHERE and ORDER clauses for Analytics server tables when you need to adjust the query for a table defined from an Oracle, IBM DB2, or SQL Server database using a database profile. This means that you can change the data retrieved for the Analytics server table, without needing to redefine the table in the **Data Definition Wizard**.

To modify the query for a table:

1. If the table you want to edit the query for is open, you must close it before you can edit the query. To close a table, right-click the table in the **Overview** tab in the **Navigator** and select **Close Table**.
2. Right-click the table in the **Overview** tab in the **Navigator** and select **Properties**.
3. In the**Table Properties** dialog box, click the **Edit Query** tab.
4. Make any necessary changes in the **WHERE clause** and **ORDER clause** text boxes and click **OK**.

   The next time you open the table the modified query will be used to select and order data from the database table(s).

Disconnect from a server

> **Note**
> Galvanize will end support for Analytics Exchange on January 1, 2023. Learn more or upgrade to Robots.

You can manually disconnect from a server whenever necessary. If the **Server Profiles** dialog box is open, you can click the **Disconnect** button to close your connection. Otherwise, you need to use the **Disconnect** menu option as described below.

1. From the Analytics main menu, select **Server > Disconnect**.

   If you are currently connected to a server the **Disconnect** dialog box displays the name of the server profile that is connected.
2. Click **Disconnect** to close the connection.
3. Click **Close**.

Viewing server activity

> **Note**
> Galvanize will end support for Analytics Exchange on January 1, 2023. Learn more or upgrade to Robots.

The **Server Activity** dialog box displays information about connections between Analytics and a server.

- **Server Results tab** – displays the details of connection attempts between Analytics and a server

  You can view the log entries to confirm that the connection was established, or to identify exactly where the error occurred if the connection failed.

- **Connection and CPU time tab** – displays the duration of connections to a server and the amount of server processing time allocated to the connection

  This information can be useful for determining how long it takes to run a command, script, or analytic, and the server processing resources required.

View the activity log

1. From the Analytics main menu, select **Server > Activity Log**.
2. Click **Close** when you have finished viewing the log information.

This page intentionally left blank

Glossary

Glossary of Galvanize product terms

The definitions in this glossary reflect the way the terms are used in Galvanize products, which in some cases may differ from the meaning of the terms in other contexts.

A

Academy
A Galvanize training resource that contains online courses, training content, and how-to tutorials.

ACL connector
An ODBC driver, bundled with Analytics, that provides an interface to an ODBC-compliant database (cloud or on-premise) or a file format. Also known as a data connector.

ACL Connector for Analytics
An ODBC driver, bundled with Analytics, that allows third-party reporting applications to access Analytics data files.

ACL Desktop
The former name of Analytics.

ACL Essentials
A suite of premium analytics that continuously assess risk in critical ERP processes.

ACL for Windows
A product that provides access to Analytics, the Analysis App window, Offline Projects, Robots, and Results. Access to each component is determined by subscription type.

ACL Network Edition
A discontinued licensing arrangement that allowed concurrent usage of a single Analytics installation.

ACL Robotics
A Galvanize product that enables you to automate repetitive but critical tasks.

ACLScript
The scripting language that forms the basis of data analysis in Analytics.

action
A feature in Projects used to remediate an issues related to a particular business area. Also known as remediation and recommendation.

activation
An action involved in completing the installation of Analytics or Analytics Exchange that validates the license(s) used to access the product.

active table
In multiple-table mode in Analytics, the table that is currently selected and displaying data.

active view
For Analytics tables with more than one view, the view that is currently selected and displaying data.

actual risk
A calculation in Projects that determines the risk that remains after controls are taken into account. Also known as net risk and residual risk.

ad hoc analytic job
An instance of an analytic script execution that is initiated by a user in real time rather than by a schedule.

ad hoc expression
An expression in Analytics that is used once and not named or saved for reuse.

ad hoc filter
A filter in Analytics that is used once and not named or saved for reuse.

Add-In for Excel
A Microsoft plug-in that adds Analytics data investigation functionality, and integration with the HighBond platform, to Microsoft Excel.

age
(1) An ACL command that groups the records in a table into aging periods, measured in days. ACLScript syntax: AGE (2) A calculation in Results that shows the average number of days that pass before a record is closed.

aging periods
User-specified date ranges for grouping records.

alternate column title
The name that appears on a column header in an Analytics view for display purposes, which can differ from the physical field name in the table layout. Also known as display name.

analysis app
An Analytics project containing one or more analytic scripts and associated data tables that is packaged for use in Analytics Exchange or the Analysis App window.

Analysis App window
A freestanding component of Analytics that guides a user through running analytics, and provides advanced data interpretations and visualizations based on analytic results.

analytic
An Analytics script with an analytic header that allows the script to run in Robots, Analytics Exchange, or the Analysis App window. Also known as an analytic script.

analytic chain
A series of analytic scripts that are scheduled to execute in sequence on AX Server in the order specified by the user.

analytic header
A series of analytic tags at the top of an Analytics script that allows the script to function as an analytic.

analytic input
A variable that gets defined when running or scheduling an analytic script and then used during the execution of the script.

analytic input set
A group of analytic input values that are saved for a particular analytic script and can then be selected to use when running or scheduling the script.

analytic job
A single instance of an analytic script execution, either scheduled or ad hoc, that contains its own results data, log information, and unique identifier.

analytic results
Any output tables, output files, or log files that are generated when an analytic script runs on AX Server. Also known as results.

analytic script
An Analytics script with an analytic header that allows the script to run in Robots, Analytics Exchange, or the Analysis App window. Also known as an analytic.

analytic tag
A declarative command inside the analytic header that defines inputs, outputs, and instructions for an analytic script.

Analytics
An application that provides a combination of data access, data analysis, and integrated reporting capabilities.

Analytics Exchange
A client-server platform that extends the functionality of Analytics to support a full range of processes from data analysis to continuous monitoring.

Analytics folder
A second-level organizational container in Analytics, similar to a Windows folder, but without any existence outside an Analytics project. Also known as a project folder.

Analytics project
The top-level organizational container in Analytics, which stores data analysis information, but does not store any source data.

Analytics project file
An Analytics application file, with an .acl extension, that stores an Analytics project.

Analytics working directory
The user-specified default location for saving Analytics projects and output tables.

append
(1) An ACL command that combines two or more Analytics tables by appending them in a new table. ACLScript syntax: APPEND (2) An option for some ACL commands that adds the output records as a group to the end of an existing Analytics table. ACLScript syntax: APPEND

archive
(1) A feature in Projects that allows you to move a project from an active to a read-only state. (2) A feature in Analytics Exchange that compresses a collection into a restorable .zip file for long-term storage after an audit completes.

argument
The actual value input into a function or command when the function or command is used within a script. Each time that a function is used, different arguments can be provided. For example, if a script uses TRIM("test "), then "test " is the argument that is provided for that specific usage of the function TRIM( ).

ASCII (American Standard Code for Information Interchange)
A character encoding standard that represents a sequence of digits as a single character. Most modern character-encoding schemes are based on ASCII, although they support many additional characters. ASCII is widely used for storing data in English and Western European languages.

assessment driver
A HighBond automation tool that allows you to keep your risk and control assessments current, in real-time.

assessment survey
A questionnaire from Results that has been connected to an assessment in Projects.

assurance
(1) A calculation in Projects and Strategy that assures the controls in place effectively mitigate risks. (2) A calculation in Projects that represents your organization's confidence in requirements being met.

Assurance Plans
A HighBond app that allows you to determine the scope of assurance activities, the areas of coverage, and the availability of resources. Also known as Audit Plans.

Audit Command Language
The scripting language, also called ACLScript, that forms the basis of data analysis in Analytics. What "ACL" stands for.

AuditBond
An audit management solution in HighBond that helps organizations improve efficiency across their entire audit workflow, from planning to reporting.

AX Client
A Windows client application used to interact with audit content and manage users on AX Server.

AX Engine Node
An optional instance of the software that executes analytic scripts, which is installed on a machine other than AX Server to increase the resources available for running scripts. Also known as engine node.

AX Server
The application server component of Analytics Exchange that handles all requests from client applications.

AX Server Configuration web application
A web application used by administrators to configure the server settings in Analytics Exchange.

AX Web Client
A web application used by non-technical specialists such as staff auditors and audit executives to view and interact with Working directory content on AX Server.

B

backquote

A special character (`) that qualifies data as a date, datetime, or time value: `20160101 22:30:30`

Benford analysis

An ACL command that counts the number of times each leading digit or digit combination occurs in a field, and compares the actual count to the expected count (which is calculated using the Benford formula). ACLScript syntax: BENFORD

boolean

See logical.

byte

A unit of digital information that most commonly consists of eight bits. Bytes are the unit of measurement for all data storage, and since bytes are so small, they are often used to measure specific data in a file, such as characters. Historically, the byte was the number of bits used to encode a single character of text in a computer and for this reason it is the smallest addressable unit of memory in many computer architectures.

C

cell selection method

In Analytics sampling, a method of record selection in which each selected record is randomly chosen from within identically sized cells or blocks of records.

certainty stratum

In Analytics classical variables sampling, an option that allows you to include in the sample all records that are greater than or equal to a specified numeric threshold.

certificate authority

An independent, trustworthy, third-party entity that issues digital security certificates that validate the authenticity of connections to AX Server.

character

1) A unit of information that corresponds with a symbol such as a letter in an alphabet. Examples of characters include letters, numerical digits, punctuation marks, and whitespace.
2) A data type that represents a series of one or more characters, either as a literal or as some kind of variable. This data type is also commonly referred to as a string.

check in
An option in Offline Projects and HighBond for iOS and Android that enables users to sync back a section to Projects when they have returned to an online environment.

check out
An option in Offline Projects and HighBond for iOS and Android that enables users to work with a Projects section while in a remote or offline environment.

citation mode
A feature in Projects that allows you to cite evidence by highlighting and linking text to files.

classical variables sampling
One of three types of sampling in Analytics, appropriate if you are interested in the total audited value of a file, or the total amount of monetary misstatement.

classify
An ACL command that groups the records in a table based on identical values in a character or numeric field. ACLScript syntax: CLASSIFY

cluster
An ACL command that groups the records in a table based on similar, or nearby, values in a numeric field. Clustering uses the K-means clustering algorithm, which is a popular machine learning algorithm. ACLScript syntax: CLUSTER

collection
(1) A container in Results used to organize analyses that relate to different departments, business processes, or data sets. (2) A top-level container with specific user permissions in Analytics Exchange that organizes content within either the Working directory or the Library.

column width
The number of characters in a column in an Analytics view, which is not the same as the field length in a table layout.

command
A computerized routine in Analytics, often broader in scope, that performs an operation on data or a maintenance task and, depending on the command, outputs results. Every line in a script starts with an ACLScript command, followed by one or more parameters. For example, CLASSIFY ON Customer_Number SUBTOTAL Trans_Amount TO "Customer_total.FIL" instructs Analytics to CLASSIFY and specifies the parameters and values to use.

command filter
A filter that applies to a single execution of an ACL command, restricting which records in a table the command processes. Also known as a local filter.

command line
A feature in Analytics that allows you to enter and process ACLScript commands one at a time.

command log
A component in Analytics that records every command executed. Also known as the log.

comment
A user-readable explanation or annotation in the source code of a script. Comments are added with the purpose of making the source code easier for anyone who may try to read, use, or understand a script, and they are ignored by the script engine when running the script. They are formatted as either block comments or inline comments.

Community
A site that allows Galvanize users to connect with each other, access content resources, ask questions, and provide product feedback.

Compliance Maps
A HighBond app that centralizes the documentation of regulatory requirements and mapped controls.

ComplianceBond
A compliance management solution in HighBond that helps organizations implement, automate, and demonstrate assurance over a compliance program.

computed field
In an Analytics table, a field created using an expression, which displays a calculated result for each record. Compare with physical field.

conditional computed field
In an Analytics table, a field created using multiple, condition-based expressions, which displays a calculated result for each record.

confidence
In Analytics sampling, the user-specified degree of certainty that a sample of records is representative of the entire population. Also known as reliability.

Content & Intelligence Gallery
A central repository for industry-specific content that can be used in Galvanize products.

control
A feature in Projects that is used to provide assurance in the achievement of an organization's objectives, and in mitigating a risk. Also known as procedure.

control design
The process of building a control that is intended to mitigate a risk, assessed using a walk-through in Projects.

control flow
The order in which individual commands, functions, or expressions are executed or evaluated. Control flow is primarily associated with conditional statements, which allow different actions depending on whether a condition evaluates to true or false.

control performance
The assurance that control activities are being performed consistently.

control performers
The frontline staff that need to perform control activities consistently.

control tasks
The occurrences of control activities previously performed and currently being performed.

control weight
A calculation in Projects that expresses the percentage of a risk or requirement that a control covers.

Control X-Ray
A feature in Projects that provides additional context about a control that is being tested.

ControlsBond
An internal controls management solution in HighBond that helps organizations manage and automate their internal controls program.

count
An ACL command that counts the total number or the filtered number of records in a table. ACLScript syntax: COUNT

cross-tabulate
An ACL command that groups the records in a table by arranging character fields in rows and columns, similar to an Excel pivot table. ACLScript syntax: CROSSTAB

custom dialog box
A dialog box created by an Analytics script writer that provides user interaction or feedback when an Analytics script is running.

D

Data Access window

A component in Analytics that lets you import and shape data from a wide range of ODBC-compliant data sources.

data analytic

A table in Results that is used to store data that is imported from a file or from Analytics.

data category, data type

Four classifications of data that can be contained and analyzed in Analytics: character, numeric, datetime, and logical. The classification controls how Analytics interprets and uses the data. For example, the value 123 can be defined as character data ("123"), or numeric data (123). Different Analytics operations are available depending on how the data is classified. The character and numeric data categories contain several data types. "Data type" is often used interchangeably with "data category". However, strictly speaking, data category refers to the sort of data that can be input to an Analytics command or function. Data type refers to the storage format of data.

data connector

An ODBC driver, bundled in Analytics, that provides an interface to an ODBC-compliant database (cloud or on-premise) or a file format. Also known as an ACL connector.

Data Definition Wizard

A sequential, multi-page dialog box in which you specify how Analytics reads source data, and in some cases import the data to a newly created Analytics table.

data format

The characteristics of values in a field, such as justification, case, and date format.

data structure

The characteristics of records in a table, such as the number and order of the fields, and the field length and data type. Also known as record structure.

data type, data category

Four classifications of data that can be contained and analyzed in Analytics: character, numeric, datetime, and logical. The classification controls how Analytics interprets and uses the data. For example, the value 123 can be defined as character data ("123"), or numeric data (123). Different Analytics operations are available depending on how the data is classified. The character and numeric data categories contain several data types. "Data type" is often used interchangeably with "data category". However, strictly speaking, data category refers to the sort of data that can be input to an Analytics command or function. Data type refers to the storage format of data.

datetime
A data type that represents an instant in time, typically expressed as a date and time of day.

default view
The Analytics table view that is automatically created when you create an Analytics table.

define
The process of specifying information about the structure and characteristics of source data so that Analytics can read it.

Desktop
The former name of Analytics.

Dialog Builder
A component in Analytics for creating custom dialog boxes that provide user interaction or feedback when an Analytics script is running.

Dice's coefficient
An algorithm that measures the similarity of two character strings. In Analytics, available as a function, and included in the fuzzy join command.

difference estimation
In Analytics classical variables sampling, one of the methods available for projecting results to an entire population.

Direct Link
An optional utility, integrated with Analytics or Analytics Exchange, that allows you to connect to your SAP system and extract data.

duplicates
An ACL command that detects whether duplicate values (identical values) or entire duplicate records exist in an Analytics table. ACLScript syntax: DUPLICATES

E

engine node
An optional instance of the software that executes analytic scripts, which is installed on a machine other than AX Server to increase the resources available for running scripts. Also known as AX Engine Node.

entity
A organizational feature in Strategy and Projects that may relate to business units, departments, or locations.

error
In Analytics sampling, a monetary misstatement, or a deviation from a control.

evaluate
An ACL command that projects errors you find in sampled data to the entire population. ACLScript syntax: EVALUATE, CVSEVALUATE

event report
A table in Results that stores responses to questionnaires deployed as permanent, anonymous web forms that anyone can use to notify stakeholders of events or incidents.

exact character comparisons
A option in Analytics that specifies whether character values are compared using the shorter or the longer of two comparison strings.

exception
A potential problem that is identified in an audit and followed up on.

export
An ACL command that exports data in a specified file format to a specified location. ACLScript syntax: EXPORT

expression
A combination of values and operators that performs a calculation and returns a result.

Expression Builder
A component in Analytics that allows you to use the mouse to create ACL expressions, rather than typing expression syntax manually.

extract
An ACL command that extracts (copies) records or fields from an Analytics table and outputs them to a new Analytics table. ACLScript syntax: EXTRACT

extract and append
An ACL command that combines two Analytics tables by copying records from one of the tables and adding them as a group to the end of the other table.

F

factory settings
The default settings for configurable options in Analytics, which can be reset with a button click.

field

In an Analytics table layout, a single unit of data, such as employee ID, that together with other units of data form a record.

field definition

The metadata such as name, data type, start position, and length that designates a unit of raw source data as a physical field in an Analytics table layout.

field length

The number of characters in a physical field in an Analytics table layout, which is not the same as the display width of a column in a view.

field name

The name of a field in an Analytics table layout, which can differ from the alternate column title that appears in a view.

fieldwork

A process of gathering, analyzing, and evaluating evidence to document observations and provide recommendations to improve business efficiency. Also known as risk categories, processes, sections, objectives.

fil file

The native format for a data file in Analytics (.fil).

filter

A feature in Analytics that creates a subset of table data based on a test or condition that you specify.

filter box

A feature in Analytics that allows you to create and apply filters to table data. Also known as filter line.

filter history

A feature in Analytics that saves previously created filters and allows you to easily reapply them.

filter line

A feature in Analytics that allows you to create and apply filters to table data. Also known as filter box.

fixed interval selection method

In Analytics sampling, a method of record selection in which each selected record is a fixed interval or distance apart.

folder
A container with specific user permissions in Analytics Exchange that organizes content within collections.

framework
A structured set of information that can be used to build projects.

Frameworks
A HighBond app that allows you to maintain a structured set of information, used for building projects.

FraudBond
A fraud management solution in HighBond that helps organizations manage anti-fraud or anti-bribery programs from detection through to case resolution.

function
A computerized routine in Analytics, narrow in scope, that performs a specific task or calculation and returns a single value or "result". Functions are like an "opaque box". The script writer does not know how the computerized routine works, just the expected input and output. For example, to convert the numeric value 22 into the character string "22", the script writer provides the numeric value as input for the STRING( ) function. When the function runs, it accepts 22 as input and returns "22" as output. It is not necessary, or even possible, for the script writer to know how the function converts the number to a character value. It is only necessary to know the expected input and output.

fuzzy duplicates
An ACL command that detects nearly identical values in a character field. ACLScript syntax: FUZZYDUP

fuzzy join
An ACL command that uses fuzzy matching to combine two Analytics tables with different record structures into a new Analytics table that contains any combination of fields from the two original tables. ACLScript syntax: FUZZYJOIN

G

gaps
An ACL command that detects whether a numeric or datetime field contains one or more gaps in its sequence. ACLScript syntax: GAPS

global filter
(1) A filter that applies to an Analytics table view and restricts which records are displayed, or processed. Also known as a view filter. (2) A Results storyboard filter applied across multiple source tables based on fields they have in common.

group
An ACL command that allows a script to execute a sequence of commands on a record before moving to the next record in a table. ACLScript syntax: GROUP

H

harmonize
The process of making the structure or format of corresponding fields in separate tables identical.

HighBond
Galvanize's enterprise governance software platform that creates stronger security, risk management, compliance, and assurance.

histogram
An ACL command that groups the records in a table and displays the groups in a vertical bar chart. ACLScript syntax: HISTOGRAM

I

Impact Reports
A subscription-based service that allows you to define a one-click report template, tailored to your needs.

import
An ACL command that specifies how to interpret records and fields in source data and imports the data to an Analytics table. ACLScript syntax: IMPORT

Import Wizard
An informal name for the Data Definition Wizard.

index
An ACL command that presents the records in an Analytics table in a sequential order without actually physically reordering the source data. ACLScript syntax: INDEX

index file
An Analytics application file, with an .inx extension, that stores the pointers required to index an Analytics table.

inherent risk
(1) A calculation in Strategy derived from dividing inherent risk by the total possible inherent risk score across all operating segments specified in your Strategy Map. (2) A calculation in Projects derived from multiplying all risk scoring factors together.

inherent risk heat

A calculation in Strategy derived from dividing inherent risk by the total possible inherent risk score across all operating segments specified in your Strategy Map.

Inspirations

A catalog of risk scenarios and tests that are categorized by process and industry, and designed to measure and monitor process integrity.

instance

An instance is any HighBond environment. Your company can have a single instance, or multiple instances that reflect different divisions, operating units, and user needs.

internal control

A workflow in Projects where a narrative is prepared to gain understanding of the project's goals, key controls within the process are identified, walkthroughs are performed to verify controls are designed appropriately, and control testing is performed to verify controls are operating effectively.

Interpretation

A bundled collection of filters, visualizations, and statistics based on a results table generated by an analytic.

interval

In Analytics sampling, the distance between the monetary units or the records selected in the sample. Also known as sampling interval.

issue

A problem or exception that has been identified within a project in Projects. Also known as deficiency, observation, finding.

Issue Tracker

A HighBond app that tracks issues across your entire organization and allows you to create customized issue reports.

J

join

An ACL command that combines two Analytics tables with different record structures into a new Analytics table that contains any combination of fields from the two original tables. ACLScript syntax: JOIN

K

keyword

A word that has special meaning in a particular context of a script. For example, the keyword "if" is interpreted as a command name if it is the first word on a line of script, while it is interpreted as a a parameter name when it follows a command on the same line.

L

Launcher

A simple window that serves as a starting point for working in Analytics, the Analysis App window, or Offline Projects.

Launchpad

A single web application for accessing Galvanize applications and resources.

layout

The part of an Analytics table that contains metadata for interpreting the source data, and specifying its location. Also known as a table layout.

length

The number of characters in a physical field in an Analytics table layout, which is not the same as the display width of a column in a view. Also known as field length.

Levenshtein distance

An algorithm that measures the difference of two character strings. In Analytics, available as a function, and included in the fuzzy duplicates and fuzzy join commands.

Library

A directory in Analytics Exchange that stores content available to administrators only unless specific user access is granted.

linked layout

A read-only shortcut to a master layout on AX Server.

linked table

A read-only shortcut to a master table on AX Server that describes the data from the master table without connecting directly to the master table's source data file.

literal

A notation for representing a fixed value in source code that may be used to represent character, numeric, logical, or datetime data. Literals are contrasted with variables, and are often

used to assign values to variables. For example, "word" is a character literal while v_char = "word" assigns the literal's value to the variable v_char.

local filter

(1) A filter that applies to a single execution of an ACL command, restricting which records in a table the command processes. Also known as command filter. (2) A filter applied inside an interpretation or metric configuration in Results.

log

A component in Analytics that records every command executed. Also known as the command log.

logical

The simplest data type. Logical data expresses a truth value of either true or false and is also known as boolean data. This data type is primarily associated with conditional statements, which allow different actions and change control flow depending on whether a condition evaluates to true or false.

loop

An ACL command, used inside the Group command, that allows a script to execute a sequence of commands repeatedly on a record before moving to the next record in a table. ACLScript syntax: LOOP

M

many-to-many join

A type of ACL join in which all occurrences of a matching primary key value are joined to all occurrences of a matching secondary key value.

many-to-one join

A type of ACL join in which all occurrences of a matching primary key value are joined to only the first occurrence of a matching secondary key value.

master layout

A standalone table layout on AX Server that is linked to by one or more linked layouts.

master table

A standalone table on AX Server that is linked to by one or more linked tables.

materiality

In Analytics sampling, the amount of monetary misstatement that you are willing to accept in a file. Also known as tolerable misstatement.

mean-per-unit estimation
In Analytics classical variables sampling, one of the methods available for projecting results to an entire population.

merge
An ACL command that combines two sorted Analytics tables into a new Analytics table that uses the same sort order as the original tables. ACLScript syntax: MERGE

Mission Control
A HighBond app that presents control information from Projects in a simplified and centralized view.

misstatement
In Analytics sampling, a monetary amount that is inaccurately recorded or reported.

monetary unit sampling
One of three types of sampling in Analytics, appropriate if you are interested in the total amount of monetary misstatement in a file. Also known as dollar-unit sampling, probability-proportional-to-size sampling.

N

named expression
An expression in Analytics that is named and saved for reuse.

named filter
A filter in Analytics that is named and saved for reuse.

narrative
A description of a business process or area under review. Also known as policy, process description, control guide.

Navigator
A feature in Analytics that uses a tree structure to organize Analytics project content.

numeric
A data type that represents values containing digits from 0 to 9 and, optionally, a negative sign and a decimal point.

O

objective
The key goals of a project of framework, and the organizing containers for work done within a project or framework in Projects. Also known as risk category, process, section, cycle, control objective.

Offline Projects
A component of ACL for Windows that allows users to use Projects without an internet connection.

operand
A literal value, function, variable, or expression that is supplied as input to an operator. For example, in an simple addition operation, two integers are supplied to the addition operator '+' as operands: 2 + 5.

operating segment
A region, business unit, division, location, or entity that you assess strategic risks across in Strategy.

operator
A symbol such as '+' that performs an operation on one or more operands in an expression. Operators can be unary, where the operation uses only one operand: -5. Or they can be binary, where the operation uses two operands: amount < 50.

options
Configurable settings that control Analytics behavior.

Options dialog box
A multi-tabbed dialog box in Analytics that contains configurable settings.

outliers
An ACL command that identifies statistical outliers in a numeric field. ACLScript syntax: OUTLIERS

output table
An Analytics table containing the output results of an ACL command.

owner
A person that is assigned responsibility for a process, control, or issue in Projects.

P

packaged analysis app

A single file with an .aclapp extension, similar to a zipped file, that combines an Analytics project (.acl file) and analysis app (.aclx file) into one package containing scripts, data files, and interpretations.

parameter

The names of the expected input for a function or command. A list of parameters is included with each function and command so that when the function or command is used, the script writer can provide the correct input values, or arguments. For example, TRIM(string) shows that the TRIM( ) function expects one parameter called string. When the function is used, one value is provided as the argument that gets assigned to string: TRIM("test ").

physical field

In an Analytics table, a field that defines a unit of actual raw data in a data source. Compare with computed field.

population

In Analytics sampling, the number of records in the data set being sampled, or the absolute value of the numeric sample field.

preferences

User-specified settings for configurable options in Analytics.

preferences file

An Analytics application file, with a .prf extension, that stores a user's preference settings. Also known as a prf file.

Premium Edition

A version of a HighBond product that provides access to a full set of capabilities.

prepare

An ACL command that stratifies a population, and calculates the size required for a sample to be statistically valid. ACLScript syntax: CVSPREPARE

Present

See Presentation.

Presentation

In the Reports app, a presentation is a communication platform that displays multiple tables, charts, and rich content in a single presentation.

presort
An option for several ACL commands that sorts the values in a key field before executing the command. ACLScript syntax: PRESORT

prf file
An Analytics application file, with a .prf extension, that stores a user's preference settings. Also known as a preferences file.

privilege
A security feature that determines the elevated access a user has in HighBond.

Procedure X-Ray
A feature in Projects that provides additional context about a procedure that is being executed.

profile
An ACL command that generates summary statistics on one or more numeric fields. ACLScript syntax: PROFILE

project
(1) A management system in Projects that enables users to define objectives, risks, and controls, perform tests, and compile information into a final report. Also known as risk assessment, compliance assessment, audit, program, engagement. (2) The top-level organizational container in Analytics, which stores data analysis information, but does not store any source data.

project folder
A second-level organizational container in Analytics, similar to a Windows folder, but without any existence outside an Analytics project. Also known as an Analytics folder.

project plan
A template that identifies procedures to be evaluated in a Workplan project in Projects. Also known as audit plan.

project state
A system classification in the Projects app that identifies a project as "Active", "Archived", "Framework", or "Deleted".

project status
A user-defined classification in the Projects app that provides the ability to group projects for workflow management and reporting purposes.

project type
A project structure in Projects that supports customizable terms.

Projects
A HighBond app that allows you to plan, manage, execute, and report work across your team and organization. Also known as Audits, Compliance Assessments, Risk Assessments.

Q

quick filter
A feature in Analytics that lets you use the mouse to select filter values and criteria when applying a filter to a view.

quick search
A feature in Analytics that lets you search a view by entering a search term.

quick sort
A feature in Analytics that lets you sort a table by right-clicking a column heading in a view.

R

random selection method
In Analytics sampling, a method of record selection in which each selected record is randomly chosen from the entire population of records, or from each population stratum.

ratio estimation
In Analytics classical variables sampling, one of the methods available for projecting results to an entire population.

record sampling
One of three types of sampling in Analytics, appropriate if you are interested in the rate of deviation from a prescribed control. Also known as attributes sampling.

record structure
The characteristics of records in a table, such as the number and order of the fields, and the field length and data type. Also known as data structure.

regular expression (regex)
A sequence of characters that defines a search pattern. Usually this pattern is then used by string searching algorithms for "find" or "find and replace" operations on strings. Each character in a regular expression is understood to be a meta character (with its special meaning), or a regular character (with its literal meaning). For example, in the regex "a.", "a" is a literal character that matches just "a", and "." is a meta character that matches every character except a newline.

relate

An ACL command that temporarily associates up to 18 Analytics tables with different record structures and allows you to access the data in the tables as if it existed in a single physical table. ACLScript syntax: DEFINE RELATION

Reports

A HighBond app that provides users with comprehensive report building capabilities.

residual risk

(1) A calculation in Projects that determines the risk that remains after controls are taken into account. Also known as net risk. (2) A calculation in Strategy that derives from an assessment of how much risk remains after controls and other mitigating factors have been put in place.

residual risk heat

A calculation in Strategy that derives from adding the residual risk score of each operating segment and dividing by the total possible inherent risk score across all operating segments specified in the Strategy Map.

results

(1) The values or records created by processing data with an ACL command. (2) Any output tables, output files, or log files that are generated when an analytic script runs on AX Server. Also knows as analytic results.

Results

A HighBond app that helps users organize, track, and remediate issues identified by data analytic results.

results tab

A component in Analytics that contains command results output to the screen, or the details of a command log entry.

risk

An effect of uncertainty on an objective, with the effect having a positive or negative deviation from what is expected. Also known as requirement.

risk control matrix

A template that contains mapped risks and controls common to an objective in an Internal Control project in Projects.

risk heat

A calculation in Strategy that provides the percentage of risk intensity across all operating segments in the organization.

risk heatmap

A graphical representation in Strategy that presents the result of a risk assessment.

risk library
A list of common key risks are disclosed across a given industry, and curated and normalized by Galvanize based on the S&P 500 10-k reports.

risk profile
A view of all identified risks within an organization in Strategy, organized by assigned states.

risk scoring factors
A feature in Strategy and Projects used to illustrate attributes that impact the achievement of objectives within an organization.

risk treatment
The measures an organization takes to mitigate risk. Measures may include initiatives, programs, policies, or control objectives, which you can create in Projects and link to strategic risks in Strategy.

risk workshop
A collaboration tool in Strategy that allows you to invite different stakeholders to collaborate in the risk assessment process.

RiskBond
A risk management solution in HighBond that helps organizations identify, assess, respond to, and monitor enterprise risks.

robot
A container in Robots that houses committed analytic scripts, any helper scripts, and related files.

Robotics
A Galvanize product containing Analytics, Robots, and Results.

Robots
A HighBond app that is used to automate repetitive tasks using scripts built in Analytics.

Robots Agent
An on-premise, or cloud-based, Robots component that uses the Analytics script engine to run scripts against data.

role
A security feature that determines the basic access a user has in HighBond.

rollforward
A feature in Projects that allows you to re-use projects or create custom project templates from archived projects.

routine
A sequence of program instructions that perform a specific task, packaged as a unit. A function is a routine.

S

sample
An ACL command that draws a selection of records from a larger population of records. ACLScript syntax: SAMPLE, CVSSAMPLE

sample size
In Analytics sampling, the number of records in the sample selection.

sampling interval
In Analytics sampling, the distance between the records selected in the sample. Also known as interval.

SAP connector
A proprietary driver, integrated with Analytics, that provides an interface to an SAP system.

schedule
A plan that specifies when control activities should be performed.

Scheduler
A HighBond app that provides a convenient way to manage project timelines and resources within an organization.

score
A value derived from assessing a single entity associated with a risk in Strategy, calculated by multiplying risk scoring factors together, with each risk scoring factor multiplied by its assign weight.

script
A series of ACLScript commands that performs a particular task, or several related tasks, which is saved so it can be executed repeatedly and automatically.

Script Editor
A component in Analytics for creating, editing, testing, and debugging Analytics scripts.

script engine
The computer program inside Analytics, the Robots Agent, and Analytics Exchange that directly runs the instructions written in a script. When you run a script, the script engine reads the script file and performs the commands in the order specified in the script.

Script Recorder
A feature in Analytics that lets you create a script by recording your actions as you work in the user interface.

ScriptHub
A resource that provides Analytics customers with access to hundreds of scripts developed by Galvanize's consulting experts.

security certificate
An electronic document used to establish a trusted, secure, encrypted connection between client applications and AX Server.

seed
A value that initializes the Analytics random number generator. Can be user-specified, or automatically generated by Analytics.

self-signed certificate
A security certificate used to establish a trusted, secure, encrypted connection that is signed by the same entity whose identity it certifies.

sequence
An ACL command that tests whether a field is sequentially ordered, and identifies out-of-sequence items. ACLScript syntax: SEQUENCE

serial datetime
A way of describing a datetime value as the number of days elapsed since January 1, 1900. The value is expressed as a numeric and the time portion of the value is a decimal. For example, on March 16, 2023, 45000 days have elapsed since January 1, 1900. Therefore 45000.25 expresses March 16, 2023 06:00:00 as a serial datetime.

server table
A table structure that has a local table layout in Analytics and a source data file on AX Server.

shared data file
A source data file that is linked to more than one table layout.

shared table
A table in Analytics Exchange that shares its data source file with one or more other tables.

shared table layout
A table layout that is linked to more than one source data file.

size
An ACL command that calculates the size required for a sample to be statistically valid. ACLScript syntax: SIZE

Solution

A package that contains pre-built content and robots. By implementing a solution, you minimize the time needed to learn and configure HighBond and get value faster, with correspondingly lower implementation and service costs. Solutions are deployed in the form of solution toolkits.

Solution toolkit

Special toolkits that populate your HighBond instance with the pre-built content and robots relevant to a corresponding solution.

sort

An ACL command that physically reorders data into a sequential order and outputs the results to a new Analytics table. ACLScript syntax: SORT

source code

A collection of commands, functions, expressions, and comments written in plain text using ACLScript syntax and saved in a script file. The source code of a script represents the specific instructions that script writers prepare for the script engine to run.

source data

The raw data that is linked to and interpreted by an Analytics table.

standalone layout

A table structure in Analytics Exchange that contains the metadata describing the structure of a source data file but that has been disassociated from its underlying data.

standalone table

A table structure in Analytics Exchange consisting of a source data file that contains raw data records and a table layout that contains the metadata describing the structure of the source data file.

Standard Edition

A version of a HighBond product that provides access to a subset of capabilities.

statistics

An ACL command that generates several measures for one or more numeric or datetime fields. ACLScript syntax: STATISTICS

status bar

A component in Analytics, dynamically updated, that displays current information such as the name of the active Analytics table and the record count.

Storyboards

A HighBond app that allows you to display multiple visualizations and rich text content in a single presentation.

strategic objective
A business process, functional area, or auditable area that relates to an operating segment in Strategy.

Strategy
A HighBond app that is used to identify, assess, and track risks.

Strategy Heatmap
A graphical representation in Strategy that identifies the relative severity of risks in an organization.

Strategy Map
A graphical representation in Strategy that displays your business and legal entity structure. It allows you to assess your organization's strategic risks.

stratification
In Analytics classical variables sampling, the process of subdividing the population into numeric ranges, or strata, which reduces the required size of the sample.

stratify
An ACL command that groups the records in a table into numeric ranges. ACLScript syntax: STRATIFY

string
A sequence of characters, either as a literal or as some kind of variable.

subscription type
A model that defines the roles and/or privileges a user can be assigned in HighBond.

summarize
An ACL command that groups the records in a table based on identical values in one or more character, numeric, or datetime fields. ACLScript syntax: SUMMARIZE

supporting evidence
Information, in the form of documents or photographs, used by auditors to support an assertion.

survey
A table in Results that stores the responses to a questionnaire that is deployed as an anonymous link or to a list of recipient email addresses.

syntax
The set of rules that defines the correct structure of a script. ACLScript has a unique syntax and does not run unless the source code follows the syntax rules.

syntax capture
A feature in Analytics that automatically generates the ACLScript syntax for commands you select, but does not process the commands.

T

table
(1) The general name for the three-part structure that Analytics uses to store, define, and display data, containing a source data file, a table layout, and a view. (2) The container in Results that arranges data in a two-dimensional structure of rows and columns.

table layout
The part of an Analytics table that contains metadata for interpreting the source data, and specifying its location.

Table Layout dialog box
A component in Analytics that you use to create or modify table layouts.

tainting
In Analytics monetary unit sampling, the percentage of a book value that a misstatement represents.

Task Tracker
A HighBond app that tracks to-dos, requests, and reviews across your entire organization.

template
A pre-built project, questionnaire, or report that can be used as a starting point.

test plan
A document that details the assessment of controls in Projects.

testing
The process of assessing the operating effectiveness of internal controls within an organization.

testing round
A phase of testing in Projects that assesses the operating effectiveness of controls.

Timesheets
A HighBond app that visually tracks time usage and overall resource utilization across the organization.

to-do
A collaboration feature in Projects that can be used to set up reminders and assign tasks.

Toolkit
A curated set of tools aimed at addressing one area of risk or compliance. For example, a toolkit might include a set of data analytics, a risk control framework, and a best practices program.

top stratum cutoff
In Analytics monetary unit sampling, an option that allows you to adjust the bias that favors selecting larger amounts.

total
An ACL command that totals the values in a numeric field. ACLScript syntax: TOTAL

trigger
A set of actions in Results that automatically execute when a specific change occurs.

U

Unicode
An industry-standard method of character encoding that supports most of the world's languages. Unicode is commonly used with global information systems, or data that contains multi-byte characters such as Asian, Cyrillic, or Arabic characters.

V

value
An expression which cannot be evaluated any further. For example, 1 + 2 is not a value because it can be reduced to the expression "3". This expression "3" cannot be reduced further and is therefore a value.

variable
A temporary storage location, or "container", with a name that is used to hold a value. The contents of the container can change but the name cannot, so the name lets a script reference and work with the value stored in the computer's memory without knowing what is actually stored in the location. For example, v_char = "my word" stores "my word" in memory, and a script can then reference this value using the variable name v_char.

verify
An ACL command that checks for data validity errors in an Analytics table. ACLScript syntax: VERIFY

view
(1) The part of an Analytics table that displays data using named columns and numbered records. (2) A component in Reports that defines the categories and fields from HighBond

that you can use to build a report.

view filter
A filter that applies to a table view and restricts which records are displayed, or processed. Also known as a global filter.

view tab
A component in Analytics that contains one or more views associated with an Analytics table.

visualization
A graphical representation in the Analysis App window, AX Web Client, and Results that presents result data visually in a number of different chart types and table formats.

Visualizer
A component of the Analysis App window, AX Web Client, and Results that is used to work with interpretations and visualizations.

W

walkthrough
A procedure performed by an auditor or frontline manager to establish the reliability of controls and test the design of controls. Also known as evaluation, outcome, procedure result.

width
The number of characters in a column in an Analytics view, which is not the same as the field length in a table layout. Also known as column width.

workflow
(1) A set of statuses in Results that identify the state of records being moved through a remediation process. (2) A plan in Projects that defines the components available in a project.

working directory
(1) The user-specified default location for saving Analytics projects and output tables. (2) A directory in Analytics Exchange that stores content available to administrators and users.

workplan
A workflow in Projects that consists of a set of steps or procedures an assurance team will execute, and the documentation of the outcome of each step.

workspace
In an Analytics project, a container for saving physical and computed field definitions and filters so that they can be reused with multiple tables.