ACL スクリプト リファレンス

バージョン: 14.1

発行済み 2019年6月19日

HighBond

目次

はじめに

Analytics でのスクリプト作成の基本事項

ACLScript は Analytics コマンドをプログラミングして自動的に実行できるようにするコマンド言語です。ACLScript の構造および要素はシンプルですが強力です。

> **メモ**
>
> まったく初めてスクリプトを作成する場合は、いきなりこのコンテンツを学習する前に、アカデミーを訪れて基本的なトレーニングを受けてください。スクリプトの作成と Analytics の使用に関するコースについては、www.highbond.com を参照してください。

コマンド

スクリプト内のどの行も ACLScript コマンドを実行するため、コマンド名で始まります。コマンドは、Analytics で操作を実行するための命令です。

コマンド名の後に *parameter_name parameter_value* の形を持つパラメーターが 1 つ以上続きます。

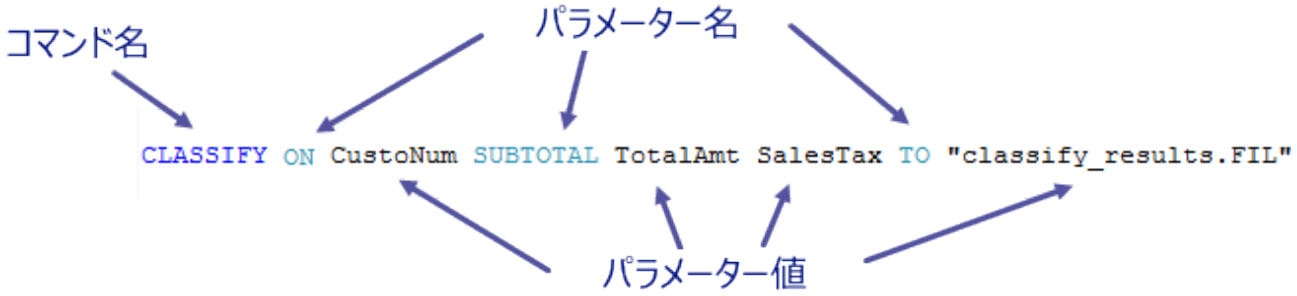
> **ヒント**
>
> コマンドによって、必須になるパラメーターとオプションになるパラメーターがあります。オプションのパラメーターの指定は必須ではありません。オプションのパラメーターは、省略しても、コマンドの実行が可能です。必須のパラメーターを省略した場合は、そのパラメーターのデフォルト値が使用されます。

CLASSIFY コマンドを使用した例

次の例は、CLASSIFY コマンドとそれに続くパラメーターを示しています。

- ON - 分類化の基準にする、テーブルのフィールドを指定します。
- SUBTOTAL - 出力における小計の対象にするオプションのフィールドを指定します。
- TO - CLASSIFY コマンドの結果を書き込むテーブルを指定します。

各パラメーターの後に 1 つまたは複数のパラメーター値が指定されていることに注目してください。



コマンド構文に関する重要なメモ

- スクリプトの行の最初の単語はコマンド名である必要がある
- ほとんどのコマンドにおいて、コマンド名の後に指定するパラメーターの順序は重要ではない
- 大半のコマンドにおいて、コマンドを実行する前にターゲット テーブルを開くこと、つまりコマンドの前に OPEN *テーブル名* を実行することが必要です。

コメント

ACLScript では、スクリプト言語のように COMMENT キーワードを使ってコメントを追加できます。コメントを使うことで、作成するコードを理解しやすくできるとともに、スクリプトを参照、使用、または理解しようとしているユーザーと対話することができます。ACLScript では、2 種類のコメントがサポートされています。

- **単一行のコメント** - COMMENT の後から行末までのすべてのテキストが無視されます。
- **複数行のコメントから成るブロック** - COMMENT から END キーワードまでのすべての行または空白行が無視されます。

詳細と例については、"コメント" ページ 19を参照してください。

データ型

ACLScript では、4 つの基本的なデータ型がサポートされています。

- **論理型** - 最もシンプルなデータ型。論理型のデータは、T(True、真) または F(False、偽) という真理値を表す
- **数値型** - 0 ～ 9 の数字のほか、必要な場合に使用される負号や小数点がある
- **文字型** - 1 つまたは複数の文字
- **日付時刻型** - サポートされる書式で表される日付、日付時刻、または時刻の値

各データ型は Analytics によって異なる方法で処理され、さまざまなコマンドや関数で使用できます。データ型の詳細については、"データ型" ページ 21を参照してください。

式

式とは、値を持つすべてのステートメントのことです。最もシンプルな形の式は 2 や "test" などのリテラルですが、式は通常計算として使用され、演算子、条件、関数、および指定可能な値を有効に組み合わせることができる範囲で複雑にすることもできます。

```
((2 + (3 - 2)) * 2) > ROOT(9,0)
```

式は通常、Analytics 内で、演算フィールドに値を入力するために使用されたり、条件付きロジックの入力として使用されします。式の詳細については、"式" ページ 22を参照してください。

関数

関数とは、一定数のパラメーターを入力とし、単一の値を返す組み込みルーチンです。関数を使用することで、コマンド内で使用する変数や、フィールド値を操作できます。

> **メモ**
>
> 関数は、フィールド データを変更せず、入力としてフィールド データまたは変数を使用するアルゴリズムまたは計算に基づいて、新しい値を生成して返します。関数が返す値をコマンドの入力として使用します。

関数は、関数名で始まり、そのすぐ後に左かっこ、関数に引数として渡される0個以上の値をカンマで区切ったリスト、右かっこと続きます。

例

BETWEEN(*値*, *最小値*, *最大値*) 関数は3つの引数を取り、値が最小値から最大値までの範囲内であればTrueを返し、範囲外であればFalseを返します。

- 値 - テストする式またはフィールド
- 最小値 - 範囲の最小値
- 最大値 - 範囲の最大値

```
BETWEEN(amount, 500, 5000)
```

関数の詳細については、"関数" ページ 25を参照してください。

変数

変数とは、値を一時的に格納する場所です。変数には対応するIDがあり、これを使ってコンピューターのメモリに格納されている値を参照、使用できます。

ACLScriptは、ASSIGNコマンドを使って、変数を作成すると同時にそれに値を代入します。

```
ASSIGN v_age_in_years = 3
```

簡略化するために、ASSIGNキーワードを省略することもできますが、ASSIGNは暗黙に使用されて実行されます。

```
v_age_in_years = 3
```

> **メモ**
>
> ACLScript ではヌル値はサポートされていません。すべての変数には、サポートされているいずれかのデータ型を持つ、対応する値を設定する必要があります。スクリプトのインタープリターは、値の代入に使用されたデータ形式およびデータ修飾子を使ってデータ型を評価します。詳細については、"データ型" ページ 21を参照してください。

変数の使用

変数は作成後、フィールド名や変数を参照する任意の場所で参照できます。また、ASSIGN コマンドを使って、それに新しい値を代入することもできます。

```
EXTRACT RECORD TO 'result.fil' IF age > v_age_in_years
v_age_in_years = 5
```

また、文字列補間または変数代入を使用して、文字列リテラルを含めるには、% 文字に変数をラッピングできます。代入される変数が Analytics で検出されると、プレースホルダーが対応する値に置換されます。

```
ASSIGN v_table = erp_data
OPEN %v_table%
```

変数の詳細については、"変数" ページ 27を参照してください。

統制構造

制御構造は、指定したパラメーターに基づいて取るべき方向を決定する、スクリプトのコンポーネントです。ACLScript には、条件分岐のロジックおよびループ構造が用意されています。

条件分岐ロジック

ACLScript には、この言語の IF コマンドや、多数のコマンドのオプション パラメーターとして、条件付きロジックが実装されています。

> **ヒント**
>
> IF コマンドは他のコマンドを実行するかどうかを制御するのに使用するのに対し、IF パラメーターは他のコマンドの実行対象とするテーブル内レコードを決定するのに使用します。

IF コマンド

```
IF v_counter > 10 CLASSIFY ON customer_no
```


IF パラメーター

```
CLASSIFY ON customer_no IF state = 'NY'
```

ループ

LOOP コマンドは、ACLScript におけるループ制御構造を提供します。このコマンドは、統制テストの式が True と評価される限り、ループ内のステートメントを処理します。

制御構造の詳細についての参照先："統制構造" ページ 29

コメント

ACLScript では、スクリプト言語のようにCOMMENTキーワードを使ってコメントを追加できます。コメントを使うことで、作成するコードを理解しやすくできるとともに、スクリプトを参照、使用、または理解しようとしているユーザーと対話することができます。

コメントの種類

ACLScript では、2 種類のコメントがサポートされています。

- **単一行のコメント** - COMMENT の後から行末までのすべてのテキストが無視されます。
- **複数行のコメントから成るブロック** - COMMENT から END キーワードまでのすべての行または空白行が無視されます。

単一行のコメント

単一行のコメントは、作成するスクリプト内の各ステップについて説明したり、変数について説明したりするために使用できます。

```
COMMENT *** 分析期間の開始日
v_Start_Date = `20150101`
```

複数行のコメントから成るブロック

複数行のコメントから成るブロックは、スクリプトまたはスクリプトのセクションについて説明する場合に使用できます。

```
COMMENT
**********************
** このスクリプトのセクションは、インポート対象となるデータを準備します
**********************
END
```

ヘッダー コメント ブロック

各スクリプトの先頭にスクリプトの主な内容を記述したヘッダーコメント ブロックを追加するのは、有効な方法です。

```
COMMENT
************************************************************
```

```
*** スクリプト名 :  {App_ID}{Script name}
*** パラメーター:   {Detailed description}
*** 出力:       {Describe parameters}
*** 作成者:   {Name}, ABC Corporation, {Month YYYY}
*** 変更者:  {Name}, ABC Corporation、スクリプトの目的とロジック
*** バージョン:      1.1.1 {app_ver.script_ver.defect.fix}
************************************************************
END
```

データ型

ACLScript では、論理、数値、文字、日付時刻という4 つの基本的なデータ型がサポートされています。

種類	説明	制限	修飾子あり	例
文字	1 つまたは複数の文字です。	32,767 バイト	一重引用符または二重引用符	◦ 'John Doe' ◦ "John Doe"
数値	数値型の値には、0 ～ 9 の数字のほか、必要な場合に使用される負号や小数点があります。	22 桁	修飾子なし	◦ 100 ◦ -5 ◦ 5.01 ◦ 22222.1232
日付時刻	サポートされる書式で表される日付、日付時刻、または時刻の値です。	◦ 最小値 = 1900-01-01 ◦ 最大値 = 9999-12-31	◦ バッククォート ◦ 時刻値の場合：先頭の 't' または単一の空白	◦ `20160101` ◦ `141231` ◦ `t2359` ◦ `20141231T23595-9` ◦ `20141231 235959`
論理	最もシンプルなデータ型です。論理型のデータは、T (True、真) または F(False、偽) という真理値を表します。 '=', '>', '<' などの比較演算子は論理型の値を返します。	◦ T ◦ F	修飾子なし	`ASSIGN v_truth = 5 > 4` は T と評価されます。

式

式とは、値を持つすべてのステートメントのことです。最もシンプルな形の式はリテラルですが、演算子、条件、関数、および指定可能な値を有効に組み合わせることができる範囲で複雑にすることもできます。

式の要素

リテラル値

リテラル値は、文字リテラル'私の値'など、文字どおりに解釈されるように記述された値です。リテラルの詳細については、"データ型" 前のページを参照してください。

演算子

演算子は、算術を実行したり、指定された値の文字列評価、比較評価、または論理的評価を実行したりするようにスクリプトのインタプリターに指示する記号です。

演算子の種類(優先順位順に一覧表示)	演算子(優先順位順に一覧表示)	例
かっこ	◦ () は優先順位を指定します ◦ () 関数の演算子	(5 + 3) * 2
単項	◦ 非論理 ◦ - 否定	v_truth = NOT (3 < 2)
算術演算子	◦ ^ 指数 ◦ * 乗算, / 除算 ◦ + 加算, - 減算 **メモ** 乗算演算子同士の優先順位は等しいため、左から右に評価されます。 加算演算子同士の優先順位は等しいため、左から右に評価されます。	1 + 5 - 3 * 2
文字列	+ 連結	"これは私の" + "スクリプトです"
比較演算子	◦ < より小さい ◦ > より大きい ◦ = 等しい ◦ >= 以上 ◦ <= 以下 ◦ <> 等しくない	IF amount <> 100

演算子の種類(優先順位順に一覧表示)	演算子(優先順位順に一覧表示)	例
	メモ 比較演算子同士の優先順位は等しいため、左から右に評価されます。	
二項論理演算子	◦ AND または & ◦ OR または |	IF amount > 5 AND amount < 10

関数

式は、関数によって返された値を使って評価されます。関数は、式の全要素のうち、最も高い優先順位で実行されます。関数の詳細については、"関数" ページ 25を参照してください。

式の例

6 として評価されます。

```
(2 + (3 - 2)) * 2
```

True として評価されます。

```
((2 + (3 - 2)) * 2) > ROOT(9,0)
```

'ACLScript チュートリアル' として評価されます。

```
'AC' + 'LScri' + 'pt ' + 'チュートリアル'
```


式を使用した演算フィールドの定義

現在開いているテーブルに式を使って追加のデータフィールドを作成するには、**演算フィールド**を使用します。演算フィールドは、開いているテーブルに追加されるフィールドであり、指定された式の値が入力されます。

演算フィールドの構文

```
DEFINE FIELD 名前 COMPUTED 式
```

- **名前** - 生成する演算フィールドの名前
- **式** - フィールドの値を生成するための演算または計算

演算フィールドの例

```
DEFINE FIELD c_full_name COMPUTED first_name + ' ' + last_name
```

> **ヒント**
>
> 演算フィールドの前には c_ という接頭辞を付けることで、このフィールドが元のソースデータでなく演算データであることを示します。

条件付き演算フィールド値の定義

演算フィールドに条件を使うことで、様々な場合用に値を定義することもできます。

```
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
```

最初の条件式が True と評価された場合、その場合に指定されていた値が使用されます。この例では、amount * general_rate は、いずれの条件式も True と評価されなかった場合に使用されるデフォルト値です。

> **メモ**
>
> DEFINE FIELD コマンドで IF、WIDTH、PIC、または AS パラメーターを指定する場合を除いて、行のコマンドと条件の間には空白行を追加する必要があります。詳細については、"DEFINE FIELD . . . COMPUTED コマンド" ページ 127を参照してください。

関数

関数とは、一定数のパラメーターを入力とし、単一の値を返す組み込みルーチンです。関数を使用することで、コマンド内で使用する変数や、フィールド値を操作できます。

> **メモ**
>
> 関数は、フィールド データを変更せず、入力としてフィールド データまたは変数を使用するアルゴリズムまたは計算に基づいて、新しい値を生成して返します。関数が返す値をコマンドの入力として使用します。

関数構文

関数は、関数名で始まり、そのすぐ後に左かっこ、関数に引数として渡される 0 個以上の値をカンマで区切ったリスト、右かっこと続きます。

例

BETWEEN(*値*, *最小値*, *最大値*) 関数は 3 つの引数を取り、値が最小値から最大値までの範囲内であれば True を返し、範囲外であれば False を返します。

- 値 - テストする式またはフィールド
- 最小値 - 範囲の最小値
- 最大値 - 範囲の最大値

```
BETWEEN(amount, 500, 5000)
```

関数の引数

関数の引数とは、その関数に渡される特定の入力値のことです。

関数の引数は引数リストを介して関数に渡されます。引数リストは、リテラル値、変数および、パラメーターのデータ型の値として評価される式をカンマで区切ったリストです。データ型の操作の詳細については、"データ型" ページ 21を参照してください。

> **メモ**
>
> プロジェクトでヨーロッパの数値書式を使用する場合、または複数のリージョンで互換性のあるスクリプトを作成する場合、符号付き数値を渡すのでなければ、カンマでなく空白文字で関数の引数を区切ってください。符合付き数値を受け入れる関数を使用するには、明示的な区切り文字が必要です。

関数とコマンド

コマンドと関数の違いは微妙ですが、この違いは ACLScript を使用する場合には重大な意味を持ちます。

関数	コマンド
入力としてフィールド、値、またはレコードを使用し、新しい値を生成して、それを返します。	入力としてテーブルを使用し、新しいレコードおよびテーブルを生成します。
式、演算フィールド、コマンド パラメーター値、変数、フィルターの中で使用され、コマンドの実行を支援、変更します。	データの分析およびインポート、ならびに結果の生成に使用されます。
スクリプト内で独立したステップになることはできません。	スクリプト内で独立したステップになることができます。

変数

変数とは、値を一時的に格納する場所です。変数には対応するIDがあり、これを使ってコンピューターのメモリに格納されている値を参照、使用できます。

ACLScript内で変数を使用する方法

変数の作成とそれへの値の代入

ACLScriptは、ASSIGNコマンドを使って、変数を作成すると同時にそれに値を代入します。

```
ASSIGN v_age_in_years = 3
```

簡略化するために、ASSIGNキーワードを省略することもできますが、ASSIGNは暗黙に使用されて実行されます。

```
v_age_in_years = 3
```

> **メモ**
>
> ACLScriptではヌル値はサポートされていません。すべての変数には、サポートされているいずれかのデータ型を持つ、対応する値を設定する必要があります。スクリプトのインタープリターは、値の代入に使用されたデータ形式およびデータ修飾子を使ってデータ型を評価します。
> 詳細については、"データ型" ページ21を参照してください。

変数の使用

変数は作成後、フィールド名や変数を参照する任意の場所で参照できます。また、ASSIGNコマンドを使って、それに新しい値を代入することもできます。

```
EXTRACT RECORD TO 'result.fil' IF age > v_age_in_years
v_age_in_years = 5
```

また、文字列補間または変数代入を使用して、文字列リテラルを含めるには、%文字に変数をラッピングできます。代入される変数がAnalyticsで検出されると、プレースホルダーが対応する値に置換されます。

```
ASSIGN v_table = erp_data
OPEN %v_table%
```

変数のタイプ

Analytics では次の種類の変数が使用されます。

- **システム生成変数** –: この変数は、コマンドを実行すると自動的に生成されます。
- **永久変数** –: この変数は、削除されるまで、コンピューターのメモリに残り、Analytics プロジェクトを閉じた後にも永続します。

  > **メモ**
  >
  > 永久変数を定義するには、識別子の前に'_'を付けます: `_v_company_name = 'Acme'`.

- **セッション変数** –: この変数は、削除されるまで、また、Analytics プロジェクトを閉じるまで、コンピューターのメモリに残ります。

変数 ID

変数 ID は大文字と小文字を区別せず、変数の種類に関連した特定の規約に従います。

- システム生成変数 ID では OUTPUTFOLDER のようにすべて大文字が使用されます。
- 永続的変数 ID には、_v_permanent のように、'_'を接頭辞として付ける必要があります。
- セッション変数 ID では規則により v_*varname* という書式が使用されますが、この命名規則に制限されることはありません。

変数値の表示

スクリプト開発中またはデバッグ中にスクリプトを実行する際、変数値を追跡すると有益な場合があります。スクリプトのログファイルに変数値を収集するには、DISPLAY コマンドを使用します。

```
DISPLAY v_age_in_years
```

ACLScript はこのコマンドに遭遇すると、そのコマンドをログファイルに記録します。スクリプト実行のこの段階で変数値を表示するには、ログに記録されたそのエントリをクリックします。

> **ヒント**
>
> 変数を使って、デバッグを支援することもできます。それには、スクリプト内にブレークポイントを挿入したうえで、**ナビゲーター**の [**変数**]タブで変数値を調べます。

統制構造

制御構造は、指定したパラメーターに基づいて取るべき方向を決定する、スクリプトのコンポーネントです。ACLScript には、条件分岐の IF ロジックおよびループ構造が用意されています。

IF を使用した条件分岐ロジック

ACLScript には、この言語の IF コマンドや、多数のコマンドのオプション パラメーターとして、条件分岐ロジックが実装されています。

- **コマンド** -の指定により、そのコマンドを実行するかどうかを制御します。
- **パラメーター**-により、そのコマンドの実行対象となる、テーブル内のレコードを決定します。

IF コマンド

IF コマンドを使用する際には、条件式と、その式が True として評価された場合に実行するコマンドを指定します。

```
IF v_counter > 10 CLASSIFY ON customer_no
```

この条件付き構造は実行するコードを制御するので、テストの式に基づいてテーブル全体を処理したい場合には、IF コマンドを使用できます。式が True として評価された場合には、コマンドはテーブル内のすべてのレコードに対して実行されます。IF コマンドの詳細については、"IF コマンド" ページ 227を参照してください。

IF パラメーター

コマンドの実行対象にするレコードをフィルターで抽出するために使用できるオプションの IF パラメーターは、数多くのコマンドが受け付けています。

```
CLASSIFY ON customer_no IF state = 'NY'
```

このステートメントが実行されると、state フィールドの値が 'NY' である、テーブル内のすべてのレコードが分類化されます。

ループ

LOOP コマンド

LOOP コマンドは、ACLScript におけるループ制御構造を提供します。

> **メモ**
>
> LOOP コマンドは、GROUP コマンド内で実行する必要があります。単独では実行できません。

このコマンドは、指定された WHILE 式が True である限り、ループ内のステートメントを処理します。

```
ASSIGN v_counter = 10
GROUP
 LOOP WHILE v_counter > 0
  v_total = v_total + amount
  v_counter = v_counter - 1
 END
END
```

この構造は amount フィールドの値を変数 v_total への加算を 10 回繰り返します。各繰り返しの終わりに v_counter 変数が 1 ずつ減算され、WHILE 式でテストされます。この式が False と評価されると、ループは終了し、スクリプトが進行します。

ループが終了すると、v_total に 10 レコードの amount フィールド値の合計が格納されます。

ループの詳細については、"LOOP コマンド" ページ 324を参照してください。

LOOPING とサブスクリプト

時々、LOOP コマンドは必要な正確なループ機能を提供しません。このような場合、DO SCRIPT コマンドを使用して別の Analytics スクリプトを呼び出すこともできます。DO SCRIPT *スクリプト名* WHILE *条件テスト*。

次の一般的な方法のいずれかを使用して、ループが終了するときを制御できます。

- **フラグ** - 論理フラグ変数が偽に設定されるまでループが続きます。
- **カウンター** - 増分または減少変数が条件しきい値を超えるまでループが続きます。

サブスクリプトの呼び出しの詳細については、"DO SCRIPT コマンド" ページ 163を参照してください。

例

C:\data フォルダーのすべての CSV ファイルをプロジェクトにインポートする必要があります。DIRECTORY コマンドを使用して、フォルダーからファイルのリストを取得できます。ただし、IMPORT コマンドを GROUP 構造内で使用できません。DIRECTORY が作成するテーブルを使用した別のループ方法が必要です。

このためには、次のメイン スクリプトを作成します。

1. DIRECTORY コマンドを実行し、結果をテーブルに保存します。
2. テーブルのレコード数を取得し、カウンターとして使用します。
3. テーブルのレコードごとにサブスクリプトを 1 回呼び出し、現在のレコードに対して IMPORT コマンドを実行します。

メインのスクリプト

```
COMMENT メインのスクリプト

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
```

サブスクリプトのインポート

```
COMMENT Import_Subscript

OPEN T_Table_To_Loop
LOCATE RECORD v_Counter

COMMENT CSV ファイルをインポートするためのコード ...

ASSIGN v_Counter = v_Counter + 1
```

変数はプロジェクトで実行されるすべてのスクリプトで共有されるため、メイン スクリプトは、v_Counter の値が v_Num_Records の値を超えるまでサブスクリプトを呼び出します。サブスクリプトが実行されるたびに、v_Counter が増えます。

この構造では、各レコードに対して IMPORT コマンドを呼び出しながら、テーブルをループできます。メインのスクリプトが完了すると、`C:\data` フォルダーからすべての CSV ファイルがインポートされます。

グループ化とループ処理

GROUP および LOOP コマンドは、一連のコマンドを繰り返し実行するための 2 つの方法を提供します。GROUP は各レコードに対して 1 つ以上のコマンドを 1 回繰り返します。LOOP は単一レコードに対して一連のコマンドを複数回繰り返して実行し、GROUP ブロック内でのみ使用できます。

GROUP の簡単な例

請求書データのテーブル **Ap_Trans** があります。このデータを使用して、請求金額の合計を計算する必要があります。

Vendor_ Number	Vendor_Name	Vendor_ City	Invoice_ Number	Invoice_ Date	Invoice_ Amount	Quantity	Unit_ Cost
11663	More Power Industries	Los Angeles	5981807	2000-11-17	618.30	90	6.87
13808	NOVATECH Wholesale	Des Moines	2275301	2000-11-17	6705.12	976	6.87
12433	Koro International	Sheveport	6585673	2000-11-17	7955.46	1158	6.87

この金額を計算するには、GROUP コマンドを使用します。GROUP の各繰り返し内で次のことを実行します。

1. 現在のレコードの時点で合計を計算します。
2. 請求書番号、金額、日付、合計を結果テーブルに抽出します。

```
OPEN AP_Trans

COMMENT 累計の初期値をゼロに設定
ASSIGN v_running_total = 0.00

COMMENT テーブルの各レコードを繰り返し、累計を計算して抽出
GROUP
 ASSIGN v_running_total = v_running_total + Invoice_Amount
 EXTRACT Invoice_Number, Invoice_Amount, Invoice_Date, v_running_total AS "Running total" TO
results1
END
```

スクリプトが実行されると、GROUP ブロック内のコマンドはテーブルの各レコードに対して上から下に処理され、合計が計算され、抽出されます。実行される GROUP コマンドを見ると、次のような仕組みになっています。

GROUP の最初の繰り返し: 合計 = 0.00 + 618.30

GROUP は最初のレコードの請求金額を 0.00 の初期合計に追加し、フィールドを結果テーブルに抽出します。

Vendor_ Number	Vendor_Name	Vendor_ City	Invoice_ Number	Invoice_ Date	Invoice_ Amount	Quantity	Unit_ Cost
11663	More Power Industries	Los Angeles	5981807	2000-11-17	618.30	90	6.87
13808	NOVATECH Wholesale	Des Moines	2275301	2000-11-17	6705.12	976	6.87
12433	Koro International	Sheveport	6585673	2000-11-17	7955.46	1158	6.87

GROUP の 2 番目の繰り返し: 合計 = 618.30 + 6705.12

GROUP は 2 番目のレコードの請求金額を新しい合計の 618.30 に追加し、フィールドを結果テーブルに抽出します。

Vendor_ Number	Vendor_Name	Vendor_ City	Invoice_ Number	Invoice_ Date	Invoice_ Amount	Quantity	Unit_ Cost
11663	More Power Industries	Los Angeles	5981807	2000-11-17	618.30	90	6.87
13808	NOVATECH Wholesale	Des Moines	2275301	2000-11-17	6705.12	976	6.87
12433	Koro International	Sheveport	6585673	2000-11-17	7955.46	1158	6.87

GROUP の 3 番目の繰り返し: 合計 = 7323.42 + 7955.46

GROUP は 3 番目のレコードの請求金額を新しい合計 7323.42 に追加し、フィールドを毛かテーブルに抽出します。

Vendor_ Number	Vendor_Name	Vendor_ City	Invoice_ Number	Invoice_ Date	Invoice_ Amount	Quantity	Unit_ Cost
11663	More Power Industries	Los Angeles	5981807	2000-11-17	618.30	90	6.87
13808	NOVATECH Wholesale	Des Moines	2275301	2000-11-17	6705.12	976	6.87
12433	Koro International	Sheveport	6585673	2000-11-	7955.46	1158	6.87

Vendor_ Number	Vendor_Name	Vendor_ City	Invoice_ Number	Invoice_ Date	Invoice_ Amount	Quantity	Unit_ Cost
				17			

最終結果テーブル

GROUP がテーブルの最終レコードを処理した後、次の結果テーブルが作成されます。

Invoice_Number	Invoice_Amount	Invoice_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

GROUP IF を使用したさまざまな場合の処理

上記と同じ **AP_Trans** テーブルを使用し、3 種類の請求書の合計を計算する必要があります。

- 高額 (1000.00 以上)
- 標準額 (100.00 ~ 1000.00)
- 低額 (100.00 未満)

GROUP コマンドは異なる場合を処理するために IF/ELSE 構造を提供します。テストする条件式を提供し、レコードが真と評価される場合は、ブロック内のコマンドが実行されます。

場合のテスト方法

場合は**上から下に**テストされ、レコードは 1 つの IF/ELSE ブロックでのみ処理できます。レコードの真に評価される最初の場合は、レコードを処理するものです。

1. GROUP が最初のレコードを処理すると、最初の IF 条件に対してテストします(Invoice_Amount >= 1000) 。真と評価される場合、この場合のコードが実行され、他の場合はテストされません。
2. 最初の場合が偽と評価されると、次の ELSE IF 条件 (Invoice_Amount >= 100) がテストされます。同時に真と評価される場合、この場合のコードが実行され、他の場合はテストされません。
3. いずれの IF または ELSE IF 場合が真と評価される場合、ELSE ブロックのデフォルトの場合がレコードを処理します。

> **メモ**
>
> レコードが複数の場合で真と評価される場合、レコードはテストする最初の IF/ELSE ブロックでのみ処理されます。レコードは GROUP コマンド内の複数の IF/ELSE ブロックでは処理されません。

```
OPEN AP_Trans

COMMENT 累計の初期値を設定
ASSIGN v_running_total_hi = 0.00
ASSIGN v_running_total_med = 0.00
ASSIGN v_running_total_low = 0.00

COMMENT GROUP IF を使用して、請求金額に応じて別の ASSIGN および EXTRACT コマンドを実行
GROUP IF Invoice_Amount >= 1000
 ASSIGN v_running_total_hi = v_running_total_hi + Invoice_Amount
 EXTRACT Invoice_Number, Invoice_Amount, Invoice_Date, v_running_total_hi AS "Running total"
TO results_hi
ELSE IF Invoice_Amount >= 100
 ASSIGN v_running_total_med = v_running_total_med + Invoice_Amount
 EXTRACT Invoice_Number, Invoice_Amount, Invoice_Date, v_running_total_med AS "Running total"
TO results_med
ELSE
 ASSIGN v_running_total_low = v_running_total_low + Invoice_Amount
 EXTRACT Invoice_Number, Invoice_Amount, Invoice_Date, v_running_total_low AS "累計" TO
results_low
END
```

スクリプトの実行時には、GROUP コマンドは各レコードの請求金額をテストします。金額によっては、レコードは 3 つの合計 (低、中、高) のいずれかを更新するために使用され、3 つの結果テーブルが生成されます。

GROUP 内の LOOP

GROUP を使用してテーブルのレコードを処理するときには、LOOP コマンドを使用して、単一のレコードに対して一連のコマンドを複数回実行できます。LOOP は GROUP 内の繰り返し内で発生する 2 番目の繰り返しであり、指定するテスト条件が偽と評価されるまで実行されます。

LOOP を使用してフィールドを分割する

請求データを含む次のテーブルがあり、部署単位で請求金額の特定の情報を分離する必要があります。1 つの請求書は複数の部署に関連付けることができ、部署コードがカンマ区切りでテーブルに保存されます。

Vendor_Number	Invoice_Number	Invoice_Date	Invoice_Amount	Department_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

請求金額を部署ごとに抽出するには

1. GROUP コマンドを使用して、テーブルの各レコードを処理します。
2. 各レコードに関連付けられた部署 (*n*) 数を計算します。
3. LOOP コマンドを使用して、レコードに関連付けられた各部署のデータを抽出する操作を *n* 回繰り返します。

> **メモ**
>
> LOOP 内では v_counter 変数を増やす必要があります。そうしない場合、WHILE テストは常に真と評価され、スクリプトは無限ループに入ります。スクリプトに SET LOOP コマンドを含め、無限ループを防止できます。詳細については、"SET コマンド" ページ 404を参照してください。

```
COMMENT GROUP を使用して、各部署コード フィールドのカンマを、レコードに関連付けられた部署数を特定する方法としてカウントするフィールドの各コードの各レコードで "LOOP" し、各コードを独自のレコードに抽出する END GROUP v_department_count = OCCURS(Department_Code,',') v_counter = 0 LOOP WHILE v_counter <= v_department_count v_dept = SPLIT(Department_Code, ',', (v_counter + 1)) EXTRACTFIELDS Invoice_Number, Invoice_Amount, v_dept AS "Department" TO result1 v_counter = v_counter + 1 END END
```

スクリプトが実行されると、GROUP ブロック内のコマンドは上から下にテーブルの各レコードに対して処理されます。各レコードでは、LOOP コマンドがカンマ区切りのリストの部署コード単位で 1 回レコードに対して繰り返され、レコードを抽出します。実行される GROUP および LOOP コマンドを見ると、次のような仕組みになっています。

GROUP の 1 番目の繰り返し: LOOP の 2 つの繰り返し

Vendor_Number	Invoice_Number	Invoice_Date	Invoice_Amount	Department_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

テーブルの 1 番目のレコードでは、v_department_count の値が 1 になるため、LOOP は 2 回繰り返します。

1. LOOP の最初の繰り返し:
   - v_counter = 0
   - v_depart = CCD

   次のレコードが抽出され、v_counter の値が 1 になるため、LOOP はもう一度繰り返します。

5981807	618.30	CCD

2. LOOP の 2 回目の繰り返し:
   - v_counter = 1
   - v_depart = RDR

   次のレコードが抽出され、v_counter の値が 2 になるため、LOOP は繰り返さず、GROUP は次のレコードに進みます。

5981807	618.30	RDR

GROUP の 2 番目の繰り返し: LOOP の 1つの繰り返し

Vendor_Number	Invoice_Number	Invoice_Date	Invoice_Amount	Department_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

テーブルの 2 番目のレコードでは、v_department_count の値が 0 になるため、LOOP は 1 回繰り返します。

- v_counter = 0
- v_depart = CCD

次のレコードが抽出され、v_counter の値が 1 になるため、LOOP は繰り返さず、GROUP は次のレコードに進みます。

2275301	6705.12	CCD

GROUP の 3 番目の繰り返し: LOOP の 3 つの繰り返し

Vendor_Number	Invoice_Number	Invoice_Date	Invoice_Amount	Department_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

テーブルの 3 番目のレコードでは、v_department_count の値が 2 であるため、LOOP は 3 回繰り返します。

1. LOOP の最初の繰り返し:
   - v_counter = 0
   - v_depart = CCD

   次のレコードが抽出され、v_counter の値が 1 になるため、LOOP はもう一度繰り返します。

6585673	7955.46	CCD
2. LOOP の 2 回目の繰り返し:
   - v_counter = 1
   - v_depart = LMO

   次のレコードが抽出され、v_counter の値が 2 になるため、LOOP はもう一度繰り返します。

6585673	7955.46	LMO

3. LOOP の 3 回 目 の繰り返し:
   - v_counter = 2
   - v_depart = RDR

   次 のレコードが抽 出 され、v_counter の値 が 3 になるため、LOOP は繰り返 さず、GROUP はテーブルの最 後 に達します。

6585673	7955.46	RDR

最 終 結 果 テーブル

GROUP がテーブルの各 レコードを処 理し、LOOP がすべての部 署 コードを繰り返 した後 、次 の結 果 テーブルが生 成 されます。

Invoice_Number	Invoice_Amount	部門
5981807	618.30	CCD
5981807	618.30	RDR
2275301	6705.12	CCD
6585673	7955.46	CCD
6585673	7955.46	LMO
6585673	7955.46	RDR

最もよく使用される 30 個の Analytics 関数

ACLScript の上位 30 の関数は、さまざまなタスクで役に立ちます。これらの関数を定期的に使用し、スクリプトでデータを準備、解析、変換、および調整できます。

先頭と末尾のスペースを削除する

Analytics 関数 テーブルの文字フィールドは、一般的に、フィールド幅が固定長であるため、先頭または末尾にスペースがあります。文字フィールドからのデータを使用して演算を実行する必要があるときには、これらのスペースを削除し、文字列に実際のデータのみが含まれるようにします。

ALLTRIM( )

入力文字列から先頭と末尾のスペースを除去した文字列を返します。

> **メモ**
>
> 別の関数の入力として使用しているフィールドで ALLTRIM() を使用し、先頭または末尾のスペースが返される値に影響しないようにすることをお勧めします。

例

Vendor_Number フィールドには値 " 1254" があります。Vendor_Number からこの余分なスペースを削除し、フィールドを別のテーブルのデータと一致させる必要があります。

```
COMMENT "1254" を返します
ALLTRIM(Vendor_Number)
```

英字の大文字と小文字を同期する

Analytics の文字列比較は大文字と小文字を区別するため、データを使用して比較、結合、または関係を実行する前に、フィールドのすべてのデータの大文字と小文字を同期することが有用です。

UPPER( )

アルファベット文字を大文字に変換した文字列を返します。

例

Last_Name フィールドには値 "Smith" があります。この値を大文字にし、別のテーブルの大文字値と比較する

必要があります。

```
COMMENT "SMITH" を返します
UPPER(Last_Name)
```

LOWER( )

アルファベット文字を小文字に変換した文字列を返します。

例

Last_Name フィールドには値 "Smith" があります。この値を小文字にし、別のテーブルの小文字値と比較する必要があります。

```
COMMENT "smith" を返します
LOWER(Last_Name)
```

PROPER( )

各単語の最初の文字を大文字に、残りの文字を小文字に設定した文字列を返します。

例

Last_Name フィールドには値 "smith" があります。出力に固有名詞として表示する必要があります。

```
COMMENT "Smith" を返します
PROPER(Last_Name)
```

文字列の計算と分離

長い文字列からデータのセグメントを抽出するか、長さや内容などの文字列の情報をテストする必要があるときには、これらの関数を使用します。

SUBSTR( )

文字列のうちの指定された部分文字列を返します。

例

GL_Account_Code フィールドには値 "001-458-873-99" があります。最初の 3 バイトまたは 3 文字を文字列から抽出する必要があります。

```
COMMENT "001" を返します
ASSIGN v_start_pos = 1
ASSIGN v_length = 3
SUBSTR(GL_Account_Code, v_start_pos, v_length)
```

LAST( )

文字列の末尾から指定された数の文字を返します。

例

GL_Account_Code フィールドには値 "001-458-873-99" があります。最初の2バイトまたは2文字を文字列から抽出する必要があります。

```
COMMENT "99" を返します
ASSIGN v_num_chars = 2
LAST(GL_Account_Code, v_num_chars)
```

SPLIT( )

文字列のうちの指定された部分を返します。

例

GL_Account_Code フィールドには値 "001-458-873-99" があります。文字列からコードの2番目のセグメントを抽出する必要があります。

```
COMMENT "458" を返します
ASSIGN v_delimiter = "-"
ASSIGN v_segment_num = 2
SPLIT(GL_Account_Code, v_delimiter, v_segment_num)
```

AT( )

文字値における部分文字列の特定の出現の開始位置を示す数値を返します。

例

GL_Account_Code フィールドには値 "001-458-873-99" があります。値 "458" の開始バイト位置を決定し、GL コードの2番目のセグメントが "458" (開始位置 "5") であるかどうかをテストする必要があります。

```
COMMENT "5" を返します
ASSIGN v_occurrence = 1
```

```
ASSIGN v_substring = "458"
AT(v_occurrence, v_substring, GL_Account_Code)
```

OCCURS( )

部分文字列が指定された文字値内に現れる回数を数えて返します。

例

GL_Account_Code フィールドには値 "001-458-873-99" があります。データに3つのハイフン文字があることを確認し、GLコードが正しい形式であることを判定する必要があります。

```
COMMENT "3" を返します
ASSIGN v_substring = "-"
OCCURS(GL_Account_Code, v_substring)
```

LENGTH( )

文字列に含まれている文字数を返します。

例

GL_Account_Code フィールドには値 "001-458-873-99" があります。データに14文字があることを確認し、GLコードが正しい形式であることを判定する必要があります。

```
COMMENT "14" を返します
LENGTH(GL_Account_Code)
```

データ型の変換

Analytics テーブルを生成したデータソースとインポート文によっては、1つのデータ型から別のデータ型にフィールドの値を変換し、演算をできるようにする必要があります。たとえば、文字 ("12345") としてインポートされたデータで演算を実行するには、数値に変換する必要があります。

STRING( )

数値を文字列に変換します。

例

Invoice_Amount フィールドには値 12345.67 があります。これを文字データに変換する必要があります。

```
COMMENT "12345.67"を返します
ASSIGN v_str_length = 8
STRING(Invoice_Amount, v_str_length)
```

VALUE( )

文字列を数値に変換します。

> **ヒント**
> 一般的に、VALUE() は ZONED() とともに使用され、先頭のゼロを追加します。

例

Invoice_Amount フィールドには値 "12345.67" があります。これを数値データに変換する必要があります。

```
COMMENT 12345.67を返します
VALUE(Invoice_Amount)
```

CTOD( )

文字または数値の日付値を日付に変換します。また、文字または数値の日付時刻値から日付を抽出し、それを日付として返すこともできます。"Character to Date" の省略形です。

例

Submission_Date フィールドには値 "April 25, 2016" があります。これを日付時刻データに変換する必要があります。

```
COMMENT `20160425`を返します
ASSIGN v_date_format = "mmm dd, yyyy"
CTOD(Submission_Date, v_date_format)
```

DATE( )

指定された日付または日付時刻から日付を抽出し、それを文字データとして返します。また、現在のオペレーティングシステム日付を返すこともできます。

例

Submission_Date フィールドには値 `20160425` があります。これを文字データに変換する必要があります。

```
COMMENT "04/25/2016"を返します
ASSIGN v_date_format = "MM/DD/YYYY"
DATE(Submission_Date, v_date_format)
```

先頭のゼロの追加

先頭のゼロが必要なフィールドと一致させる必要があるときに、数値データを文字データに変換し、出力に先頭のゼロを追加します。

ZONED( )

数値データを文字データに変換し、出力の先頭にゼロを追加します。

例

Employee_Number フィールドには値 "254879" があります。値を先頭ゼロ埋めの 10 桁の文字列に変換する必要があります。

> **ヒント**
>
> VALUE() 関数を使用して文字を数値データに変換してから、ZONED() への入力値として数値を使用します。

```
COMMENT "0000254879" を返します
ASSIGN v_str_length = 10
ASSIGN v_num_decimals = 0
ZONED(VALUE(Employee_Number, v_num_decimals), v_str_length)
```

BINTOSTR( )

ZONED または EBCDIC 文字データから変換された Unicode 文字データを返します。"Binary to String" の省略形です。

> **メモ**
>
> Unicode 版のみ。非 Unicode 版については、上記の ZONED() を参照してください。

例

Employee_Number フィールドには値 "254879" があります。値を先頭ゼロ埋めの 10 桁の文字列に変換する必要があります。

> **ヒント**
>
> VALUE() 関数を使用して文字を数値データに変換してから、ZONED() への入力値として数値を使用します。そこで BINTOSTR() を使用し、ZONED() から返された ASCII データを Unicode に変換します。

```
COMMENT "0000254879" を返します
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)
```

日付時刻部分の抽出

これらの関数を使用して、日付時刻値の特定のコンポーネントを分離および抽出します。

MONTH( )

指定された日付または日付時刻から月を抽出し、それを数値 (1 ～ 12) として返します。

例

Transaction_Date フィールドには値 `20160815 100252` があります。先頭にゼロがある文字データとして月を抽出する必要があります。

```
COMMENT "08" を返します
ASSIGN v_str_length = 8
ZONED(MONTH(Transaction_Date), v_str_length)
```

DAY( )

指定された日付または日付時刻から日にちを抽出し、それを数値 (1 ～ 31) として返します。

例

Transaction_Date フィールドには値 `20160815 100252` があります。文字データとして日を抽出する必要があります。

```
COMMENT "15" を返します
ASSIGN v_str_length = 8
STRING(DAY(Transaction_Date), v_str_length)
```

YEAR( )

指定された日付または日付時刻から年を抽出し、それをYYYY書式の数値として返します。

例

Transaction_Dateフィールドには値 `20160815 100252` があります。数値として年を抽出する必要があります。

```
COMMENT 2016を返します
YEAR(Transaction_Date)
```

HOUR( )

指定された時刻または日付時刻から時間を抽出し、それを24時間制の数値として返します。

例

Transaction_Dateフィールドには値 `20160815 100252` があります。数値として時間を抽出する必要があります。

```
COMMENT 10を返します
HOUR(Transaction_Date)
```

MINUTE( )

指定された時刻または日付時刻から分数を抽出し、それを数値として返します。

例

Transaction_Dateフィールドには値 `20160815 100252` があります。数値として分を抽出する必要があります。

```
COMMENT 2を返します
MINUTE(Transaction_Date)
```

SECOND( )

指定された時刻または日付時刻から秒数を抽出し、それを数値として返します。

例

Transaction_Dateフィールドには値 `20160815 100252` があります。数値として秒を抽出する必要があります。

```
COMMENT 52を返します
SECOND(Transaction_Date)
```

CDOW( )

指定された日付または日付時刻の曜日を返します。"Character Day of Week" の省略形です。

例

Transaction_Date フィールドには値 `20160815 100252` があります。文字データとして日の名前を抽出する必要があります。

```
COMMENT "Mon" を返します
CDOW(Transaction_Date, 3)
```

CMOY( )

指定された日付または日付時刻の月の名前を返します。"Character Month of Year" の省略形です。

例

Transaction_Date フィールドには値 `20160815 100252` があります。文字データとして月の名前を抽出する必要があります。

```
COMMENT "Aug" を返します
CMOY(Transaction_Date, 3)
```

文字列の操作

これらの関数を使用して、文字フィールドのセグメントを削除または置換します。

INCLUDE( )

指定した文字のみを含む文字列を返します。

例

Address フィールドには値 "12345 ABC Corporation" があります。住所番号を抽出し、会社の名前を除外する必要があります。

```
COMMENT "12345"を返します
ASSIGN v_chars_to_return = "0123456789"
INCLUDE(Address, v_chars_to_return)
```

EXCLUDE( )

指定した文字を除外する文字列を返します。

例

Address フィールドには値 "12345 ABC Corporation" があります。会社名を抽出し、住所番号を除外する必要があります。

```
COMMENT "ABC Corporation" を返します
ASSIGN v_chars_to_exclude = "0123456789"
EXCLUDE(Address, v_chars_to_exclude)
```

REPLACE( )

指定された文字列のすべてのインスタンスを新しい文字列で置き換えます。

例

Address フィールドには値 "12345 Acme&Sons" があります。"&" 文字を単語 " and " で置換する必要があります。

```
COMMENT "12345 Acme and Sons"を返します
ASSIGN v_target_char = "&"
ASSIGN v_replacement_char = " and "
REPLACE(Address, v_target_char, v_replacement_char)
```

OMIT( )

指定した 1 つ以上の部分文字列が削除された文字列を返します。

例

Address フィールドには値 "12345 Fake St" があります。番地の接尾辞がない住所を抽出する必要があります。

```
COMMENT "12345 Fake"を返します
ASSIGN v_chars_to_omit = "St"
OMIT(Address, v_chars_to_omit)
```

REVERSE( )

文字の順番を逆にした文字列を返します。

例

Report_Line フィールドには値 "001 Correction 5874.39 CR " があります。値を逆にし、先頭または末尾のスペースを省略する必要があります。

```
COMMENT "RC 93.4785 noitcerroC 100" を返します
REVERSE(ALLTRIM(Report_Line))
```

BLANKS( )

指定された数の空白スペースを含んでいる文字列を返します。

例

region_code フィールドの値に基づいて、領域名の演算フィールドを作成する必要があります。コマンドの最後に指定するデフォルト値が少なくとも最も長い入力値と同じ長さであることを確認する必要があります。

```
COMMENT BLANKS 8 " " 文字の文字列を返します
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)
```

コマンド

ACCEPT コマンド

1 つ以上のスクリプト入力値をインタラクティブにユーザーに確認するダイアログボックスを作成します。各入力値は名前付き文字変数に格納されます。

> **メモ**
>
> ACCEPT コマンドを使用してパスワードを入力することは安全ではありません。代わりに "PASSWORD コマンド" ページ 345 を使用してください。
>
> ACCEPT コマンドは、ACL Server 分析ではサポートされません。
>
> "DIALOG コマンド" ページ 145では高度なインタラクティブ ダイアログボックスを作成できます。

構文

ACCEPT {*メッセージテキスト* <FIELDS *プロジェクト項目カテゴリ*> TO *変数名*} <...*n*>

パラメーター

名前	説明
メッセージ テキスト	入力するよう指示するために使用されるテキストとして、ダイアログ ボックスに表示されるラベルです。引用符で囲んだ文字列か、文字型変数を指定する必要があります。 複数のプロンプトを入力する場合は、カンマで区切ることができます。カンマを使用すると、スクリプト判読性を向上させますが、必須ではありません。 `ACCEPT "開始日を指定してください:" TO v_start_date, "終了日を指定してください:" TO v_end_date`
FIELDS *プロジェクト項目カテゴリ* 省略可能	テキスト ボックスではなく、ユーザー入力用のプロジェクト項目ドロップダウン リストを作成します。ユーザーはリストから、プロジェクト項目、フィールド、または変数の 1 つの値を選ぶことができます。 *プロジェクト項目カテゴリ*は、リストに表示する項目タイプを指定します。たとえば、xf と指定すると、リストにすべてのプロジェクト テーブルが表示されます。*プロジェクト項目カテゴリ*は引用符で囲みます。 `FIELDS "xf"` カテゴリを指定するために使用するコードについては、"プロジェクト項目カテゴリーのコード" ページ 54を参照してください。 同じプロンプトで複数のコードを指定できますが、プロジェクト項目、フィールド、変数を混在させることはできません。
TO *変数名*	ユーザー入力を格納するために使用する文字変数の名前。変数自体が存在していない場合は、変数が作成されます。

名前	説明
	この変数が既に存在する場合は、現在の値がデフォルト値としてダイアログ ボックスに表示されます。 **メモ** 代入変数に使用される変数の名前に、é のような英語以外の文字は使用しないでください。変数名に英語以外の文字が含まれていると、スクリプト エラーになります。 ACCEPT コマンドは文字変数のみを作成します。別のデータ型の入力が必要な場合は、スクリプトの後の処理で、文字変数を必要な型に変換する必要があります。詳細については、"入力データ型" ページ 55を参照してください。

例

ユーザーに対し、開くAnalytics テーブルを選択するよう求める

開くテーブルの名前を選択するようユーザーに求めるダイアログ ボックスが必要です。ユーザーが選択したテーブルをスクリプトによって開きます。

```
ACCEPT "開くテーブルの選択:" FIELDS "xf" TO v_table_name
OPEN %v_table_name%
```

パーセント記号を使用する理由は、開こうとするテーブルの名前が *v_table_name* 変数に格納されていることを示すためです。パーセント記号を省略した場合には、スクリプトは "v_table_name" というテーブルを開こうとします。

必要な入力を収集するために多数のダイアログ ボックスを使用する

スクリプト ユーザーが入力する必要がある各値のダイアログ ボックスを個別に作成したいとします。

ACCEPT コマンドの各インスタンス内に1つのプロンプト文字列を指定します。スクリプトにより、以下の各項目を入力するダイアログボックスを別々に生成します。

- テーブル名
- サンプリングに使用するフィールド
- サンプリング間隔
- ランダムな開始値

```
ACCEPT "分析するテーブル名を入力してください" TO v_table_name
OPEN %v_table_name%
ACCEPT "サンプルを抽出するフィールドを選択してください" FIELDS "N" to v_field_to_sample
ACCEPT "サンプリング間隔を入力してください" TO v_sampling_interval
ACCEPT "ランダム開始値を入力してください" TO v_random_start_value
SAMPLE ON %v_field_to_sample% INTERVAL v_sampling_interval FIXED v_random_start_value
RECORD TO Sample_output OPEN
```

スクリプトを実行すると、次の動作が行われます。

1. 最初のダイアログボックスは、テーブルの名前を入力するように求めます。
2. 2番目のダイアログボックスでは、FIELDS "N" によって、数値フィールドのドロップダウンリストからのフィールドの選択を求めます。
3. 3番目のダイアログボックスでは、サンプリング間隔値の入力を求めます。
4. 4番目のダイアログボックスでは、ランダム開始値の入力を求めます。

必要な入力を収集するために多数のプロンプトを備えた単一のダイアログボックスを使用する

スクリプト ユーザーが入力する必要があるすべての値のダイアログボックスを作成したいとします。

ユーザーに複数の入力値を要求する場合には、ACCEPT コマンド内に複数のプロンプトをカンマで区切って指定します。次の例では、日付範囲の開始日と終了日のプロンプトが同じダイアログボックスに表示されます。

```
ACCEPT "開始日を指定してください:" TO v_start_date, "終了日を指定してください:" TO v_end_date
```

備考

インタラクティブ

インタラクティブ スクリプトを作成するには、ACCEPT コマンドを使用します。ACCEPT コマンドが処理されるときにはスクリプトが中断し、Analytics が次の処理で使用する入力をユーザーに促すダイアログボックスが表示されます。

項目ごとに個別のダイアログボックスを作成して、一度に1項目について入力を求めることもできますが、1つのダイアログボックスで複数項目について入力を求めることもできます。

DIALOG と ACCEPT

DIALOG コマンドでは、次の1つ以上のタイプのコントロールを使用できるより高度なインタラクティブ ダイアログボックスを作成できます。

- テキストボックス
- チェックボックス
- ラジオボタン
- カスタマイズされた値のドロップダウンリスト
- プロジェクト項目一覧

ダイアログボックスのレイアウトは柔軟にカスタマイズできます。詳細については、"DIALOG コマンド" ページ 145 を参照してください。

プロジェクト項目カテゴリーのコード

次のコードを使用して、ドロップダウンリストで表示するプロジェクト項目のカテゴリーを指定します。

プロジェクト カテゴリー

コード	カテゴリー
xf	テーブル
xb	スクリプト
xi	インデックス
xr	ビューとレポート
xw	ワークスペース

フィールド カテゴリー

コード	カテゴリー
C	文字フィールド
N	数値フィールド
D	日付時刻フィールド
L	論理フィールド

変数カテゴリ

コード	カテゴリー
c	文字変数
n	数値変数
d	日付時刻変数
l	論理変数

入力データ型

ACCEPT はユーザー入力を 1 つ以上の文字変数として保存します。数値や日付時刻値として保存するには、VALUE( ) 関数や CTOD( ) 関数を使用します。文字変数の値がそれぞれ数値や日付時刻値に変換されます。

```
SET FILTER TO BETWEEN(%v_date_field%, CTOD(%v_start_date%), CTOD(%v_end_date%))
```

この例では、フィルターの開始日付および終了日付は文字値として保存されています。これらの日付は、日付時刻データ型を使用する日付フィールドで使用するには、日付型の値に変換する必要があります。

変数名をパーセント記号 (%) で囲むと、変数名の変数によって含まれる文字値を代入します。CTOD( ) 関数は文字値を日付値に変換します。

ACCEPT コマンドの位置

可能な限り、ACCEPT コマンドはすべてスクリプトの先頭部分に配置することをお勧めします。最初にすべての情報入力を要求し、必要な情報が入力されていれば、その後スクリプトを円滑に実行できます。

> **メモ**
>
> GROUP コマンド内では ACCEPT コマンドを使用できません。

ACCESSDATA コマンド

さまざまな ODBC 準拠データソースからデータをインポートします。

64 ビットと 32 ビットのどちらの ODBC ドライバーを使用するかによって、このコマンドは ACCESSDATA64 または ACCESSDATA32 という形を取ります。

構文

{ACCESSDATA64 | ACCESSDATA32} {CONNECTOR | ODBC {"Driver"|"Dsn"|"File"}} NAME *値* <USER *ユーザー ID*> <PASSWORD *数値* | PROMPT_PASSWORD> TO *テーブル名* CHARMAX *最大フィールド長* MEMOMAX *最大フィールド長* ALLCHARACTER SOURCE (*接続設定*) <HASH(*ソルト値*, *フィールド*)>
SQL_QUERY
(*SQL 構文*)
END_QUERY

パラメーター

名前	説明
CONNECTOR | ODBC {"Driver"|"Dsn"|"File"}	行いたい ODBC 接続の種類。 ◦ **CONNECTOR** - - ネイティブの Analytics データコネクターを使って接続する ◦ **ODBC "Driver"** - - お使いのコンピューターにインストールされている Windows ODBC ドライバーを使って接続する ◦ **ODBC "Dsn"** - - お使いのコンピューターに保存されている DSN(データソース名) を使って接続する ◦ **ODBC "File"** - - DSN ファイル(保存されている .dsn ファイル) を使って接続する
NAME *値* 省略可能	Analytics データコネクター、ODBC ドライバー、または DSN の名前。 例 : ◦ NAME "Amazon Redshift" ◦ NAME "Microsoft Access Driver (*.mdb, *.accdb)" ◦ NAME "My Excel DSN" ◦ NAME "excel.dsn"
USER *ユーザー ID* 省略可能	ユーザー ID が必要なデータソースのユーザー ID。
PASSWORD *数値* | PROMPT_PASSWORD 省略可能	パスワードが必要なデータソース: ◦ **PASSWORD 数値** - - 数値のパスワード定義を使用します。 ◦ **PROMPT_PASSWORD** - - パスワード プロンプトを表示します。 パスワード プロンプトでは、*ユーザー ID* を変更できます。

名前	説明
	PASSWORD *数値*を使用する場合は、以前に作成したパスワード定義を指定する必要があります。詳細については、"PASSWORD コマンド" ページ 345と"SET コマンド" ページ 404を参照してください。 **ヒント** PASSWORD コマンドを PASSWORD パラメーター*num* とともに使用すると、PROMPT_PASSWORD の使用と同様の結果になります。両方のアプローチでユーザーにパスワードを要求します。PROMPT_PASSWORD にはユーザー *ID* を変更できるという利点があります。
TO *テーブル名*	コマンドの結果を送信する場所： ◦ ***テーブル名*** -は、結果の保存先となる Analytics テーブルのことです。 *テーブル名*には、ファイル拡張子 .FIL を付けた文字列を引用符で囲んで指定する必要があります。例：TO "Output.FIL" デフォルトでは、テーブル データ ファイル(.fil) は、Analytics プロジェクトが入っているフォルダーに保存されます。 既存の異なるフォルダーにデータ ファイルを保存するには、絶対または相対ファイル パスを指定します。 • TO "C:\Output.FIL" • TO "Results\Output.FIL" **メモ** テーブル名は 64 文字の英数字(.FIL 拡張子を含まない) に制限されています。名前にはアンダースコア文字(_) を使用できますが、他の特殊文字やスペースは使用できません。名前の先頭を数字にすることはできません。
CHARMAX *最大フィールド長*	インポートしているソースの文字データから発生する Analytics テーブルの任意のフィールドの文字の最大長。 デフォルト値は 50 です。最大フィールド長を超えるデータは、Analytics にインポートすると切り捨てられます。
MEMOMAX *最大フィールド長*	インポートするテキスト、ノート、またはメモ フィールドの文字の最大長。 デフォルト値は 100 です。最大フィールド長を超えるデータは、Analytics にインポートすると切り捨てられます。
ALLCHARACTER 省略可能	インポートされたすべてのフィールドには、自動的に文字のデータ型が割り当てられます。 Analytics にインポートされたデータのフィールドには、数値や日付時刻などのさまざまなデータ型を割り当て、書式の詳細を指定することができます。 **ヒント** ALLCHARACTER は、数値 ID 値を含むテーブルをインポートする際に有効な場合があります。ALLCHARACTER を使用すると、Analytics は自動的に数値データ型を、文字データ型を使用すべき値に割り当てることができません。
SOURCE *接続設定*	データ ソースへの接続に必要な接続設定(接続文字列) 。
HASH(*salt 値*, *フィールド*) 省略可能	暗号ハッシュ値として指定されたフィールドをインポートします。ハッシュ値は単方向の変換であり、フィールドのインポート後に復号化できません。

名前	説明
	◦ ***salt 値*** - フィールドの値のハッシュを強化するためにソースデータ値と連結された英数字文字列。引用符で囲まれた文字列としてハッシュ値を入力します。 ソルト値は 128 文字以下です。( ) " 文字は使用しないでください。 ◦ ***フィールド*** - ハッシュ化する 1 つ以上のフィールドのリスト。引用符で囲まれた文字列としてフィールドを入力し、各フィールドをカンマで区切ります。 データソースの物理フィールド名ではなく、[データアクセス]ウィンドウプレビューとステージング領域に表示されるフィールド名を指定する必要があります。 **メモ** [データアクセス]ウィンドウプレビューに表示されるフィールド名は、SQL クエリ("フィールド名" AS "エイリアス") のフィールド エイリアス値です。エイリアス値を使用して、フィールドを参照する必要があります。 `HASH("QZ3x7", "SSN_NO, CC_NO, Last_Name")` ACLScript でのハッシュ値のインポート中におけるハッシュ値の比較については、"ACCESSDATA でハッシュ化されたデータを ACLScript HASH( ) 関数でハッシュ化されたデータと比較する" ページ 62を参照してください。
SQL_QUERY (*SQL構文*) END_QUERY	SQL のインポート文。 丸括弧内はすべて SQL クエリの構成要素のため有効な SQL である必要があります。

例

ネイティブの Analytics データコネクターを使用してデータをインポートする

Amazon Redshift クラウド データ サービスからのデータをインポートする必要があるとします。そうするには、Analytics Amazon Redshift データコネクターを使用します。

```
ACCESSDATA64 CONNECTOR NAME "Amazon Redshift" USER "ACL_user" PROMPT_
PASSWORD TO "Entitlement_History.FIL" CHARMAX 50 MEMOMAX 100
 SOURCE(
boolsaschar=0;cachesize=100;database=usage;declarefetchmode=0;maxbytea=255;maxlongvarchar-
=8190;maxvarchar=255;port=5439;servername=acl_
test.highbond.com;singlerowmode=1;sslmode=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
```

Windows ODBC ドライバーを使用してデータをインポートする

Microsoft Access データベースからデータをインポートする必要があるとします。インポートを行うには、Windows ODBC ドライバーを使用して MS Access に接続します。

```
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;maxbuffersize=2048;maxscanrows=8;pagetimeout=5;safetransactions=0;threads=3;userco-
mmitsync=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
```

Windows DSN(データソース名) を使 用してデータをインポートする

Microsoft Excel ファイルからデータをインポートする必 要 があるとします。 インポートを行うには、Windows DSN を使 用して Excel に接 続します。

```
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
```

備考

> **メモ**
>
> このコマンドの動 作 の詳 細 については、Analytics のヘルプを参 照してください。

ODBC 接 続 設 定とSQL インポート文 の作 成

ODBC 接 続 設 定とSQL インポート文 は例 に示 すように通 常 、きわめて長く複 雑 です。

ACCESSDATA コマンドのこれらの部分を最も簡単に作成するには、まず Analytics の Data Access ウィンドウを使ってターゲットのデータソースに接続してからデータをインポートします。次に、ログから、接続設定やインポート文など ACCESSDATA コマンド全体をコピーし、必要に応じてこのコマンドをカスタマイズします。

抑制されるパスワード値

Analytics のデータアクセスウィンドウを使用して ACCESSDATA コマンドを実行し、パスワードを入力すると、パスワード値はログに記録されません。PROMPT_PASSWORD パラメーターが代用されます。

ACCESSDATA ログファイル

2 つのログファイルは ACCESSDATA コマンドに関連付けられたトランザクションを記録し、データ接続が失敗した場合にトラブルシューティングで使用できます。

- **ServerDataAccess.log** - はデータをインポートする前のすべての作業とエラーを記録します

  場所：`C:\Users\<ユーザー アカウント>\AppData\Local\ACL\ACL for Windows\Data Access\ServerDataAccess.log`

  > **メモ**
  >
  > `ServerDataAccess.log` の "Server" は Analytics がインストールされているコンピューターでローカルに実行されている Analytics のデータアクセスコンポーネントを参照します。

- **DataAccess.log** - には、インポート操作に関する情報と、

  場所 `..\<Analytics プロジェクト フォルダー>\DataAccess.log`
  にインポートしようとしているデータが含まれる Analytics プロジェクトに関する情報が記録されます。

ACCESSDATA でハッシュ化されたデータを ACLScript HASH( ) 関数でハッシュ化されたデータと比較する

ハッシュデータの未加工値を読み取れない場合でも、データの結合または分析時に役立ちます。

インポート中に ACCESSDATA でハッシュ化された値を、ACLScript の HASH( ) 関数を使用してハッシュ化された値と比較する場合は、Analytics の数値または日付時刻フィールドを文字値に変換し、データをハッシュ化する前に前後のスペースを切り取る必要があります。

文字に変換するときには、日付時刻フィールドは次の形式を使用する必要があります。

- **日付時刻** - "YYYY-MM-DD hh:mm:ss"
- **日付** - "YYYY-MM-DD"
- **時刻** - "hh:mm:ss"

次の例は、STRING( ) および ALLTRIM( ) 関数を使用して、ACLScript's HASH( ) 関数を使用して値をハッシュ化する前に、数値のクレジットカード番号フィールドをを文字データに変換します。

```
COMMENT ACL HASH 関数は、データのインポート後に使用されます
HASH(ALLTRIM(STRING(CC_No, 16)), "QZ3x7")
```

Analytics 値をハッシュ化すると、ACCESSDATA コマンド インポートの一部としてハッシュ化された値を比較できます。

ACTIVATE コマンド

Analytics ワークスペースに保存されたフィールド定義を Analytics テーブルレイアウトのフィールド定義の既存のセットに追加します。

構文

```
ACTIVATE <WORKSPACE> ワークスペース名 <OK>
```

パラメーター

名前	説明
WORKSPACE *ワークスペース名*	アクティブにするワークスペースの名前。
OK 省略可能	アクションを確認せずに、項目を削除または上書きします。 アクティブ化されたワークスペースのフィールドと同じ名前のテーブルのフィールドがある場合は、確認なしで上書きされます。ただし、演算フィールドが参照しているフィールドを置き換えることはできません。

例

Analytics プロジェクトでワークスペースをアクティブにする

ComplexFormulas ワークスペースをアクティブにします。

```
ACTIVATE WORKSPACE ComplexFormulas OK
```

Analytics プロジェクトと同じフォルダー内のファイル(.wsp) として保存されたワークスペースをアクティブにする

.wsp ファイルとして保存された **ComplexFormulas** ワークスペースをアクティブにします。

```
ACTIVATE WORKSPACE ComplexFormulas.WSP OK
```

備考

機能の仕組み

ACTIVATE は、ワークスペースのフィールド定義をアクティブなテーブルで使用できるようにします。ワークスペースをアクティブにしたら、アクティブ テーブルを閉じない限りは、テーブルでワークスペースのフィールドを使用できます。

テーブル レイアウトの編集

次の場合、ワークスペースのフィールドはテーブル レイアウトに永続的に追加されます。

- ワークスペースをアクティブにした後でテーブル レイアウトを編集した場合
- テーブル レイアウトが保存されるような変更を行った場合

ワークスペースのフィールドをテーブル レイアウトに保存すると、以下の操作を行えるようになります。

1. DEFINE COLUMN コマンドを使ってビューにそれらのフィールドを追加する
2. SAVE コマンドを使って変更を保存する

AGE コマンド

日付または日付時刻フィールドの値に基づいて、レコードを年齢調べ間隔でグループ化します。各期間のレコード数をカウントし、指定した数値フィールドの小計を期間ごとに求めます。

構文

AGE <ON> *日付フィールド* <CUTOFF *締切日*> <INTERVAL *日* <,...*n*>> <SUPPRESS> <SUBTOTAL *数値フィールド* <...*n*>|SUBTOTAL ALL> <IF *テスト*> <WHILE *テスト*> <FIRST *範囲*|NEXT *範囲*> <TO SCREEN|*ファイル名*|GRAPH|PRINT> <KEY *内訳フィールド*> <HEADER *ヘッダーテキスト*> <FOOTER *フッターテキスト*> <APPEND> <LOCAL> <STATISTICS>

パラメーター

名前	説明
ON *日付フィールド*	年齢調べに用いる日付時刻フィールドまたは式の名前。 日付時刻フィールドを対象として年齢調べを行うことはできますが、日付時刻値の日付部分しか考慮されません。時刻部分は無視されます。時刻データだけで年齢調べを行うことはできません。
CUTOFF *締切日* 省略可能	*日付フィールド*の値と比較される日付。 *締切日*は、日付フィールドの形式に関係なく、引用符で囲まれていない YYMMDD または YYYYMMDD の形式で指定する必要があります。例: CUTOFF 20141231 締切日を省略した場合は、締切日として、現在のシステム日付が使用されます。
INTERVAL *日数* <,...*n*> 省略可能	年齢調べ間隔の計算で使用する日付の間隔(つまり、日数)。 *日数*は、*締切日*から過去に遡る各年齢調べ間隔の始まりを表します。 ◦ 最初の*日数*値は、最初の年齢調べ間隔の始まりを決定します。 ◦ 最初の*日数*値を '0' にした場合、最初の年齢調べ間隔は指定した*締切日*から始まります。 ◦ 最後の*日数*値は、最終の年齢調べ間隔の終わりを決定します。 間隔は、引用符で囲まれていないカンマ区切りの文字列値として指定する必要があります。 `INTERVAL 0,90,180,270,365` デフォルトの年齢調べ間隔は "0"、"30"、"60"、"90"、"120"、"10,000" 日です。10,000 日の期間は、無効と思われる日付が入っているレコードを分離するために使用されます。 必要に応じて、ほかの内部年齢調べレポートを反映するように、日付の間隔をカスタマイズすることができます。

名前	説明
SUPPRESS 省略可能	指定した年齢調べ間隔に該当しない日付をコマンドの出力から除外します。
SUBTOTAL *数値フィールド <...n>* | SUBTOTAL ALL 省略可能	グループごとに小計を計算する 1 つ以上の数値フィールドまたは式。 複数のフィールドはスペースで区切る必要があります。テーブル内のすべての数値フィールドについて小計を求める場合は ALL を指定します。
IF *テスト* 省略可能	各レコードを処理するために真である必要がある条件式。コマンドは、その条件を満たすレコードに対してのみ実行されます。 **メモ** IF パラメーターは、任意の範囲パラメーター(WHILE、FIRST、NEXT) が適用された後に、テーブルに残るレコードに対してのみ評価されます。
WHILE *テスト* 省略可能	各レコードを処理するために真である必要がある条件式。条件が false と評価するか、テーブルの最後に達したら、コマンドは実行を中止します。 **メモ** WHILE を FIRST または NEXT とともに使用する場合は、1 つの制限に達するとすぐに、レコードの処理が停止します。
FIRST *範囲* | NEXT *範囲* 省略可能	処理するレコード数: ◦ **FIRST** - 指定されたレコード数に達するまで、最初のレコードから処理を開始します ◦ **NEXT** - 指定されたレコード数に達するまで、現在選択されているレコードから処理を開始します *範囲*は処理するレコード数を指定します。 FIRST と NEXT を省略すると、すべてのレコードがデフォルトで処理されます。
TO SCREEN | *ファイル名* | GRAPH | PRINT	コマンドの結果を送信する場所: ◦ **SCREEN** - は Analytics の表示領域に結果を表示します ◦ ***ファイル名*** -は結果の保存先となるファイルです。 *ファイル名*には、適切なファイル拡張子を付けた文字列を引用符で囲んで指定します。例: `TO "Output.TXT"` デフォルトでは、テーブル ファイルは、Analytics プロジェクトが入っているフォルダーに保存されます。 既存の異なるフォルダーにファイルを保存するには、絶対または相対ファイル パスを指定します。 • `TO "C:\Output.TXT"` • `TO "Results\Output.TXT"` ◦ **GRAPH** - は結果をグラフに表示し、それを Analytics の表示領域に表示します ◦ **印刷** -- 通常使うプリンターに結果を送信します
KEY *ブレークフィールド* 省略可能	小計計算をグループ化するフィールドまたは式。*ブレークフィールド*の値が変わるたびに、小計が計算されます。 *ブレークフィールド*は、文字フィールドか式である必要があります。指定できるフィールドは 1 つだけですが、1 つ以上のフィールドを含んでいる式を使用することができます。

名前	説明
HEADER *ヘッダーテキスト* 省略可能	レポートの各ページの最上部に挿入されるテキスト。 *ヘッダーテキスト*は引用符で囲んだ文字列として指定する必要があります。この値は、Analytics のシステム変数である HEADER の値よりも優先されます。
FOOTER *フッターテキスト* 省略可能	レポートの各ページの最下部に挿入されるテキスト。 *フッターテキスト*は引用符で囲んだ文字列として指定する必要があります。この値は、Analytics のシステム変数である FOOTER の値よりも優先されます。
APPEND 省略可能	コマンドの出力を既存ファイルに上書きしないで、そのファイルの末尾に追加します。 **メモ** コマンドの出力と既存のファイルの構造が同一であるようにする必要があります。 • 同じフィールド • 同じフィールド順序 • 一致するフィールドが同じ長さ • 一致するフィールドが同じデータ型 出力は、既存ファイルとの間でファイル構造が違っている場合でも、Analytics によって既存ファイルに追加されます。出力と既存のファイルの構造が一致しない場合は、データが混在、不足、不正確になります。
LOCAL 省略可能	Analytics プロジェクトと同じ場所に出力ファイルを保存します。 **メモ** Analytics テーブルである出力ファイルを含むサーバーテーブルに対してコマンドを実行するときにのみ適用されます。
STATISTICS 省略可能	**メモ** SUBTOTAL も指定されていない場合は使用できません。 すべての SUBTOTAL フィールドの平均値、最小値、および最大値を計算します。

例

小計された金額がある請求書の年齢調べ

Invoice_Date(請求日) フィールドに基づいて Ar(売掛金) テーブルの年齢調べを行い、**Invoice_Amount** (請求金額) フィールドの小計を求めたいとします。

請求書は以下の 30 日の期間によってグループ化されます。

- 締切日からその 29 日前まで
- 締切日の 30 日前から 59 日前まで
- といった具合です。

次のコマンドにより、各期間の未払い請求金額の合計が出力されます。

```
OPEN Ar
AGE ON Invoice_Date CUTOFF 20141231 INTERVAL 0,30,60,90,120,10000 SUBTOTAL Invoice_
Amount TO SCREEN
```

備考

> **メモ**
> このコマンドの動作の詳細については、Analytics のヘルプを参照してください。

年齢調べ間隔

AGE コマンドは、日付または日付時刻フィールドに基づいて、レコードを年齢調べ間隔でグループ化します。出力結果には、期間ごとに 1 つのレコードが含まれ、各レコードには、ソーステーブル内でその期間に該当するレコードの数が含まれます。

間隔測定

年齢調べ間隔は、現在のシステム日付、あるいは会計年度末の日付など、指定した締切日から過去に遡る日数の間隔に基づきます。

将来の期間

日付の間隔に負の値を入力することによって、締切日より後の年齢調べ間隔を作成することができます。たとえば、次のコマンドでは、締切日から経過する年齢調べの間隔 (マイナスの年齢調べ間隔) と、締切日から遡る年齢調べの間隔が作成されます。

```
INTERVAL -60,-30,0,30,60,90
```

この手法でさまざまな時点を使用すれば、テーブル内の全レコードの日付を含んだ日付プロファイルを作成することができます。

一般的な用途

年齢調べは、販売傾向の評価、取扱量の調査、未払い日数による請求書のグループ化などでよく使用されます。

Analytics は、指定された年齢調べ間隔に該当しない日付用として、1 つまたは 2 つの追加年齢調べ間隔を自動的に作成します。ただし、これは SUPPRESS を使用していないことを前提とします。

APPEND コマンド

2 つ以上の Analytics テーブルからのレコードを結合するには、それらのレコードを新しい Analytics テーブルに追加します。

構文

APPEND *テーブル1*, *テーブル2*, *<...n>* TO *テーブル名* <OPEN> <ASCHAR> <ALLCHAR>

パラメーター

名前	説明
テーブル1, *テーブル2*, *<...n>*	追加するテーブル。 各テーブル内のレコードが、テーブルを指定した順に追加されます。出力テーブルにはテーブル *1* のレコード、テーブル*2* のレコード、という順に格納されます。 各ソース テーブルは、レコード構造が同じでも異なっていてもどちらでも構わず、また、並べ替えても並べ替えなくても構いません。
TO *テーブル名*	コマンドの結果を送信する場所： ◦ ***テーブル名*** -は、結果の保存先となる Analytics テーブルのことです。 *テーブル名*には、ファイル拡張子 .FIL を付けた文字列を引用符で囲んで指定する必要があります。例：`TO "Output.FIL"` デフォルトでは、テーブル データ ファイル(.fil) は、Analytics プロジェクトが入っているフォルダーに保存されます。 既存の異なるフォルダーにデータ ファイルを保存するには、絶対または相対ファイル パスを指定します。 • `TO "C:\Output.FIL"` • `TO "Results\Output.FIL"` **メモ** テーブル名は 64 文字の英数字(.FIL 拡張子を含まない) に制限されています。名前にはアンダースコア文字(_) を使用できますが、他の特殊文字やスペースは使用できません。名前の先頭を数字にすることはできません。
COMMONFIELDS 省略可能	追加対象となるすべてのテーブルに共通のフィールドのみが、出力テーブルに追加されます。 COMMONFIELDS を省略した場合は、すべてのテーブルの全フィールドが出力テーブルに追加されます。ソース テーブルにフィールドがない場所は、出力テーブルでは空白値になります。 **ヒント** これら 2 つのオプションの例を示した図と画面キャプチャについては、テーブルへの追加を参照してください。

名前	説明
	> **メモ** APPEND コマンドでは、演算フィールドの追加がサポートされていないためです。詳細については、"演算フィールドがサポートされていない" ページ 75を参照してください。 **フィールドが共通と見なされる条件とは** フィールドが共通と見なされるには、フィールドが次のようである必要があります。 ◦ すべてのソーステーブルに出現している ◦ 同じ物理名を持つ ◦ 同じデータ型(下記)に属している: • 文字 • 数値 • 日付時刻 • 論理 **同じ名前、異なるデータカテゴリ** 2 つのフィールドが同じ名前を持っているが、異なるデータカテゴリに属している場合には、エラーメッセージが表示され、APPEND コマンドは実行されません。 このエラーメッセージには、APPEND で指定された全テーブルにおけるデータカテゴリの不一致のすべてが表示されます。このメッセージはコマンドログに保存されます。 > **メモ** ASCHAR または ALLCHAR を使用してデータカテゴリを一致させることで、このような状況を回避できます。
OPEN 省略可能	コマンドを実行した後、コマンドによって作成されたテーブルを開きます。コマンドが出力テーブルを作成する場合にのみ有効です。
ASCHAR 省略可能	非文字フィールドを文字データカテゴリに変換することで、同じ名前で異なるデータカテゴリのフィールドを一致させます。 たとえば、Employee_ID フィールドは 1 つのテーブルの文字データであり、もう 1 つの表の数値データである 2 つのテーブルを追加します。数値的 Employee_ID フィールドは文字データに変換され、2 つのフィールドはエラーなしで追加されます。Employee_ID フィールドは文字データに変換され、2 つのフィールドはエラーなしで最後に追加されます。 ALLCHAR も指定されている場合、ASCHAR は無視されます。
ALLCHAR 省略可能	文字データカテゴリに最後に追加されるすべてのテーブルのすべての非文字フィールドを変換します。 この文字データへのグローバル変換により、すべて同じ名前のフィールドはエラーなしで最後に追加されます。 > **メモ** 最後に追加後は、フィールドに含まれているデータに応じて、最後に追加されたフィールド全体のデータカテゴリを変更することができます。

例

3 つの月次取引テーブルを最後に追加する

以下の例では、3 つの月次取引テーブルが追加され、それらのテーブル内のすべてのフィールドを追加した四半期取引テーブルが出力されます。

```
APPEND Trans_Jan, Trans_Feb, Trans_Mar TO Trans_Q1
```

3 つの従業員テーブルを最後に追加し、共通フィールドのみを含める

以下の例では、3 つの課員テーブルが追加され、それらのテーブル内の共通フィールドのみを追加したマスター従業員テーブルが出力されます。

```
APPEND Employees_central, Employees_east, Employees_west TO Employees_master
COMMONFIELDS
```

3 つの従業員テーブルを最後に追加し、異なるデータカテゴリのフィールドを一致させる

以下の例では、3 つの部門別従業員テーブルを最後に追加されており、そこでは、いくつかの同じ名前のフィールドは異なるデータカテゴリを使用しています。

最初の例では、一致のために必要な場合のみ、非文字フィールドを文字データカテゴリに変換しています:

```
APPEND Employees_central, Employees_east, Employees_west TO Employees_master ASCHAR
```

2 番目の例では、一致に必要かどうか関わらず、すべての非文字フィールドを文字データカテゴリに変換しています:

```
APPEND Employees_central, Employees_east, Employees_west TO Employees_master ALLCHAR
```

備考

> **メモ**
>
> このコマンドの動作の詳細については、Analytics のヘルプを参照してください。

機能の仕組み

APPEND コマンドは、2 つ以上のテーブル内のレコードを追加した新しいテーブルを作成することで、それらのテーブル内のレコードを結合します。ここで、追加とはレコードグループを別のレコードグループの末尾に付加することです。

同じ物理名と同じデータカテゴリを持つソーステーブルフィールドは、互いのフィールドに直接追加されます。

すべてのソーステーブルにおいて一意である物理名を持つフィールドは、出力テーブルには追加されますが、互いのフィールドには直接追加されません。

> **ヒント**
>
> 名前が異なるフィールドを直接追加したい場合は、事前に各テーブルレイアウト内のフィールドの物理名を統一しておいてください (フィールドが同じデータカテゴリに属しているか、ASCHAR または ALLCHAR を使用してフィールドのデータカテゴリを一致させていると仮定します。)

APPEND の使用に適する場面

APPEND は、構造が同じまたは類似である複数のテーブル内のデータを結合したい場合に使用できます。たとえば、月次テーブルまたは四半期テーブルを年次テーブルに結合する場合に使用することをお勧めします。

> **ヒント**
>
> APPEND コマンドを 1 回実行しただけで、APPEND オプションを指定した EXTRACT コマンドの複数回の実行結果が置き換えられます。

JOIN や DEFINE RELATION の代替としては使用できない

APPEND は JOIN および DEFINE RELATION コマンドの代替としては使用できません。その理由は、共通キーフィールド内の一致する、または一致しない値に基づいてレコードを包含または除外することができないためです。APPEND では、すべてのソーステーブル内の全レコードが出力テーブルに追加されます。

類似性がまったくないテーブルの追加

類似性がまったくないテーブル、つまり共通のフィールドがない 2 つ以上のテーブルを追加することができます。類似性がないテーブルを追加することが分析目的に資する場合もあります(これは APPEND コマンドの意図した主な用途ではありませんが) 。

日付時刻フィールドの追加

2 つ以上の日付時刻フィールドを追加するには、次の条件を満たす必要があります。

- 物理名が同じ
- データカテゴリが同じ(日付時刻)
- データサブタイプが同じ(日付、日付時刻、時刻)
- タイムゾーンインジケーターの使用有無の一致 - 追加対象となるすべてのフィールドで使用するか、使用しないかのいずれか

2 つの日付時刻フィールドが同じ名前を持っているが、その他の条件のいずれかを満たしていない場合には、エラーメッセージが表示され、APPEND コマンドは実行されません。

エラーメッセージには、APPEND で指定された全テーブルにおいて満たされなかったすべての条件が表示されます。このメッセージはコマンドログに保存されます。

> **メモ**
>
> 異なる日付時刻フィールドを文字データカテゴリに変換してすることでこれらのフィールドを一致させ、それから最後に追加ことができます。このアプローチでは、データを 1 つのテーブルに結合することができます。ただし、ソースデータの性質によっては、結合されたデータを日付時刻データに戻すことができない場合があります。

自動調整

次の場合、APPEND コマンドにより、フィールドを追加するために、フィールドが自動的に調整されます。

フィールドのデータカテゴリ	調整の実行
文字	◦ 異なるフィールド長は調整されます。 ◦ フィールドを ASCII または UNICODE データ型に変換することで、カスタム、PCASCII、EBCDIC など様々な文字データ型が調整されます。
数値	◦ 異なるフィールド長は調整されます。フィールドは ACL データ型に変換されます。 ◦ 異なる小数点以下桁数が定義されている場合は、調整されます。小数点以下桁数は、必要に応じて数値の末尾にゼロが付加されて、最大の桁数に統一されます。フィールドは ACL データ型に変換されます。 ◦ フィールドを ACL データ型に変換することで、Print、Float、EBCDIC、Micro など複数のデータ型が調整されます。
日付時刻	◦ フィールドを Analytics のデフォルト書式に変換することで、ソースデータ内にある複数の日付書式、日付時刻書式、時刻書式が調整されます。 • YYYYMMDD • YYYYMMDD hh:mm:ss • hh:mm:ss

自動調整が実行されない場合とは

次の場合、フィールドは Analytics によって自動調整されません。エラーメッセージが表示され、追加操作は実行されません。

- 名前が同じ 2 つのフィールドが異なるデータカテゴリに属する
- 名前が同じ 2 つの日付時刻フィールドが異なる日付時刻サブタイプ(日付、日付時刻、時刻) に属する
- 名前が同じ 2 つの日付時刻フィールドにおいて、タイムゾーンインジケーターの使用有無が一致していない

> **メモ**
>
> 同じ名前で異なるデータカテゴリのフィールドのユーザー指定の一致については上記で説明されています。詳細については、"ASCHAR" ページ 71と"ALLCHAR" ページ 71を参照してください。

演算フィールドがサポートされていない

APPENDコマンドでは、演算フィールドの追加がサポートされていないためです。テーブルを追加する場合、ソーステーブル内のすべての演算フィールドは自動的に出力テーブルから除外されます。

あるソーステーブル内の演算フィールドが、別のソーステーブル内の物理フィールドと同じ名前を持っている場合には、エラーメッセージが表示され、APPENDコマンドは実行されません。

> **ヒント**
>
> 演算フィールドを追加するには、まずこのようなフィールドを抽出して、物理フィールドに変換します(詳細については、"EXTRACTコマンド" ページ 193を参照してください)。次に、演算フィールドの抽出先となったテーブルを追加操作で使用します。
>
> 別のアプローチとしては、追加された出力テーブルに演算フィールドを再作成します。

レコード ノート フィールドがサポートされていない

APPENDコマンドでは、レコード ノート フィールドの追加がサポートされていないためです。テーブルを追加する場合、ソーステーブル内のすべてのレコード ノート フィールドは自動的に出力テーブルから除外されます。

あるソーステーブル内のレコード ノート フィールドが、別のソーステーブル内の物理フィールドと同じ名前を持っている場合には、エラーメッセージが表示され、APPENDコマンドは実行されません。

レコード ノート フィールドは、レコードにノートを追加したときにAnalyticsによって自動的に生成されるものです。

レコード長

追加時にすべてのソーステーブル内の全フィールドを含めるには、出力テーブルのレコード長がソーステーブル内の最も長いレコードより長くなる可能性があります。

出力テーブルのレコード長がAnalyticsの上限である32 KBを超えると、エラーメッセージが表示されます。

追加と小数点以下桁数

小数位がある数値フィールドの追加は、特定の動作によって制御されます。

小数位設定

APPENDコマンドでは、テーブル レイアウトのフィールド定義にある**小数位**設定に定義された小数点以下桁数が使用されます。

> **メモ**
>
> **小数位**設定は、ソースデータ内の実際の小数点以下桁数とは同じでない場合があります。**小数位**設定を超える小数点以下桁数は未定義のため、計算では丸められます。

統一されていない小数位設定

追加された数値フィールドの**小数位**設定が統一されていない場合には、それらのフィールドはACLデータ型に変換され、最も長い**小数位**設定に自動的に調整されます。

最も長い**小数位**設定を超えるソースデータファイル内の小数点以下桁数は、APPENDで生成される出力テーブルから**除外**されます。

統一されている小数位設定

追加された数値フィールドの**小数位**設定が統一されている場合には、データ型変換も調整も行われません。

小数位設定を超えるソースデータファイル内の小数点以下桁数は、APPENDで生成される出力テーブルに**追加**されます。

並べ替え

ソーステーブルに存在するすべての並べ替え順は、出力テーブル内のそれぞれのレコードセットで別々に保持されます。

たとえ両方のテーブルのレコードが並べ替えられていても、結果として生じる結合されたテーブルは、並べ替えられていないテーブルと見なされます。これは、抽出されたレコードを出力先テーブルの末尾にグループとして追加するときに、出力先テーブルの既存の並べ替え順が何も考慮されないからです。

たとえば、月次または四半期テーブルを追加して年次テーブルを作成する場合は、月次または四半期データの内部並べ替えは保持されます。必要であれば、追加操作を実行した後で、出力テーブルを並べ替えることもできます。

フィールドの順序の使用方法

共通フィールド

ソーステーブルの共通フィールドは、同じ順序になっていなくても追加できます。

たとえば、以下のフィールドは順序が異なっていても正常に追加されます。

テーブル	*フィールド*
テーブル1	ラストネーム | ファーストネーム | ミドルネーム
テーブル2	ファーストネーム | ミドルネーム | ラストネーム

出力テーブル内でのフィールドの順序は、APPENDコマンドに指定した最初のテーブルによって決まります。したがって、上の例では、出力テーブルの順序は次のようになります。

- ラストネーム | ファーストネーム | ミドルネーム

共通でないフィールド

ソーステーブル内の共通でないフィールドは、選択したテーブルグループに出現している順に出力テーブルに出力されます。

たとえば、以下の2つのテーブルを追加する場合を考えます。

テーブル	フィールド
テーブル1	役職 | ラストネーム | ファースト ネーム | ミドルネーム
テーブル2	ファーストネーム | ミドルネーム | ラストネーム | 生年月日

出力テーブルでの順序は次のようになります。

- 役職 | ラストネーム | ファーストネーム | ミドルネーム | 生年月日

代替列見出し

ソーステーブルの代替列見出しは、出力テーブルにも出力されます。複数のソーステーブルで同じフィールドに代替列見出しがある場合は、最初に選択したテーブルの代替列見出しが優先的に出力されます。

ASSIGN コマンド

変数を作成し、変数に値を割り当てます。

構文

```
ASSIGN 変数名 = 値 <IF テスト>
```

> **ヒント**
>
> Analytics は、次の構文を自動的に割り当て操作として解釈するので、ASSIGN キーワードは省略できます。
>
> ```
> 変数名 = 値
> ```

パラメーター

名前	説明
変数名	値を割り当てる変数の名前。変数自体が存在していない場合は、変数が作成されます。変数が既に存在する場合は、新しい値で更新されます。 変数の名前に、é のような英語以外の文字は使用しないでください。変数名に英語以外の文字が含まれていると、スクリプト エラーになります。 **メモ** 変数の名前は、31 文字までの英数字に制限されます。名前にはアンダースコア文字 (_) を使用できますが、他の特殊文字やスペースは使用できません。名前の先頭を数字にすることはできません。
値	変数に割り当てる値。新しい変数が作成される場合、変数の型は*値*のデータ型に基づいて決定されます。
IF テスト 省略可能	変数を作成したり変数に値を割り当てたりするには true となる必要のある条件式。

例

変数に値を割り当てる

現在のレコード内の Amount フィールドの値を変数 *v_current_amount* に割り当てるとします。*v_current_*

amount は変数であるため、その値は別の ASSIGN コマンドによって変更されない限り変わりません。

```
ASSIGN v_current_amount = Amount
```

変数に条件付きで値を割り当てる

変数 *v_counter* が 10 未満の場合にのみ、変数 *v_quantity* の値を 1 に更新する必要があるとします。

v_counter が 10 以上の場合は割り当てが行われず、*v_quantity* の値は変わりません。

オプションの ASSIGN キーワードは省略されている点に注目してください。

```
v_quantity = 1 IF v_counter < 10
```

備考

変数の期間

アンダースコアが先頭に付けられていない変数は現在の Analytics セッションの間のみ有効です。

変数を Analytics プロジェクトで永続的に保存する場合は、変数名の先頭にアンダースコア(_) を付けてください。

```
ASSIGN 値 = _変数名
```

演算フィールドまたは GROUP で使用される変数の再割り当て

以下の場合に既存の変数に値を割り当てると、その新しい値は割り当てられるものの、前の値の長さおよび小数点以下桁数は保持されます。

- 変数が演算フィールド内で使用されている場合
- GROUP 内で変数に値が再度割り当てられる場合

必要に応じて、新しい値の長さを伸ばしたり切り詰めたり、小数点以下桁数が調整されます。

上記以外の場合に変数に値を再度割り当てると、前の値、その長さおよび小数点以下桁数は上書きされます。

Analytics コマンドによって作成される変数

Analytics のダイアログボックスで情報を入力する、またはスクリプトを実行することで、特定のコマンドが実行されると、Analytics によってシステム変数が自動的に作成されます。これらの変数とそれに含まれる値は、後続の Analytics コマンドを処理する際に利用できます。

システム変数の値は、同じコマンドを再度実行すると、更新された値に置き換わります。

詳細については、"Analyticsコマンドによって作成される変数" ページ 924を参照してください。

BENFORD コマンド

フィールドに表示される最初の桁(1 ～ 9)や最初の桁の組み合わせの数をカウントし、実数と予測数を比較します。予測数はベンフォードの法則によって計算されます。

構文

BENFORD <ON> *数値型フィールド* <LEADING *n*> <IF *テスト*> <BOUNDS> <TO {SCREEN|*テーブル名*|GRAPH|PRINT}> <HEADER *ヘッダーテキスト*> <FOOTER *フッターテキスト*> <WHILE *テスト*> <FIRST *範囲*|NEXT *範囲*> <APPEND> <OPEN> <LOCAL>

パラメーター

名前	説明
ON *数値フィールド*	分析する数値フィールド。 **メモ** 取引金額などの「自然発生数」を含むフィールドを選択します。ベンフォード分析は、いかなる方法でも制約を受けた数値データには適していません。 詳細については以下を参照 "ベンフォード分析を使用してテストできるデータ" ページ 83
LEADING *n* 省略可能	分析する先頭桁数。*n* の値は 1 から 6 の範囲で指定します。 LEADING を省略した場合は、デフォルト値の 1 が使用されます。
IF *テスト* 省略可能	各レコードを処理するために真である必要がある条件式。コマンドは、その条件を満たすレコードに対してのみ実行されます。 **メモ** IF パラメーターは、任意の範囲パラメーター(WHILE、FIRST、NEXT) が適用された後に、テーブルに残るレコードに対してのみ評価されます。
BOUNDS 省略可能	出力結果に計算された上限および下限値を含めます。 出力結果の複数の桁または桁の組み合わせの実際のカウントが境界のいずれかを超える場合、データは操作された可能性があり、調査が必要になります。
TO SCREEN | *テーブル名* | GRAPH | PRINT 省略可能	コマンドの結果を送信する場所: ◦ **SCREEN** - は Analytics の表示領域に結果を表示します ◦ ***テーブル名*** -は、結果の保存先となる Analytics テーブルのことです。 *テーブル名*には、ファイル拡張子 .FIL を付けた文字列を引用符で囲んで指定する必要があります。例: TO "Output.FIL" デフォルトでは、テーブル データ ファイル(.fil) は、Analytics プロジェクトが入っているフォルダー

名前	説明
	に保存されます。 既存の異なるフォルダーにデータ ファイルを保存するには、絶対または相対ファイル パスを指定します。 • TO "C:\Output.FIL" • TO "Results\Output.FIL" **メモ** テーブル名は 64 文字の英数字(.FIL 拡張子を含まない) に制限されています。名前にはアンダースコア文字(_) を使用できますが、他の特殊文字やスペースは使用できません。名前の先頭を数字にすることはできません。 ◦ **GRAPH** - は結果をグラフに表示し、それを Analytics の表示領域に表示します ◦ **印刷** -- 通常使うプリンターに結果を送信します
HEADER *ヘッダーテキスト* 省略可能	レポートの各ページの最上部に挿入されるテキスト。 *ヘッダーテキスト*は引用符で囲んだ文字列として指定する必要があります。この値は、Analytics のシステム変数である HEADER の値よりも優先されます。
FOOTER *フッターテキスト* 省略可能	レポートの各ページの最下部に挿入されるテキスト。 *フッターテキスト*は引用符で囲んだ文字列として指定する必要があります。この値は、Analytics のシステム変数である FOOTER の値よりも優先されます。
WHILE *テスト* 省略可能	各レコードを処理するために真である必要がある条件式。条件が false と評価するか、テーブルの最後に達したら、コマンドは実行を中止します。 **メモ** WHILE を FIRST または NEXT とともに使用する場合は、1 つの制限に達するとすぐに、レコードの処理が停止します。
FIRST *範囲* | NEXT *範囲* 省略可能	処理するレコード数: ◦ **FIRST** - 指定されたレコード数に達するまで、最初のレコードから処理を開始します ◦ **NEXT** - 指定されたレコード数に達するまで、現在選択されているレコードから処理を開始します *範囲*は処理するレコード数を指定します。 FIRST と NEXT を省略すると、すべてのレコードがデフォルトで処理されます。
APPEND 省略可能	コマンドの出力を既存ファイルに上書きしないで、そのファイルの末尾に追加します。 **メモ** コマンドの出力と既存のファイルの構造が同一であるようにする必要があります。 • 同じフィールド • 同じフィールド順序 • 一致するフィールドが同じ長さ • 一致するフィールドが同じデータ型 出力は、既存ファイルとの間でファイル構造が違っている場合でも、Analytics によって既存ファイルに追加されます。出力と既存のファイルの構造が一致しない場合は、データが混在、不足、不正確になります。
OPEN	コマンドを実行した後、コマンドによって作成されたテーブルを開きます。コマンドが出力テーブル

名前	説明
省略可能	を作成する場合にのみ有効です。
LOCAL 省略可能	Analytics プロジェクトと同じ場所に出力ファイルを保存します。 **メモ** Analytics テーブルである出力ファイルを含むサーバー テーブルに対してコマンドを実行するときにのみ適用されます。

例

結果をグラフに出力する

Amount フィールドに対して BENFORD コマンドを実行し、その結果をグラフに出力します。

```
BENFORD ON Amount LEADING 2 BOUNDS TO GRAPH
```

備考

ベンフォード分析を使用してテストできるデータ

ベンフォード分析は、会計金額、取引金額、費用、またはアドレス番号などの「自然発生数」から構成される数値データをテストする目的でのみ、使用してください。ベンフォード分析は、いかなる方法でも制約を受けた数値データには適していません。

ベンフォード分析に適した数値データを特定するには、次のガイドラインに従ってください。

- **データセットのサイズ** - 有効は分布をサポートするには、データセットが十分に大きいサイズである必要があります。ベンフォード分析では、499 レコード以下の場合には、信頼できる結果が得られない場合があります。
- **先頭の数の要件** -1 ～ 9 のすべての数値が、先頭の数として発生する可能性がなければなりません。
- **先頭の数の組み合わせの要件** -0 ～ 9 のすべての数値が、先頭から 2 番目の数および分析対象の追加の数として発生する可能性がなければなりません。
- **制約されたデータ** - あらかじめ定義されたパターンに従って割り当てまたは生成された数値データは、ベンフォード分析に適していません。たとえば、次の分析では、ベンフォード分析を使用しないでください。
  - 連番の小切手または請求書番号
  - 特定のパターンにマッピングされる社会保障番号または電話番号
  - 特定の数字が出現しない範囲がある番号体系
- **乱数** -乱数生成器で生成された数値は、ベンフォード分析に適していません。

CALCULATE コマンド

1 つ以上の式の値を計算します。

構文

```
CALCULATE {式 <AS 結果ラベル> <,...n>
```

パラメーター

名前	説明
式	計算する式。 式は次の 4 種類のいずれかにすることができます。 ◦ 文字 ◦ 数値 ◦ 日付時刻 ◦ 論理 式が複数ある場合は、カンマで区切ります。 `CALCULATE 4.7 * 18.5, 1 + 2, "a" + "b"`
AS *結果ラベル* 省略可能	画面とAnalytics コマンド ログに表示する結果の名前。 *結果ラベル* パラメーターには、引用符で囲まれた文字列か有効な文字式を指定する必要があります。 このパラメーターを指定しない場合は、計算を行う式が結果名として使用されます。

例

単純な計算を行う

CALCULATE を使用して、4.70 を 18.50 で乗算し、結果の 86.95 を返します。

```
CALCULATE 4.70 * 18.50
```

計算結果に名前を付ける

定義済みの販売価格と単価のフィールドを使って、現在選択されているレコードの粗利益を導出するには、

CALCULATE を使用します。

```
CALCULATE 販売価格 - 単価 AS "Margin"
```

結果は画面とログで、"Margin" と表示されます。

備考

機能の仕組み

CALCULATE には、Analytics の関数、変数、および現在選択されているレコード内データにアクセスしながら計算する機能があります。

コマンドの出力

CALCULATE をどこで実行するかによって、異なる場所に結果が出力されます。

- **コマンド ラインでコマンドを入力した場合：**-結果は画面に表示されます。
- **スクリプト内でコマンドを使用した場合：**-結果はログに記録されます。

結果 ラベル パラメーターの値は変数ではないのでスクリプト内で使用できません。これは単に画面またはログで計算を識別するために使用されます。

出力における小数点以下桁数

数値計算の結果の小数位は、式の中で小数位が最も多い式要素と同じになります。

1 が返されます。

```
CALCULATE 365/52/7
```

1.0027 が返されます。

```
CALCULATE 365.0000/52/7
```

テーブル入力の操作

式にフィールド値が含まれる場合は、そのフィールドが属するテーブルを開いておく必要があります。
CALCULATE コマンドで分析するレコードに移動するためには、FIND、SEEK、または LOCATE コマンドを使用することができます。

CLASSIFY コマンド

文字または数値フィールドの等しい値に基づいてレコードをグループ化する各グループのレコード数をカウントし、指定した数値フィールドの小計をグループごとに求めます。

構文

CLASSIFY <ON> *キーフィールド* <SUBTOTAL *数値フィールド <...n>*|ALL> <INTERVALS *数値*> <SUPPRESS> <TO SCREEN|*テーブル名*|GRAPH|PRINT> <IF *テスト*> <WHILE *テスト*> <FIRST *範囲*|NEXT *範囲*> <HEADER *ヘッダーテキスト*> <FOOTER *フッターテキスト*> <KEY *内訳フィールド*> <OPEN> <APPEND> <LOCAL> <STATISTICS>

パラメーター

名前	説明
ON *キーフィールド*	分類する文字または数値フィールド。 キーフィールドの最大長は 2048 文字です。 2048 文字より長いキーフィールドを使用してテーブルを分類化したい場合は、SUMMARIZE コマンドを使用します。SUMMARIZE コマンドにはキーフィールドの長さ制限はありません。
SUBTOTAL *数値フィールド <...n>* | SUBTOTAL ALL 省略可能	グループごとに小計を計算する 1 つ以上の数値フィールドまたは式。 複数のフィールドはスペースで区切る必要があります。テーブル内のすべての数値フィールドについて小計を求める場合は ALL を指定します。
INTERVALS *数* 省略可能	出力結果のグループの最大数。 分類化されるフィールドにある同一値セットの数が指定された最大数を超える場合は、列の先頭から数えた同一値セット数のみの同一値セットが使用されます。 最大数を超える同一値セットは、"OTHER"(その他) というグループにまとめられます。 INTERVALS を省略した場合は、分類化されるフィールドの同一値セットごとに 1 つのグループが作成されます。 **メモ** このパラメーターは Analytics のユーザーインターフェイスには実装されていません。ACLScript 構文の一部としてのみ使用することができます。
SUPPRESS 省略可能	**メモ** INTERVALS も指定されていない場合は使用できません。SUPPRESS は Analytics のユーザーインターフェイスには実装されていません。スクリプトまたはコマンドラインで、ACLScript 構文の一部としてのみ使用することができます。

名前	説明
	INTERVALS で指定された最大数を超える同一値セットは、コマンド出力から除外します。
TO SCREEN | *テーブル名* | GRAPH | PRINT	コマンドの結果を送信する場所： ◦ **SCREEN** - は Analytics の表示領域に結果を表示します ◦ ***テーブル名*** -は、結果の保存先となる Analytics テーブルのことです。 *テーブル名*には、ファイル拡張子 .FIL を付けた文字列を引用符で囲んで指定する必要があります。例：TO "Output.FIL" デフォルトでは、テーブル データ ファイル(.fil) は、Analytics プロジェクトが入っているフォルダーに保存されます。 既存の異なるフォルダーにデータ ファイルを保存するには、絶対または相対ファイル パスを指定します。 • TO "C:\Output.FIL" • TO "Results\Output.FIL" **メモ** テーブル名は 64 文字の英数字(.FIL 拡張子を含まない) に制限されています。名前にはアンダースコア文字(_) を使用できますが、他の特殊文字やスペースは使用できません。名前の先頭を数字にすることはできません。 ◦ **GRAPH** - は結果をグラフに表示し、それを Analytics の表示領域に表示します ◦ **印刷** -- 通常使うプリンターに結果を送信します
IF *テスト* 省略可能	各レコードを処理するために真である必要がある条件式。コマンドは、その条件を満たすレコードに対してのみ実行されます。 **メモ** IF パラメーターは、任意の範囲パラメーター(WHILE、FIRST、NEXT) が適用された後に、テーブルに残るレコードに対してのみ評価されます。
WHILE *テスト* 省略可能	各レコードを処理するために真である必要がある条件式。条件が false と評価するか、テーブルの最後に達したら、コマンドは実行を中止します。 **メモ** WHILE を FIRST または NEXT とともに使用する場合は、1 つの制限に達するとすぐに、レコードの処理が停止します。
FIRST *範囲* | NEXT *範囲* 省略可能	処理するレコード数： ◦ **FIRST** - 指定されたレコード数に達するまで、最初のレコードから処理を開始します ◦ **NEXT** - 指定されたレコード数に達するまで、現在選択されているレコードから処理を開始します *範囲*は処理するレコード数を指定します。 FIRST と NEXT を省略すると、すべてのレコードがデフォルトで処理されます。
HEADER *ヘッダーテキスト* 省略可能	レポートの各ページの最上部に挿入されるテキスト。 ヘッダーテキストは引用符で囲んだ文字列として指定する必要があります。この値は、Analytics のシステム変数である HEADER の値よりも優先されます。
FOOTER *フッターテキスト*	レポートの各ページの最下部に挿入されるテキスト。

名前	説明
省略可能	*フッターテキスト*は引用符で囲んだ文字列として指定する必要があります。この値は、Analytics のシステム変数である FOOTER の値よりも優先されます。
KEY *ブレークフィールド* 省略可能	小計計算をグループ化するフィールドまたは式。*ブレークフィールド*の値が変わるたびに、小計が計算されます。 *ブレークフィールド*は、文字フィールドか式である必要があります。指定できるフィールドは 1 つだけですが、1 つ以上のフィールドを含んでいる式を使用することができます。
OPEN 省略可能	コマンドを実行した後、コマンドによって作成されたテーブルを開きます。コマンドが出力テーブルを作成する場合にのみ有効です。
APPEND 省略可能	コマンドの出力を既存ファイルに上書きしないで、そのファイルの末尾に追加します。 **メモ** コマンドの出力と既存のファイルの構造が同一であるようにする必要があります。 • 同じフィールド • 同じフィールド順序 • 一致するフィールドが同じ長さ • 一致するフィールドが同じデータ型 出力は、既存ファイルとの間でファイル構造が違っている場合でも、Analytics によって既存ファイルに追加されます。出力と既存のファイルの構造が一致しない場合は、データが混在、不足、不正確になります。
LOCAL 省略可能	Analytics プロジェクトと同じ場所に出力ファイルを保存します。 **メモ** Analytics テーブルである出力ファイルを含むサーバーテーブルに対してコマンドを実行するときにのみ適用されます。
STATISTICS 省略可能	**メモ** SUBTOTAL も指定されていない場合は使用できません。 すべての SUBTOTAL フィールドの平均値、最小値、および最大値を計算します。

例

顧客ごとの総取引額

Customer_Number(顧客番号)フィールドに基づいて Ar(売掛金)テーブルを分類化し、**Trans_Amount**(取引額)フィールドの小計を求めたいとします。出力結果は、顧客ごとでグループ化され、各顧客の総取引額が含まれます。

```
OPEN Ar
CLASSIFY ON Customer_Number SUBTOTAL Trans_Amount TO "Customer_total.FIL"
```

顧客ごとの取引額の合計、平均、最小、および最大

前の例と同様に、**Customer_Number**(顧客番号) フィールドに基づいてAr(売掛金) テーブルを分類化し、**Trans_Amount**(取引額) フィールドの小計を求めます。

今回は、各顧客の取引額の平均値、最小値、および最大値を計算するためにSTATISTICS を追加します。

```
OPEN Ar
CLASSIFY ON Customer_Number SUBTOTAL Trans_Amount TO "Customer_stats.FIL"
STATISTICS
```

同じ請求金額

AP_Trans(買掛取引) テーブルに2 回以上出現する請求金額を特定したいとします。

それには、テーブルを **Invoice_Amount**(請求金額) フィールドで分類化します。出力結果は請求金額によってグループ化され、2 回以上出現する請求金額を特定するために使用できる関連カウントとともに表示されます。

```
OPEN AP_Trans
CLASSIFY ON Invoice_Amount TO "Grouped_invoice_amounts.FIL" OPEN
SET FILTER TO COUNT > 1
```

備考

> **メモ**
>
> このコマンドの動作の詳細については、Analytics のヘルプを参照してください。

機能の仕組み

CLASSIFY は、文字または数値フィールドに同じ値を持つレコードをグループ化します。

出力にはグループごとに、ソーステーブルのうち、そのグループに属するレコードの数が記録された特別なレコードが含まれます。

並べ替えとCLASSIFY

CLASSIFY では、データは並べ替えられていてもいなくても処理できます。出力は、自動的に昇順に並べ替えられます。

自動生成された小計と統計フィールドの名前

STATISTICS を使用して、1 つ以上の SUBTOTAL フィールドで統計演算を実行し、結果を Analytics テーブルに出力する場合は、パラメーターによって自動生成されたフィールドの名前は次のようになります。

自動生成されたフィールドの説明	出力テーブルのフィールド名	出力テーブルの列見出し(表示名)
小計フィールド	*集計対象となる、ソース テーブルのフィールド名*	**Total** + *集計対象となる、ソース テーブルの代替列見出し*
平均フィールド	**a_***集計対象となる、ソース テーブルのフィールド名*	**Average** + *集計対象となる、ソース テーブルの代替列見出し*
最小フィールド	**m_***集計対象となる、ソース テーブルのフィールド名*	**Minimum** + *集計対象となる、ソース テーブルの代替列見出し*
最大フィールド	**x_***集計対象となる、ソース テーブルのフィールド名*	**Maximum** + *集計対象となる、ソース テーブルの代替列見出し*

CLOSE コマンド

Analytics のテーブル、インデックス ファイル、またはログ ファイルを閉じ、**スクリプト記録**セッションを終了します。

構文

```
CLOSE <テーブル名|PRIMARY|SECONDARY|INDEX|LOG|LEARN>
```

パラメーター

名前	説明
テーブル名 |PRIMARY|SECONDARY|INDEX|LOG|LEARN 省略可能	閉じる対象となる項目は次のとおりです。 ◦ ***テーブル名*** -: 閉じる対象となる Analytics テーブルの名前 ◦ **PRIMARY** - を指定すると、Analytics の主テーブルが閉じられます。 CLOSE をパラメーターなしで使用すると主テーブルを閉じます。 ◦ **SECONDARY** - では、Analytics の副テーブルが閉じられます。 ◦ **INDEX** - では、Analytics テーブルに適用されている現在のインデックスが閉じられます。 ◦ **LOG** - では、SET LOG コマンドを使用してデフォルト以外のログ ファイルが指定されている場合に、ログ ファイルをデフォルトのコマンド ログに再設定します。 ◦ **LEARN** - では、アクティブな**スクリプト レコーダー** セッションを終了し、セッションが記録されたスクリプト ファイルを保存するかどうかを自身で確認するようにします。 LEARN はスクリプト内で使用することができますが、その目的とする用途はコマンド ラインにあります。**スクリプト レコーダー**は、Analytics のユーザー インターフェイスでダイアログボックスを使用して実行されるコマンドの ACLScript 構文を記録します。

例

名前によってテーブルを閉じる

Inventory という名前のテーブルを閉じるには、次のようにします。

```
CLOSE Inventory
```

種類によってテーブルを閉じる

現在の副テーブルを閉じるには、次のようにします。

```
CLOSE SECONDARY
```

デフォルトの Analytics コマンド ログに戻す

スクリプトのデータ検証フェーズを別のログファイルに取り込んだ後で、ログファイルをデフォルトのコマンド ログに再設定するには、次のようにします。

```
SET LOG TO "DataVerificationPhase.log"
COMMENT データ検証コマンドの実行
CLOSE LOG
```

備考

CLOSE を使用しない場合とは

Analytics テーブルは通常、閉じる必要がありません。アクティブな Analytics テーブルは、ユーザーが別のテーブルを開けば自動的に閉じられるからです。主テーブルも、OPEN コマンドや QUIT コマンドを実行する前に自動的に閉じられます。

CLOSE で Analytics プロジェクトを閉じることはできません。代わりに QUIT を使用します。

関連するフィールドおよびテーブル

主テーブルや副テーブルを閉じると、関連するすべてのフィールド定義がメモリから解放されます。テーブルが閉じられる前に、テーブル レイアウトへの変更が保存されます。

テーブルの関連付けが定義された Analytics プロジェクトで CLOSE コマンドを実行すると、主テーブルと副テーブルがともに閉じられます。また、関連するテーブルも閉じられます。

CLUSTER コマンド

1 つ以上の数値フィールドの類似した値に基づいて、レコードをクラスターにグループ化します。クラスターは単次元または多次元です。

構文

CLUSTER ON *キーフィールド* <...*n*> KVALUE *クラスター数* ITERATIONS *繰り返し数* INITIALIZATIONS *初期化数* <SEED *シード値*> <OTHER *フィールド* < ...*n*>> TO *テーブル名* <IF *テスト*> <WHILE *テスト*> <FIRST *範囲*|NEXT *範囲*> OPEN {*キーワードなし*|NOCENTER|NOSCALE}

パラメーター

名前	説明
ON *キーフィールド* <...*n*>	クラスター化する 1 つ以上の数値フィールド。複数のフィールドはスペースで区切る必要があります。
KVALUE *クラスター数*	出力結果で生成されるクラスター数。
ITERATIONS *繰り返し数*	クラスター計算が再実行される最大回数。
INITIALIZATIONS *初期化数*	ランダム中心の初期セットを生成する回数。
SEED *シード値* 省略可能	Analytics の乱数ジェネレーターを初期化するために使用するシード値。 SEED を省略した場合は、シード値がランダムに選択されます。
OTHER *フィールド* <...*n*> 省略可能	出力に含める 1 つ以上の追加フィールド。 **メモ** キーフィールドは、自動的に出力テーブルに追加されるため、OTHER
TO *テーブル名*	コマンドの結果を送信する場所： ◦ ***テーブル名*** -は、結果の保存先となる Analytics テーブルのことです。 *テーブル名*には、ファイル拡張子 .FIL を付けた文字列を引用符で囲んで指定する必要があります。例：TO "Output.FIL" デフォルトでは、テーブル データ ファイル(.fil) は、Analytics プロジェクトが入っているフォルダーに保存されます。 既存の異なるフォルダーにデータ ファイルを保存するには、絶対または相対ファイル パスを指定します。 • TO "C:\Output.FIL"

名前	説明
	• TO "Results\Output.FIL" **メモ** テーブル名は 64 文字の英数字(.FIL 拡張子を含まない) に制限されています。名前にはアンダースコア文字(_) を使用できますが、他の特殊文字やスペースは使用できません。名前の先頭を数字にすることはできません。
IF *テスト* 省略可能	各レコードを処理するために真である必要がある条件式。コマンドは、その条件を満たすレコードに対してのみ実行されます。 **メモ** IF パラメーターは、任意の範囲パラメーター(WHILE、FIRST、NEXT) が適用された後に、テーブルに残るレコードに対してのみ評価されます。
WHILE *テスト* 省略可能	各レコードを処理するために真である必要がある条件式。条件が false と評価するか、テーブルの最後に達したら、コマンドは実行を中止します。 **メモ** WHILE を FIRST または NEXT とともに使用する場合は、1 つの制限に達するとすぐに、レコードの処理が停止します。
FIRST *範囲* | NEXT *範囲* 省略可能	処理するレコード数: ◦ **FIRST** - 指定されたレコード数に達するまで、最初のレコードから処理を開始します ◦ **NEXT** - 指定されたレコード数に達するまで、現在選択されているレコードから処理を開始します *範囲*は処理するレコード数を指定します。 FIRST とNEXT を省略すると、すべてのレコードがデフォルトで処理されます。
OPEN 省略可能	コマンドを実行した後、コマンドによって作成されたテーブルを開きます。コマンドが出力テーブルを作成する場合にのみ有効です。
キーワードなし | NOCENTER | NOSCALE	キーフィールド数値を標準化する方法。 ◦ ***キーフィールドなし*** - ゼロ(0) 周辺のキーフィールド値を中央化し、クラスターを計算するときに値を単位分散に調整します ◦ **NOCENTER** - クラスターを計算するときに値を単位分散に調整しますが、ゼロ(0) 周辺の値を中央化しません ◦ **NOSCALE** - クラスターを計算するときに、未調整の未加工キーフィールド値を使用します

例

請求金額でのクラスター

Invoice_Amount フィールドで売掛金テーブルを階層化するほかに、同じフィールドでクラスター化することも決定します。

- 階層化により、$1000 間隔などのあらかじめ定義された数値の境界を使用して、金額が階層にグループ化されます。
- クラスターにより、前もって数値の境界を決定せずに、データに存在する金額の有機グループを検出します。

```
Open Ar
CLUSTER ON Invoice_Amount KVALUE 8 ITERATIONS 30 INITIALIZATIONS 10 OTHER No Due
Date Ref Type TO "Clustered_invoices" NOSCALE
```

各出力クラスターに含まれるレコード数を簡単に検出する方法として、Clustered_invoices 出力テーブルを **Cluster** フィールドで分類します。

```
OPEN Clustered_invoices
CLASSIFY ON Cluster TO SCREEN
```

備考

> **メモ**
>
> このコマンドの動作の詳細については、Analytics のヘルプを参照してください。

COMMENT コマンド

処理に影響を与えずに、スクリプトに注釈を追加します。

構文

単一行コメント

```
COMMENT コメントテキスト
```

複数行コメント

```
COMMENT
 コメントテキスト
  <...n>
<END>
```

> **メモ**
>
> コメント テキストの行の前には、^ キャラット文字を使用しないでください。キャラットは .acl プロジェクト ファイルにおいて特殊な使用方法があり、コメントの前にキャラットを置くと、個別の行のコメント テキストは保存されません。

パラメーター

名前	説明
コメントテキスト	追加するコメント。 ◦ **単一行のコメント** -は、改行なしでコメント テキスト全体を入力します。 ◦ **複数行のコメント** -は、COMMENT コマンドの直後の行で始まるコメント テキストを任意の数だけ入力します 個別の行で END キーワードを使用するか、改行を入れると、複数行コメントが終了します。
END 省略可能	複数コメント行を伴う COMMENT コマンドの終了。 END を使用する場合は、これを最後のコメント行の直後の行に入力する必要があります。 END を使用しない場合、最後のコメント行の直後には空白行を挿入する必要があります。

例

単一行コメント

単一行コメントは、スクリプトを将来保守するユーザーへの説明を追加するため、各コマンドの前に使用します。

```
COMMENT 標準偏差と平均を生成します。
STATISTICS ON %v_amt% STD TO SCREEN NUMBER 5
COMMENT 標準偏差と平均を格納するためにフィールドを作成します。
DEFINE FIELD Standard_Dev COMPUTED STDDEV1
DEFINE FIELD Average COMPUTED AVERAGE1
```

複数行コメント

作成する各スクリプトの先頭に、そのスクリプトの目的を説明する複数行コメントを記述することができます。

```
COMMENT
 このアナリティクスは、取引日値が等しいか
 1日違う共通の取引発生元ID(業者IDや
 Merchant ID)を持つ複数のレコードを特定します。
 このアナリティクスによって、分割請求書、分割発注書、分割購買依頼書、
 あるいは分割法人カード取引ができます。
END
```

備考

COMMENT の用途

COMMENT の用途は、スクリプトの目的や使用するロジックに関する情報、またスクリプトで必要な入力、定義した各変数の目的などの情報を含めることです。

スクリプトが実行されるたびに、スクリプト中のコメントも Analytics コマンド ログに書き込まれます。

COUNT コマンド

現在のビュー内のレコードの合計数や、指定した条件を満たすレコード件数をカウントします。

構文

```
COUNT <IF テスト> <WHILE テスト> <FIRST 範囲|NEXT 範囲>
```

パラメーター

名前	説明
IF *テスト* 省略可能	各レコードを処理するために真である必要がある条件式。コマンドは、その条件を満たすレコードに対してのみ実行されます。 **メモ** IF パラメーターは、任意の範囲パラメーター(WHILE、FIRST、NEXT) が適用された後に、テーブルに残るレコードに対してのみ評価されます。
WHILE *テスト* 省略可能	各レコードを処理するために真である必要がある条件式。条件が false と評価するか、テーブルの最後に達したら、コマンドは実行を中止します。 **メモ** WHILE を FIRST または NEXT とともに使用する場合は、1 つの制限に達するとすぐに、レコードの処理が停止します。
FIRST *範囲* | NEXT *範囲* 省略可能	処理するレコード数: ◦ **FIRST** - 指定されたレコード数に達するまで、最初のレコードから処理を開始します ◦ **NEXT** - 指定されたレコード数に達するまで、現在選択されているレコードから処理を開始します *範囲*は処理するレコード数を指定します。 FIRST と NEXT を省略すると、すべてのレコードがデフォルトで処理されます。

Analytics の出力変数

名前	含む
COUNT*n*	コマンドによって計算されたレコード数。 ◦ 変数名が COUNT1 の場合、それは最近実行されたコマンドのレコード数を格納しています。 ◦ 変数名 COUNT*n* の *n* が 1 より大きい場合、変数は GROUP コマンドの内で実行されたコ

名前	含む
	マンドのレコード数を格納しています。 *n* の値は GROUP 内のコマンドの行番号に基づいて割り当てられます。たとえば、コマンドが GROUP コマンドの 1 行下にある場合、値は COUNT2 が割り当てられます。コマンドが GROUP コマンドの 4 行下にある場合、値は COUNT5 が割り当てられます。

例

COUNT1 を保管する

COUNT コマンドの結果は COUNT1 出力変数に保管されます。この変数の値を取得してユーザー定義変数に保管することができます。

COUNT コマンドが実行されるたびに、COUNT1 変数が上書きされます。したがって、この変数の値は、テーブルにフィルターを適用した後でこのコマンドが再度実行される前に、ユーザー定義変数に格納しておく必要があります。

```
OPEN CustomerAddress
COUNT
TotalRec = COUNT1
SET FILTER TO ModifiedDate > '20100101'
COUNT
TotalFilteredRec = COUNT1
```

備考

COUNT の用途

COUNT コマンドの用途は、Analytics テーブル内のレコード数をカウントしたり、特定のテスト条件を満たすレコードのみをカウントしたりすることです。テスト条件を指定しないと、Analytics テーブル内のレコードの合計が表示されます。

COUNT へのフィルターの影響

ビューにフィルターが適用されている場合は、フィルターリング条件の適用後、ビュー内に残っているレコード数がカウントされます。

CREATE LAYOUT コマンド

特定のスクリプト作成状況で必要になる場合がある、空の Analytics テーブル レイアウトを作成します。

構文

```
CREATE LAYOUT レイアウト名 WIDTH 文字 <RECORD 0|RECORD 1>
```

パラメーター

名前	説明
レイアウト名	レイアウトの名前。
WIDTH *文字数*	レコード長の文字単位。
RECORD 0 | RECORD 1 省略可能	◦ RECORD 0 を指定するか、またはこのパラメーターを省略した場合は、レコードもソース データ ファイルもないテーブル レイアウトが作成されます。 ◦ RECORD 1 を指定した場合は、1 件の空レコードと *レイアウト名*.fil というソース データ ファイルを持つテーブル レイアウトが作成されます。

例

レコードのない空のテーブル レイアウトを作成する

レコード長が 100 文字である空のテーブル レイアウトを作成するには、次のようにします。

```
CREATE LAYOUT empty_table WIDTH 100
```

1 件のレコードを含む空のテーブル レイアウトを作成する

次を作成します。

- 1 件の空のレコードを含む空のテーブル レイアウト
- 50 文字のレコード長
- 対応する Analytics データ ファイルの名前は `empty_table.fil`

```
CREATE LAYOUT empty_table WIDTH 50 RECORD 1
```

備考

作成されるこの空のテーブルレイアウトには、**Field_1** という文字フィールドがあります。このフィールドの長さは、WIDTH で指定したレコード長と同じです。

> **メモ**
>
> このコマンドは、AX Server 上の Analytics でのアナリティクス実行では使用できません。

CROSSTAB コマンド

2 つ以上の文字または数値フィールドの値の等しい組み合わせに基づいてレコードをグループ化し、結果のグループを行と列のグリッドに表示します。各グループのレコード数をカウントし、指定した数値フィールドの小計をグループごとに求めます。

構文

CROSSTAB <ON> *行フィールド* <...*n*> COLUMNS *列フィールド* <SUBTOTAL {*数値フィールド* <...*n*>|SUBTOTAL ALL> TO {SCREEN|*テーブル名*|*ファイル名*|GRAPH|PRINT} <IF *テスト*> <WHILE *テスト*> <FIRST *範囲*|NEXT *範囲*> <APPEND> <COUNT> <OPEN> <LOCAL> <HEADER *ヘッダーテキスト*> <FOOTER *フッターテキスト*>

パラメーター

名前	説明
ON *行フィールド* <...*n*>	結果として生じる行列グリッドの行に使用する、フィールドまたは式。行の基準として、1 つ以上のフィールドや式を指定できます。
COLUMNS *列フィールド*	結果として生じる行列グリッドの列に使用する、フィールドまたは式。列に使用するフィールドや式は、1 つしか指定できません。
SUBTOTAL *数値フィールド* <...*n*> | SUBTOTAL ALL 省略可能	グループごとに小計を計算する 1 つ以上の数値フィールドまたは式。 複数のフィールドはスペースで区切る必要があります。テーブル内のすべての数値フィールドについて小計を求める場合は ALL を指定します。
TO SCREEN | *テーブル名* | *ファイル名* | GRAPH | PRINT	コマンドの結果を送信する場所： ◦ **SCREEN** - は Analytics の表示領域に結果を表示します ◦ ***テーブル名*** -は、結果の保存先となる Analytics テーブルのことです。 *テーブル名*には、ファイル拡張子 .FIL を付けた文字列を引用符で囲んで指定する必要があります。例：TO "Output.FIL" デフォルトでは、テーブル データ ファイル(.fil) は、Analytics プロジェクトが入っているフォルダーに保存されます。 既存の異なるフォルダーにデータ ファイルを保存するには、絶対または相対ファイル パスを指定します。 • TO "C:\Output.FIL" • TO "Results\Output.FIL" **メモ** テーブル名は 64 文字の英数字(.FIL 拡張子を含まない) に制限されています。名前にはアンダースコア文字(_) を使用できますが、他の特殊文字やスペースは使用できません。名前の先頭を数字にすることはできません。

名前	説明
	◦ ***ファイル名*** -は結果の保存先となるファイルです。 *ファイル名*には、適切なファイル拡張子を付けた文字列を引用符で囲んで指定します。例：TO "Output.TXT" デフォルトでは、テーブル ファイルは、Analytics プロジェクトが入っているフォルダーに保存されます。 既存の異なるフォルダーにファイルを保存するには、絶対または相対ファイル パスを指定します。 • TO "C:\Output.TXT" • TO "Results\Output.TXT" ◦ **GRAPH** - は結果をグラフに表示し、それを Analytics の表示領域に表示します ◦ **印刷** -- 通常使うプリンターに結果を送信します
IF *テスト* 省略可能	各レコードを処理するために真である必要がある条件式。コマンドは、その条件を満たすレコードに対してのみ実行されます。 **メモ** IF パラメーターは、任意の範囲パラメーター(WHILE、FIRST、NEXT) が適用された後に、テーブルに残るレコードに対してのみ評価されます。
WHILE *テスト* 省略可能	各レコードを処理するために真である必要がある条件式。条件が false と評価するか、テーブルの最後に達したら、コマンドは実行を中止します。 **メモ** WHILE を FIRST または NEXT とともに使用する場合は、1 つの制限に達するとすぐに、レコードの処理が停止します。
FIRST *範囲* | NEXT *範囲* 省略可能	処理するレコード数： ◦ **FIRST** - 指定されたレコード数に達するまで、最初のレコードから処理を開始します ◦ **NEXT** - 指定されたレコード数に達するまで、現在選択されているレコードから処理を開始します *範囲*は処理するレコード数を指定します。 FIRST とNEXT を省略すると、すべてのレコードがデフォルトで処理されます。
APPEND 省略可能	コマンドの出力を既存ファイルに上書きしないで、そのファイルの末尾に追加します。 **メモ** コマンドの出力と既存のファイルの構造が同一であるようにする必要があります。 • 同じフィールド • 同じフィールド順序 • 一致するフィールドが同じ長さ • 一致するフィールドが同じデータ型 出力は、既存ファイルとの間でファイル構造が違っている場合でも、Analytics によって既存ファイルに追加されます。出力と既存のファイルの構造が一致しない場合は、データが混在、不足、不正確になります。
COUNT 省略可能	レコード数が列に挿入されます。カウントは、SUBTOTAL を使用する場合に便利です。 小計フィールドが何も選択されていない場合は、自動的にカウントが含められます。

名前	説明
OPEN 省略可能	コマンドを実行した後、コマンドによって作成されたテーブルを開きます。コマンドが出力テーブルを作成する場合にのみ有効です。
LOCAL 省略可能	Analytics プロジェクトと同じ場所に出力ファイルを保存します。 **メモ** Analytics テーブルである出力ファイルを含むサーバー テーブルに対してコマンドを実行するときにのみ適用されます。
HEADER *ヘッダーテキスト* 省略可能	レポートの各ページの最上部に挿入されるテキスト。 *ヘッダーテキスト*は引用符で囲んだ文字列として指定する必要があります。この値は、Analytics のシステム変数である HEADER の値よりも優先されます。
FOOTER *フッターテキスト* 省略可能	レポートの各ページの最下部に挿入されるテキスト。 *フッターテキスト*は引用符で囲んだ文字列として指定する必要があります。この値は、Analytics のシステム変数である FOOTER の値よりも優先されます。

例

SUBTOTAL を使って売掛金(Ar)テーブルをクロス集計する

顧客番号(Customer_Number) フィールドと**取引タイプ(Trans_Type)** フィールドを基準にして売掛金(Ar)テーブルをクロス集計したいとします。また、**取引金額(Trans_Amount)** フィールドの値の小計も求めたいとします。

出力は顧客別にグループ化され、さらに各顧客内では取引タイプ別にグループ化されます。出力には、取引タイプごとに各顧客の総取引額が含まれます。

```
OPEN Ar
CROSSTAB ON Customer_Number COLUMNS Trans_Type SUBTOTAL Trans_Amount COUNT
TO SCREEN
```

売掛金(Ar)テーブルをクロス集計することで重複取引を検出する

売掛金(Ar)テーブルから重複取引の証拠を検出したいとします。

これを行うには、売掛金テーブルを**取引金額(Trans_Amount)** フィールドと**取引タイプ(Trans_Type)** フィールドに基づいてクロス集計します。出力では、取引タイプごとに同一の取引額がグループ化され、カウントされます。

```
OPEN Ar
CROSSTAB ON Trans_Amount COLUMNS Trans_Type TO SCREEN
```

備考

> **メモ**
> このコマンドの動作の詳細については、Analytics のヘルプを参照してください。

機能の仕組み

CROSSTAB は、2 つ以上の文字または数値フィールドの値で、同じ組み合わせを持つレコードをグループ化します。

出力には、ピボット テーブルと似た行と列で構成されるグリッドが含まれます。出力にはグループごとに 1 つの行と列の交差が含まれるほか、そのグループに属するソース テーブル内のレコードの数が含まれます。

並べ替えとCROSSTAB

CROSSTAB では、データは並べ替えられていてもいなくても処理できます。出力の*行フィールド*と*列フィールド*は、どちらも自動的に昇順に並べ替えられます。

複数の*行フィールド*を指定した場合、フィールドは、最初に指定された*行フィールド*を 1 番目の並べ替えフィールドとして、入れ子の並べ替えを使用します。

CVSEVALUATE コマンド

従来の変数サンプリングのために、サンプル分析の結果を母集団全体に対して推定するための4つの方法があります。

構文

CVSEVALUATE BOOKED *簿価フィールド* AUDITED *監査値フィールド* ETYPE {MPU|DIFFERENCE|RATIO SEPARATE|RATIO COMBINED} STRATA *境界値* <,...*n*> POPULATION *層カウント*,*層簿価* <,...*n*> CONFIDENCE *信頼度* CUTOFF *金額*,*確実性層数*,*確実性層簿価* ERRORLIMIT *数値* PLIMIT {BOTH|UPPER|LOWER} <TO SCREEN|*ファイル名*>

パラメーター

> **メモ**
>
> CVSPREPARE および CVSSAMPLE コマンドの出力結果を CVSEVALUATE コマンドの入力として使用している場合は、多数のパラメーター値が既に指定され、変数に格納されています。詳細については、"CVSPREPARE コマンド" ページ 110と"CVSSAMPLE コマンド" ページ 114を参照してください。
>
> 値を指定する際、3桁の区切り記号やパーセント記号は含めないでください。

名前	説明
BOOKED *簿価フィールド*	評価で使用する数値型の簿価フィールド
AUDITED *監査値フィールド*	評価で使用する数値型の監査金額フィールド
ETYPE MPU | DIFFERENCE | RATIO SEPARATE | RATIO COMBINED	使用する推定タイプ: • MPU(平均) • 差異 • 比率分離 • 比率結合 詳細については以下を参照 "どの推定タイプを使用すればよいですか?" ページ 108
STRATA *境界値* <,...*n*>	*簿価フィールド*を階層化するために使用される上限の境界値
POPULATION *層カウント*, *層値* <,...*n*>	*簿価フィールド*の各層のレコード数と合計値
CONFIDENCE *信頼度*	従来の変数サンプルの準備段階で使用される信頼度
CUTOFF *値*, *確実性層数*,	◦ ***値:*** -従来の変数サンプルの準備およびサンプル段階で使用される確実性層カットオフ値

名前	説明
確実性層簿価	◦ ***確実性層カウント:*** -確実性層のレコード数 ◦ ***確実性層値:*** -確実性層のレコードの合計簿価
ERRORLIMIT *数値*	サンプルで想定する最低誤謬数。 **メモ** サンプルを分析したときに見つかった実際の誤謬数がERRORLIMIT *数値*より小さい場合は、使用可能な評価方法は平均推定のみです。
PLIMIT BOTH | UPPER | LOWER	使用する精度制限のタイプ: ◦ BOTH: 上限および下限 ◦ UPPER: 上限 ◦ LOWER: 下限 詳細については、"CVSPREPARE コマンド" ページ 110を参照してください。
TO SCREEN | *ファイル名*	コマンドの結果を送信する場所: ◦ **SCREEN** - はAnalyticsの表示領域に結果を表示します ◦ ***ファイル名*** -は結果の保存先となるファイルです。 *ファイル名*には、適切なファイル拡張子を付けた文字列を引用符で囲んで指定します。例: `TO "Output.TXT"` デフォルトでは、テーブルファイルは、Analyticsプロジェクトが入っているフォルダーに保存されます。 既存の異なるフォルダーにファイルを保存するには、絶対または相対ファイルパスを指定します。 • `TO "C:\Output.TXT"` • `TO "Results\Output.TXT"`

例

レコードの母集団全体に対して、サンプルされたデータで見つかった誤謬を推定する

サンプルデータのテストを完了し、見つかった虚偽表示を記録しました。検出したすべての誤謬を母集団に対して投影できます。

以下の例は、サンプル分析の結果から母集団全体について推定するには、推定タイプDIFFERENCEを使用します。

```
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
```

備考

> **メモ**
>
> このコマンドの動作の詳細については、Analytics のヘルプを参照してください。

どの推定タイプを使用すればよいですか?

使用しなければならない推定タイプは、データの性質、サンプル簿価、サンプル監査金額、およびそれらの関係によって決まります。

ガイドライン

推定タイプの選択に役立つガイドラインを次に示します。複数の推定タイプについてはこの評価ステージを繰り返し、各推定タイプの評価結果同士を比較できます。

推定タイプ	虚偽表示の存在	虚偽表示のサイズ	簿価の符号	層間の比率の比較
平均推定	**虚偽表示がないか、非常に少ない虚偽表示しかない。** 監査したサンプル母集団に虚偽表示がないか、非常に少ない場合に、唯一有効な推定タイプ	該当なし	該当なし	該当なし
差異	**虚偽表示が必要** 監査したサンプル母集団にはいくつかの虚偽表示が必要です。 たとえば、サンプルの 5% 以上に虚偽表示が含まれているなどの場合です。	**虚偽表示が比例しない場合** 虚偽表示が関連する簿価に比例しない場合、つまり虚偽表示のサイズが関連する簿価のサイズに比例しない場合に適しています。 つまり、簿価の大小に関係なく、それに含まれる虚偽表示が大きかったり小さかったりする場合です。	該当なし	該当なし
比率分離		**虚偽表示が関連する簿価に比例する場合** 虚偽表示が関連する簿価に比例する場合、つまり虚偽表示のサイズが関連する簿価のサイズに比例する	**すべての簿価が同じ符号を持つ** サンプルの全簿価が同じ符合を持つ、つまりすべてプラスか、すべてマイナスかのどちらかである必要がありま	**比率が異なる** サンプルの監査金額の平均とサンプル簿価の平均との比率が層間で大きく異なる場合に適しています。

推定タイプ	虚偽表示の存在	虚偽表示のサイズ	簿価の符号	層間の比率の比較
		場合に適しています。 つまり、簿価が小さければ虚偽表示が小さく、簿価が大きければ虚偽表示が大きくなる場合です。	す。	
比率結合				**比率間の差異があまりない** サンプルの監査金額の平均とサンプル簿価の平均との比率が層間であまり差異がない場合に適しています。

CVSPREPARE コマンド

母集団を階層化し、各層の統計的に有効なサンプルサイズを従来の変数サンプリングのために計算します。

構文

CVSPREPARE ON *簿価フィールド* NUMSTRATA *数値* MINIMUM *層のサンプルサイズの最小値* PRECISION *値* CONFIDENCE *信頼度* <CUTOFF *値*> NCELLS *数値* PLIMIT {BOTH|UPPER|LOWER} ERRORLIMIT *数値* <MINSAMPSIZE *最低サンプルサイズ*> TO {SCREEN|*ファイル名*}

パラメーター

> **メモ**
> 値を指定する際、3桁の区切り記号やパーセント記号は含めないでください。

名前	説明
ON *簿価フィールド*	従来の変数サンプルの準備の基準として使用する数値型の簿価フィールド。
NUMSTRATA *数値*	*簿価フィールド*を数値的に階層化するために使用する層の数。 層の数は下限が1、上限が256です。 NUMSTRATA 1を指定してもCUTOFFを指定していない場合には、サンプルを抽出する前に母集団が階層化されなくなります。 **メモ** 層の数はNCELLSに対して指定されたセルの数 55% を超過できません。
MINIMUM *層サンプルの最低数*	各層からサンプリングする最小レコード数。 最低数を指定する理由が特にない限り、デフォルト値ゼロ(0)を使用してください。
PRECISION *値*	許容虚偽表示および勘定で想定される虚偽表示の間の差異である金額。 ◦ **許容虚偽表示** - サンプルフィールドの値として許容する最大の虚偽表示合計金額のことです。重大な虚偽表示額とまでは見なされません。 ◦ **推定虚偽表示額** - サンプルフィールドの値として許容する最大の虚偽表示合計金額のことです。 精度は勘定が公正に表示されていることの許容度の範囲を決定します。 精度を下げると、サンプルサイズを大きくする必要がある許容度の範囲(誤謬のマージン)が小さくなります。
CONFIDENCE *信頼度*	必要な信頼度。この信頼度で、結果のサンプルが母集団全体を表します。

名前	説明
	たとえば、95 を指定した場合は、サンプルが実際に95% の確率で母集団を代表しているとお客様が信頼したいということを意味します。信頼度は"サンプリングリスク" の補数です。信頼度が95% ということはサンプリングリスクが5% ということと同じです。 ◦ PLIMIT にBOTH を指定した場合には、最低信頼度は10%、最大信頼度は99.5% になります。 ◦ PLIMIT にUPPER またはLOWER を指定した場合には、最低信頼度は55%、最大信頼度は99.5% になります。
CUTOFF *値* 省略可能	確実性層のカットオフ値。 カットオフ値以上の*簿価フィールド*の金額が自動的に選択され、サンプルに取り込まれます。 CUTOFF を省略した場合は、*簿価フィールド*の最大金額に等しいデフォルトのカットオフ値が使用されます。
NCELLS *数値*	*簿価フィールド*をあらかじめ階層化するために使用するセルの数。 セルの数は、層の数より分割可能な数が少なくなります。階層化の前処理は、層の境界の位置を最適化する内部処理の一部です。最終的に階層化された出力には、セルは保持されません。 セルの数は下限が2、上限が999 です。 **メモ** セル数は少なくとも層のNUMSTRATA 数の2 倍である必要があります。
PLIMIT BOTH | UPPER | LOWER	使用する精度制限のタイプ。 ◦ BOTH - 次の場合にこのオプションを指定します。 • 勘定全体が過剰表示または過小表示である可能性がある • 方向のいずれかの虚偽表示が指定されたPRECISION を超過するかどうかを推定する ◦ UPPER - 次の場合にこのオプションを指定します。 • 全体としての勘定が過小評価されている可能性が高い • 過小評価の合計金額が指定されたPRECISION を超過するかどうかのみを推定する ◦ LOWER - 次の場合にこのオプションを指定します。 • 全体としての勘定が過大評価されている可能性が高い • 過大評価の合計金額が指定されたPRECISION を超過するかどうかのみを推定する **注意** 指定するオプションがわからない場合は、BOTH を指定します。
ERRORLIMIT *数値*	サンプルで想定する最低誤謬数。 **メモ** サンプルを分析したときに見つかった実際の誤謬数がERRORLIMIT に指定した*数値*より小さい場合は、使用可能な評価方法は平均推定のみです。
MINSAMPSIZE *最低サンプルサイズ* 省略可能	母集団全体からサンプリングする最低レコード数。 最低数を指定する理由が特にない限り、デフォルト値ゼロ(0) を使用してください。
TO SCREEN | *ファイル名*	コマンドの結果を送信する場所: ◦ **SCREEN** - はAnalytics の表示領域に結果を表示します ◦ ***ファイル名*** -は結果の保存先となるファイルです。

名前	説明
	*ファイル名*には、適切なファイル拡張子を付けた文字列を引用符で囲んで指定します。例：TO "Output.TXT" デフォルトでは、テーブル ファイルは、Analytics プロジェクトが入っているフォルダーに保存されます。 既存の異なるフォルダーにファイルを保存するには、絶対または相対ファイル パスを指定します。 • TO "C:\Output.TXT" • TO "Results\Output.TXT"

Analytics の出力変数

名前	含む
CONFIDENCE	ユーザーが指定した信頼度。
ERRLIMIT	ユーザーが指定した最低誤謬数。
NSTRATA	ユーザーが指定した層の数。
PLIMIT	ユーザーが指定した精度限度のタイプ。
S_TOP	ユーザーが指定した確実性層カットオフ値を格納します。何も指定されていない場合は、コマンドが計算した上位の層の上限境界。
SAMPLEFIELD	ユーザーが指定した簿価フィールド。
SBOTTOM	コマンドで計算された下限の層の下限境界線より低い値。
SBOUNDARY	コマンドで計算されたすべての層の上限境界および S_TOP。SBOTTOM は格納しないでください。
SPOPULATION	各層のレコード数のカウントと合計値。確実性層カットオフの上の項目を除外します。
SSAMPLE	コマンドによって計算された、各層のサンプル サイズ。

例

従来の変数サンプリングの準備

従来の変数サンプリングを使用して、請求書を含む勘定の金額虚偽表示の合計金額を推定することを決定しました。

サンプルを抽出する前に、母集団を階層化し、各層の統計的に有効なサンプル サイズを計算します。

Analytics によって抽出されるサンプルの 95% の時間が全体として母集団を表す信頼度が必要です。

指定された信頼度を使用して、以下の例は、**invoice_amount** フィールドを基準にしてテーブルを階層化し、確実性層と各層のサンプルサイズを計算します。

```
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
```

備考

> **メモ**
>
> このコマンドの動作の詳細については、Analytics のヘルプを参照してください。

数値の長さ制限

従来の変数サンプリングの準備段階では、いくつかの内部計算が行われます。これらの計算では、最大 17 桁の数値がサポートされます。計算結果が 17 桁を超える場合には、その計算結果が出力に含まれなくなるため、サンプリング処理を続行できなくなります。

注意：17 桁未満のソースデータの数値から、17 桁を超える内部計算結果が生成される場合もあります。

CVSSAMPLE コマンド

従来の変数サンプリング方法を使用して、レコードのサンプルを抽出します。

構文

CVSSAMPLE ON *簿価フィールド* NUMSTRATA *数値* <SEED *シード値*> CUTOFF *値* STRATA *境界値* <,...*n*> SAMPLESIZE *数値* <,...*n*> POPULATION *層カウント*,*層カウント* <,...*n*> TO *テーブル名*

パラメーター

> **メモ**
>
> CVSPREPARE コマンドの出力結果を CVSSAMPLE コマンドの入力として使用している場合は、多数のパラメーター値が既に指定され、変数に格納されています。詳細については、"CVSPREPARE コマンド" ページ 110を参照してください。
>
> 値を指定する際、3 桁の区切り記号やパーセント記号は含めないでください。

名前	説明
ON *簿価フィールド*	サンプルの基準として使用する数値簿価フィールド。
NUMSTRATA *数値*	*簿価フィールド*を階層化するために使用する層の数。
SEED *シード値* 省略可能	Analytics の乱数ジェネレーターを初期化するために使用するシード値。 SEED を省略した場合は、シード値がランダムに選択されます。
CUTOFF *値*	確実性層のカットオフ値。 カットオフ*値*以上の*簿価フィールド*の金額が自動的に選択され、サンプルに取り込まれます。
STRATA *境界値* <,...*n*>	*簿価フィールド*を階層化するために使用される上限の境界値
SAMPLESIZE *数値* <,...*n*>	各層からサンプリングするレコード数。
POPULATION *層カウント*, *層値* <,...*n*>	各層のレコード数と各層の合計値。
TO *テーブル名*	コマンドの結果を送信する場所： ◦ ***テーブル名*** -は、結果の保存先となる Analytics テーブルのことです。 *テーブル名*には、ファイル拡張子 .FIL を付けた文字列を引用符で囲んで指定する必要があります。例：TO "Output.FIL" デフォルトでは、テーブル データ ファイル(.fil) は、Analytics プロジェクトが入っているフォル

名前	説明
	ダーに保存されます。 既存の異なるフォルダーにデータ ファイルを保存するには、絶対または相対ファイル パスを指定します。 • TO "C:\Output.FIL" • TO "Results\Output.FIL" **メモ** テーブル名は 64 文字の英数字(.FIL 拡張子を含まない) に制限されています。名前にはアンダースコア文字(_) を使用できますが、他の特殊文字やスペースは使用できません。名前の先頭を数字にすることはできません。

Analytics の出力変数

名前	含む
S_TOPEV	ユーザーが指定した確実性層カットオフ値を格納します。何も指定されていない場合は、CVSPREPARE コマンドで以前に計算された最上位層の上限の境界。 また、確実性層と合計金額値のレコード数のカウントを格納します。
SBOTTOMEV	コマンドで計算された下限の層の下限境界線より低い値。
BOUNDARYEV	コマンドによってあらかじめ入力、またはユーザーが指定したすべての層の上限の境界。S_TOPEV または SBOTTOMEV を格納しないでください。
SPOPULATION	各層のレコード数のカウントと合計値。確実性層カットオフの上の項目を除外します。

例

従来の変数サンプリングの抽出

従来の変数サンプリングを使用して、請求書を含む勘定の金額虚偽表示の合計金額を推定します。

母集団を階層化し、各層の統計的に有効なサンプル サイズを計算した後に、サンプルを抽出できます。

以下の例は、**invoice_amount** フィールドを基準にしてレコードの階層化サンプルを抽出し、その階層化されたレコードを `Invoices_sample` てテーブルに出力します。

```
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"
```

備考

> **メモ**
>
> このコマンドの動作の詳細については、Analytics のヘルプを参照してください。

システム生成フィールド

Analytics により、自動的に4 つのフィールドが生成され、サンプル出力テーブルに追加されます。サンプルに追加される各レコードのフィールドには、以下の記述的情報が格納されます。

- **STRATUM** - レコードが割り当てられる層の数
- **ORIGIN_RECORD_NUMBER** - ソースデータテーブル内の元のレコード番号
- **SELECTION_ORDER** - 各層においてレコードがランダムに選択された順序
- **SAMPLE_RECORD_NUMBER** - サンプル出力テーブル内のレコード番号

DEFINE COLUMN コマンド

1 つ以上の列を作成し、既存のビューに追加します。

構文

DEFINE COLUMN *ビュー名 フィールド名* <AS *表示名*> <POSITION *n*> <WIDTH *文字*> <PIC *書式*> <SORT|SORT D> <KEY> <PAGE> <NODUPS> <NOZEROS> <LINE *n*>

パラメーター

名前	説明
ビュー名	列を追加するビュー。
フィールド名	列を作成するフィールド。 関連テーブルのフィールドを使用する場合は、「*テーブル名.フィールド名*」の形式でフィールド名を指定します。
AS *表示名* 省略可能	ビューにおけるフィールドの表示名(代替列見出し)。表示名をフィールド名と同じにしたい場合は、AS を使用しないでください。 *表示名*の値は引用符で囲まれた文字列。列見出しを改行したい場合は、語句の間にセミコロン(;)を入れます。
POSITION *n* 省略可能	ビュー内の列の位置を、左から右へ数えた数字。 ◦ 省略する場合、列は追加した時点で一番右の列として配置されます。 ◦ 指定した位置番号が見当たらない場合、列が連続して配置されるように列の位置を調整します。 ◦ 位置番号が既に使用されている場合、新しい列は現在その位置番号を使用している列の左側に配置されます。
WIDTH *文字数* 省略可能	フィールドの表示幅(文字数)。 指定した値によって、Analytics のビューおよびレポートにおけるフィールドの表示幅が決まります。表示幅はデータを変更するものではありませんが、表示幅がフィールド長より短い場合にはデータが隠れる可能性があります。 WIDTH を省略した場合には、表示幅は、テーブルレイアウトのフィールドに指定された幅に設定されます。

名前	説明
	メモ WIDTH で指定する文字数は、固定幅の文字数です。実際の文字の幅に関係なく、各文字には同じ空白量が割り当てられます。 Analytics のビューでは、固定幅の文字間隔と対応しないプロポーショナル フォントがデフォルトで使用されます。 WIDTH の値とビューの文字数との間に 1 対 1 の対応が必要な場合は、 **[オプション]**ダイアログ ボックスの **[プロポーショナル フォント]**設定を Courier New などの固定幅フォントに変更することができます。
PIC *書式* 省略可能	**メモ** 数値フィールドまたは日付時刻フィールドにのみ適用されます。 ◦ **数値フィールド** - Analytics のビューとレポートに含まれる数値の表示形式。 ◦ **日付時刻フィールド** -ソース データの日付時刻値の物理形式(日付時刻文字、区切り文字の順など) **メモ** 日付時刻フィールドの場合、*形式*はソース データの物理形式と正確に一致する必要があります。たとえば、ソース データが 12/31/2014 である場合は、書式を "MM/DD/YYYY" として入力します。 *書式*は引用符で囲む必要があります。
SORT | SORT D 省略可能	カラム(列) で並べ替えます。 ◦ **昇順** - SORT ◦ **降順** - SORT D
KEY 省略可能	列はレポートのブレーク フィールドに指定されています。列の値が変更されると、レポートは小計され、分割されます。ブレーク フィールドに対し次のような制限があります。 ◦ 文字フィールドか式である必要があります。 ◦ フィールドにブレーク列を設定する場合、その列は一番左の列でなければなりません。 ◦ ビュー内の最後の列をブレーク フィールドにすることはできません。 ◦ 複数の列をブレーク フィールドにする場合、追加するブレークフィールドの左側にある列もすべてブレークフィールドにしなければなりません。
PAGE 省略可能	ブレークフィールド値が変化するたびに改ページを挿入します。
NODUPS 省略可能	フィールドの繰り返し値に空白値を使用します。 たとえば、請求書レコードごとに顧客名が記載されている場合、各顧客名の最初のインスタンスだけを記載すれば、レポートが読みやすくなるかもしれません。
NOZEROS 省略可能	フィールドのゼロ値に空白値を使用します。 たとえば、レポートのフィールド大量のゼロ値がある場合、ゼロ以外の値のみを表示すると読みやすくなります。
LINE *n* 省略可能	列の行数。値が指定されない場合、列は単一行をデフォルトとします。*n* の値は 2 から 60 の間で指定する必要があります。

例

6 つの列を使ってビューを定義する

AR テーブルを開いた状態で、6 つの列を使って AR_Report というビューを定義します。これらの列は、指定した順序で表示されます。

```
OPEN Ar
DEFINE VIEW AR_Report OK
DEFINE COLUMN AR_Report No AS "顧客番号" WIDTH 7 KEY
DEFINE COLUMN AR_Report Date AS "請求日" WIDTH 10
DEFINE COLUMN AR_Report Due AS "支払期日" WIDTH 10
DEFINE COLUMN AR_Report Reference AS "参照番号" WIDTH 6
DEFINE COLUMN AR_Report Type AS "取引タイプ" WIDTH 5
DEFINE COLUMN AR_Report Amount AS "取引金額" WIDTH 12 PIC "-9999999999.99"
```

DEFINE FIELD コマンド

Analytics テーブル レイアウトで物理データ フィールドを定義します。

構文

DEFINE FIELD *フィールド名 データ型 開始位置 長さ<小数点以下桁数|日付書式>*
<NDATETIME> <PIC *書式*> <AS *表示名*> <WIDTH *バイト数*> <SUPPRESS> <*フィールド メモ*>

パラメーター

名前	説明
フィールド名	フィールドの名前。 **メモ** フィールド名は 256 文字までの小文字の英字に制限されます。名前にはアンダースコア文字 (_) を使用できますが、他の特殊文字やスペースは使用できません。名前の先頭を数字にすることはできません。 Analytics には多くの予約キーワードがあり、フィールド名にこのキーワードを使用することはできません。詳細については、"予約キーワード" ページ 928を参照してください。
データ型	データを解釈するときに使用するデータ型。サポートされるデータ型の一覧については、"サポートされているデータ型" ページ 125を参照してください。 たとえば、請求書番号がソース内に数値として格納されているとします。これらの値を数値ではなく、文字列として処理するには、文字データとしてフィールドを定義できます。
開始位置	Analytics データ ファイル内のフィールドの開始バイトを指定します。 **メモ** 非 Unicode 版 Analytics: 1 バイト = 1 文字 Unicode 版 Analytics、拡張 ASCII (ANSI) データ: 1 バイト = 1 文字 Unicode 版 Analytics、Unicode データ: 2 バイト = 1 文字 Unicode データでは、一般的に、奇数で開始するバイト位置を指定してください。偶数の開始位置を指定すると、文字が正しく表示されない可能性があります。
長さ	フィールド長のバイト数。

名前	説明
	メモ 非 Unicode 版 Analytics: 1 バイト = 1 文字 Unicode 版 Analytics、拡張 ASCII (ANSI) データ: 1 バイト = 1 文字 Unicode 版 Analytics、Unicode データ: 2 バイト = 1 文字 Unicode データでは、偶数バイトのみを指定します。奇数バイトを指定すると、文字が正しく表示されない可能性があります。
小数位 省略可能	数値フィールドの小数点以下の桁数
日付書式 省略可能	ソース日付フィールドでの日付書式。 日付時刻フィールドまたは時刻フィールドについては、代わりに PIC *書式*を使用します。PIC *書式*は、日付フィールドにも使用できます。 ソース データにスラッシュなどの区切り文字が含まれている場合は、*日付書式*書式にその区切り文字を含める必要があります。たとえば、ソース データが 12/31/2014 である場合は、書式を MM/DD/YYYY として入力します。*日付書式*を引用符で囲まないでください。
NDATETIME 省略可能	数値フィールドに格納されている日付、日付時刻、または時刻の値を日付時刻データとして扱います。 NDATETIME を使用するには、PIC *書式*を使ってソースの日付時刻書式も指定されている必要があります。
PIC *書式* 省略可能	**メモ** 数値フィールドまたは日付時刻フィールドにのみ適用されます。 ◦ **数値フィールド** - Analytics のビューとレポートに含まれる数値の表示形式。 ◦ **日付時刻フィールド** -ソース データの日付時刻値の物理形式(日付時刻文字、区切り文字の順など) **メモ** 日付時刻フィールドの場合、*形式*はソース データの物理形式と正確に一致する必要があります。たとえば、ソース データが 12/31/2014 である場合は、書式を "MM/DD/YYYY" として入力します。 *書式*は引用符で囲む必要があります。
AS *表示名* 省略可能	ビューにおけるフィールドの表示名(代替列見出し) 。表示名をフィールド名と同じにしたい場合は、AS を使用しないでください。 *表示名*の値は引用符で囲まれた文字列。列見出しを改行したい場合は、語句の間にセミコロン(;) を入れます。
WIDTH *文字数* 省略可能	フィールドの表示幅(文字数) 。 指定した値によって、Analytics のビューおよびレポートにおけるフィールドの表示幅が決まります。表示幅はデータを変更するものではありませんが、表示幅がフィールド長より短い場合には

名前	説明
	データが隠れる可能性があります。 このため、表示幅は*フィールド名*や*表示名*の長さより短い値を指定しないでください。 WIDTH を省略した場合には、表示幅はフィールド長の文字数に設定されます。 **メモ** WIDTH で指定する文字数は、固定幅の文字数です。実際の文字の幅に関係なく、各文字には同じ空白量が割り当てられます。 Analytics のビューでは、固定幅の文字間隔と対応しないプロポーショナルフォントがデフォルトで使用されます。 WIDTH の値とビューの文字数との間に 1 対 1 の対応が必要な場合は、**[オプション]**ダイアログ ボックスの **[プロポーショナル フォント]**設定を Courier New などの固定幅フォントに変更することができます。
SUPPRESS 省略可能	数値フィールドにのみ適用されます。 Analytics レポートの数値フィールドの自動合計を抑制します。 一部の数値フィールドの合計は適切ではありません。たとえば、単価フィールドまたは値引率フィールドです。
フィールドノート 省略可能	テーブル レイアウトでフィールド定義に追加するフィールド ノートのテキスト。 *フィールドノート*は、その他すべての必須およびオプション パラメーターよりも後の、最後に指定する必要があります。テキストは、複数行にすることはできません。引用符は必要ありません。

例

文字フィールドの定義

文字フィールド **ProdDesc** を定義します。ビューでのこの列の見出しは、**製品説明** です。

非 Unicode 版 Analytics

- 開始: バイト 12(文字位置 12)
- 長さ: 24 バイト(24 文字)

```
DEFINE FIELD ProdDesc ASCII 12 24 AS "製品説明"
```

Unicode 版 Analytics、拡張 ASCII (ANSI) データ

- 開始: バイト 12
- 長さ: 24 バイト(24 文字)

```
DEFINE FIELD ProdDesc ASCII 12 24 AS "製品説明"
```

Unicode 版 Analytics、Unicode データ

- 開始：バイト 13
- 長さ：48 バイト（24 文字）

```
DEFINE FIELD ProdDesc UNICODE 13 48 AS "製品説明"
```

数値フィールドの定義

数値フィールド **QtyOH** を定義します。ビューでは、列は指定された表示書式を使用し、ビューの列見出しは **在庫数** です。

- 開始：バイト 61
- 長さ：10 バイト
- 小数点以下桁数：なし

```
DEFINE FIELD QtyOH NUMERIC 61 10 0 PIC "(9,999,999)" AS "在庫数"
```

文字データからの日付時刻フィールドの定義

ソース文字データから、以下の最初の 2 つの例は、日付時刻フィールドの**取引日**を定義します。ソースデータでは、日付形式は DD/MM/YYYY です。列見出しは、指定されていないため、デフォルトでフィールド名になります。

- 開始：バイト 20
- 長さ：10 バイト

ここでは、*日付形式* を使用して、日付形式が指定されます。

```
DEFINE FIELD Transaction_date DATETIME 20 10 DD/MM/YYYY
```

ここでは、PIC *形式* を使用して、日付形式が指定されます。

```
DEFINE FIELD Transaction_date DATETIME 20 10 PIC "DD/MM/YYYY"
```

時刻データを含む日付時刻フィールドを定義するときには、PIC *形式*を使用する必要があります。

以下の例は、日付時刻フィールドの**電子メール タイムスタンプ**を定義します。ソースデータでは、日付時刻形式は、YYYY/MM/DD hh:mm:ss-hh:mm です。

- 開始：バイト 1
- 長さ：25 バイト

```
DEFINE FIELD email_timestamp DATETIME 1 25 PIC "YYYY/MM/DD hh:mm:ss-hh:mm"
```

数値データからの日付時刻フィールドの定義

次の例では、ソースの数値型データを使用して、そのソースデータに指定されている日付書式を持っていて、

Receipt_timestamp という日付時刻型フィールドを定義しています。

- 開始：バイト 15
- 長さ：15 バイト

```
DEFINE FIELD Receipt_timestamp DATETIME 15 15 PIC "YYYYMMDD.hhmmss"
```

"数値" 日付時刻フィールドの定義

次の例では、ソースの数値型データを使用して、そのソースデータに指定されている日付書式を持っていて、**Receipt_timestamp** という数値型フィールドを定義しています。

次の例では、NDATETIME パラメーターにより、Analytics が、数値フィールドに格納されている日付時刻値を日付時刻データとして扱うことができるようになっています。

- 開始：バイト 15
- 長さ：15 バイト
- 小数点以下桁数：6

```
DEFINE FIELD Receipt_timestamp PRINT 15 15 6 NDATETIME PIC "YYYYMMDD.hhmmss"
```

メインフレームのパックデータを読み取る物理データフィールドを定義する

NDATETIME オプションを使用して、パックされた数値フィールドから日付値を読み取る物理データフィールドを作成できます。

Analytics では、1 桁あたり 1 バイト未満に圧縮された、日付書式でない数値の日付は認識できません。したがって、NDATETIME で数値をアンパックしてすべての桁の数字を取得してから、PIC で日付書式を指定する必要があります。

それぞれの数字と年、月、日の対応を正確に示すには、パック型レコードレイアウトの日付書式と同じものを指定します。

```
DEFINE FIELD 日付フィールド名 NUMERIC 1 8 0 NDATETIME PIC "YYYYMMDD"
```

備考

> **メモ**
> このコマンドの動作の詳細については、Analytics のヘルプを参照してください。

スクリプトを使ってフィールドを上書きする

既存のフィールドと同じ名前を使用するフィールドを定義することによって、テーブルレイアウト内のフィールドを上書きすることができます。SET SAFETY ON を実行した場合は、既存のフィールドを上書きする前に確認ダイアログボックスが表示されます。

スクリプトが中断されないようにするには、SET SAFETY OFF を実行します。これにより、追加の確認が行われることなく、既存のフィールドが上書きされます。

サポートされているデータ型

データ カテゴリ	データ型
文字	ASCII
	CUSTOM
	EBCDIC
	NOTE
	PCASCII
	UNICODE
数値	ACCPAC
	ACL
	BASIC
	BINARY
	FLOAT
	HALFBYTE
	IBMFLOAT
	MICRO
	NUMERIC
	PACKED
	PRINT
	UNISYS
	UNSIGNED
	VAXFLOAT
	ZONED

データ カテゴリ	データ型
日付時刻	DATETIME
論理	LOGICAL

DEFINE FIELD . . . COMPUTED コマンド

Analytics テーブル レイアウトの演算フィールドを定義します。

構文

演算フィールドを定義するには

```
DEFINE FIELD フィールド名 COMPUTED 式
```

オプション パラメーターを持つ演算フィールドを定義するには

```
DEFINE FIELD フィールド名 COMPUTED
<IF テスト> <STATIC> <PIC 書式> <AS 表示名> <WIDTH 文字> <SUPPRESS> <フィールド メモ>
式
```

条件付き演算フィールドを定義するには

```
DEFINE FIELD フィールド名 COMPUTED
*** BLANK_LINE ***
値 IF 条件
<値 IF 条件>
<...n>
デフォルト値
```

オプション パラメーターを持つ条件付き演算フィールドを定義するには

```
DEFINE FIELD フィールド名 COMPUTED
<IF テスト> <STATIC> <PIC 書式> <AS 表示名> <WIDTH 文字> <SUPPRESS> <フィールド メモ>
値 IF 条件
<値 IF 条件>
<...n>
デフォルト値
```

> **メモ**
>
> 上記の一般的な構文は、以下の例に示すように、複数行の構文は正確に構造化される必要があります。

パラメーター

名前	説明
フィールド名	演算フィールド名。 **メモ** フィールド名は256文字までの小文字の英字に制限されます。名前にはアンダースコア文字(_) を使用できますが、他の特殊文字やスペースは使用できません。名前の先頭を数字にすることはできません。 Analyticsには多くの予約キーワードがあり、フィールド名にこのキーワードを使用することはできません。詳細については、"予約キーワード" ページ 928を参照してください。
式	演算フィールドの値を定義する有効なAnalytics式。
IF *テスト* 省略可能	各レコードを処理するために真である必要がある条件式。コマンドは、その条件を満たすレコードに対してのみ実行されます。 **メモ** IFパラメーターは、任意の範囲パラメーター(WHILE、FIRST、NEXT) が適用された後に、テーブルに残るレコードに対してのみ評価されます。
STATIC 省略可能	異なる値を検出するまで、テーブル内の対象フィールドの各行にはすべて同じ値が表示されます。 たとえば、次のソースデータに姓フィールドがある場合: ◦ 最初のレコードが値 "Smith"を表示する ◦ 次の5レコードが空白の行を表示する ◦ 7番目のレコードが値 "Wong" を表示する この場合、"Smith" は6つの連続する行に表示され、"Wong" は7番目の行に表示されます。
PIC *書式* 省略可能	**メモ** 数値型のフィールドにのみ適用されます。 Analyticsのビューおよびレポートに含まれる数値の表示形式。 *書式*は引用符で囲む必要があります。
AS *表示名* 省略可能	ビューにおけるフィールドの表示名(代替列見出し) 。表示名をフィールド名と同じにしたい場合は、ASを使用しないでください。 *表示名*の値は引用符で囲まれた文字列。列見出しを改行したい場合は、語句の間にセミコロン(;) を入れます。
WIDTH *文字数* 省略可能	フィールドの表示幅(文字数) 。 指定した値によって、Analyticsのビューおよびレポートにおけるフィールドの表示幅が決まります。表示幅はデータを変更するものではありませんが、表示幅がフィールド長より短い場合にはデータが隠れる可能性があります。 このため、表示幅には*フィールド名*や*表示名*の長さより短い値を指定しないでください。 WIDTHを省略した場合には、表示幅はフィールド長の文字数に設定されます。

名前	説明
	メモ WIDTH で指定する文字数は、固定幅の文字数です。実際の文字の幅に関係なく、各文字には同じ空白量が割り当てられます。 Analytics のビューでは、固定幅の文字間隔と対応しないプロポーショナルフォントがデフォルトで使用されます。 WIDTH の値とビューの文字数との間に 1 対 1 の対応が必要な場合は、**[オプション]**ダイアログボックスの **[プロポーショナル フォント]**設定を Courier New などの固定幅フォントに変更することができます。
SUPPRESS 省略可能	数値型のフィールドにのみ適用されます。 Analytics レポートの数値演算フィールドの自動合計を抑制します。 一部の数値フィールドの合計は適切ではありません。たとえば、単価フィールドまたは値引率フィールドです。
フィールドノート 省略可能	テーブルレイアウトでフィールド定義に追加するフィールド ノートのテキスト。 *フィールドノート*は、その他すべての必須およびオプション パラメーターよりも後の、最後に指定する必要があります。テキストは、複数行にすることはできません。引用符は必要ありません。
値 IF *条件*	条件付き演算フィールドのみ。 ◦ ***値*** -、*条件*が True と評価される場合に使用する演算フィールド値または式。 ◦ ***条件*** -評価される論理テスト。
デフォルト値	条件付き演算フィールドのみ。 条件のどれも True と評価されない場合に演算フィールドで使用する値または式。 **メモ** すべての数値演算値の小数精度は、*デフォルト値* の精度で統制されます。たとえば、既定値として 0.00 を指定すると、演算値はすべて小数点以下 2 桁まで計算され、必要に応じて四捨五入されます。精度を高めるには、*デフォルト値* の小数点以下桁数を増やします。

例

演算フィールドを定義する

Cost フィールドと **Quantity** フィールドの積を計算した **Value** フィールドを定義するには、次のように指定します。

```
DEFINE FIELD Value COMPUTED Cost * Quantity
```

オプションを使って演算フィールドを定義する

7 つのオプションを定義することで、**Value_03** という演算フィールドを定義するとします。演算フィールドで処理されるレコードを制限する IF 条件を追加します。

```
DEFINE FIELD Value_03 COMPUTED
IF Product_Class = "03" PIC "($9,999,999.99)" AS "Value Prod Class 3" 値はコストを数量で乗算した値です
Cost * Quantity
```

条件付き演算フィールドを定義する

取引が発生した州に応じて異なる消費税を計算する条件演算フィールド **Sales_tax** を定義するとします。3つの州以外で発生した取引の消費税はデフォルトの$0.00です。

> **メモ**
>
> 2番目の行は空白にする必要があります。省略可能なパラメーターはありません。

```
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
```

オプションを使って条件付き演算フィールドを定義する

取引が発生した州に応じて異なる消費税を計算する条件演算フィールド **Sales_tax_100** を定義するとします。このフィールドでは、$100以上の金額にかかる税のみを計算します。

3つの州以外で発生した取引の消費税はデフォルトの$0.00です。

> **メモ**
>
> 省略可能なパラメーターを指定するときには、改行を残さないでください。

```
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
```

備考

> **メモ**
>
> このコマンドの動作の詳細については、Analytics のヘルプを参照してください。

2 種類の演算フィールド

演算フィールドには 2 種類あります。

- **標準演算フィールド**

  標準演算フィールドはテーブルのすべてのレコードに対して同じ計算を実行します。

  たとえば、Inventory テーブルでは、Cost フィールドの値を Quantity フィールドの値で乗算し、各レコードのコストで棚卸資産の価額を計算する演算フィールドを作成できます。

- **条件付き演算フィールド**

  条件付き演算フィールドは、指定する条件群に基づいて、テーブルのレコードに対して異なる計算を実行できます。レコードに対して実行された計算は、レコードが満たす条件によって異なります。

  たとえば、Transactions テーブルで、取引が発生した州に基づいて調整されるレートを使用して、売上税を計算する条件付き演算フィールドを作成できます。`IF State = "CA"` や `IF State = "NY"` などの条件が各レコードをテストし、使用するレートを特定します。

条件付き演算フィールドの作成のためのガイドライン

> **メモ**
>
> 条件付き演算フィールドを定義するときには、2 行目にオプションのパラメーターをどれも指定しない場合は、2 行目を**空白**のままにしておいてください。

条件付きの演算フィールドを作成するには、デフォルト値に加えて、少なくとも 1 つの条件値が必要です。条件付きの演算フィールドを定義するには、次のような複数行構文を使用する必要があります。

- オプションのパラメーターは 2 行目に配置します
- オプションのパラメーターがない場合、2 行目が空白になります
- 最初の条件は 3 行目に配置します
- 各追加条件ステートメントは別々の行に配置します
- デフォルト値は最後の行に配置します

フィールド定義の上書き

既存のフィールドと同じ名前を使用するフィールドを定義することによって、テーブルレイアウト内のフィールド定義を上書きすることができます。

SET SAFETY を ON にした場合は、既存のフィールドを上書きする前に確認ダイアログボックスが表示されます。スクリプトが中断されないようにするには、SET SAFETY を OFF に設定します。そのようにすると、既存のフィールドが、確認を求められることなく上書きされます。

DEFINE RELATION コマンド

2 つの Analytics テーブル間の関連付けを定義します。

> **メモ**
>
> 最大 18 の Analytics テーブルを関連付け、それらのテーブルのフィールドは単独のテーブルに存在しているかのように扱うことができ、任意に組み合わせてデータにアクセスし分析することが可能です。関連するテーブルの各ペアの個別の DEFINE RELATION コマンドを指定する必要があります。

構文

DEFINE RELATION *キーフィールド名* WITH *関連テーブル名* INDEX *インデックス名* <AS *リレーション名* >

パラメーター

名前	説明
キーフィールド名	親テーブルのキーフィールド。 指定できるキーフィールドは、各リレーションにつき 1 つのみです。 > **メモ** > 親テーブルと孫テーブルの関係を作成するときには、*テーブル名.フィールド名* の形式で完全修飾キーフィールド名を指定する必要があります。 > "3 つのテーブルを関連付ける" 見開きページで、Vouchers.created_by を参照してください。
WITH *関連テーブル名*	関連テーブルの名前。
INDEX *インデックス名*	関連テーブルのキーフィールドのインデックスの名前。 テーブルを関連付ける前に、キーフィールドを基に関連テーブルにインデックスを作成する必要があります。
AS *リレーション名* 省略可能	一意のリレーション名。 デフォルトでは、子テーブルの名前がリレーション名となります。同じ子テーブルに対し、さらに関連付けを追加して定義する場合は、一意な名前を指定する必要があります。

例

2 つのテーブルの関連付け

以下の例では、顧客番号フィールド(**CustNum**)をキーフィールドに指定して、現在開いているテーブルを**Customer**テーブルと関連付けます。

```
DEFINE RELATION CustNum WITH Customer INDEX Customer_on_CustNum
```

Customer_on_CustNumは、キーフィールドの子テーブルインデックスの名前です。子テーブルインデックスはテーブルを関連付けるときに必要です。

子テーブルインデックスがDEFINE RELATIONコマンドを実行するときに存在していない場合は、エラーメッセージが表示され、関連付けが実行されません。

> **ヒント**
>
> Analytics のユーザーインターフェイスで関係を定義する場合、子テーブルインデックスが自動的に作成されます。

2 つのテーブルを関連付ける前に子テーブル インデックスを作成できます。

必要に応じて、2つのテーブルを関連付ける直前に子テーブルインデックスを作成できます。次の例は、**Ar**テーブルを**Customer**テーブルに関連付ける前に、**Customer**子テーブルでインデックスを作成する方法を示します。

```
OPEN Customer
INDEX ON CustNum TO Customer_on_CustNum
Open Ar
DEFINE RELATION CustNum WITH Customer INDEX Customer_on_CustNum
```

3 つのテーブルを関連付ける

以下の例は、`ACL_Rockwood.ACL`サンプルプロジェクトで3つのテーブルを関連付けます。

- **Vouchers_items** - 親テーブル
- **Vouchers** - 子テーブル
- **Employees** - 孫テーブル

Vouchersテーブルを関係の中間テーブルとして使用すると、各伝票項目を、項目を処理した従業員に関連付けることができます。

```
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
```

構文ロジックの説明

1. Vouchers テーブルを開き、voucher_number フィールドにインデックスを作成します。
2. Vouchers_items テーブルを開き、voucher_number をキーフィールドとして使用し、Vouchers テーブルに関連付けます。
3. Employees テーブルを開き、employee_number フィールドにインデックスを作成します。
4. Vouchers_items テーブルを開き、Vouchers.created_by をキーフィールドとして使用し、Employees テーブルに関連付けます。

> **メモ**
>
> Vouchers.created_by は 2 番目の関係でキーフィールドとして使用できます。最初の関係で Vouchers_items と Vouchers を既に関連付けているためです。

備考

> **メモ**
>
> このコマンドの動作の詳細については、Analytics のヘルプを参照してください。

DEFINE REPORT コマンド

新しいビューを作成するか、既存のビューを開きます。

構文

```
DEFINE REPORT ビュー名
```

パラメーター

名前	説明
ビュー名	既存のビューまたは新しいビューの名前。 ◦ **新しいビュー** -は、開いているテーブルで、指定された名前の空のビューを作成します ビュー名のスペースは、アンダースコア文字に置換されます。 ◦ **既存のビュー** -は、開いているテーブルで指定されたビューを開きます

例

新しいビューを作成する

新しいビュー **Q4_AR_review** を使用します。

```
DEFINE REPORT Q4_AR_review
```

DEFINE TABLE DB コマンド

AX コネクターを使用してデータベース テーブルに接続し、Analytics サーバー テーブルを定義します。Microsoft SQL Server、Oracle、DB2 データベースに接続することができます。

構文

DEFINE TABLE DB {SOURCE *データベース プロファイル*<PASSWORD *数値*> <PASSWORD *数値*> | SERVER *サーバー プロファイル*<PASSWORD *数値*>} <FORMAT *書式名*> SCHEMA *スキーマ* <TITLED *ACL テーブル名*> <PRIMARY|SECONDARY> DBTABLE *DB テーブル名* FIELDS {*フィールド名*|ALL} <...*n*> <WHERE *条件*> <ORDER *フィールド名*>

パラメーター

SOURCE *データベース プロファイル*	データベース エンジンにアクセスするために使用する Analytics データベース プロファイル。 データベース プロファイルには、次のように、データベース エンジンに接続するために必要な情報があります。 ◦ 関連付けられたサーバー プロファイルへの参照 ◦ データベースのタイプ ◦ データベースの名前 ◦ ユーザー アカウント情報 **メモ** DEFINE TABLE DB では、Microsoft SQL Server、Oracle、DB2 データベースにのみ接続できます。
PASSWORD *番号* 省略可能	使用するパスワード定義。 実際のパスワードを入力することを求めたり指定したりするのには、PASSWORD *番号*の構文は使用しません。パスワード定義とは、以前に PASSWORD コマンドか SET PASSWORD コマンドを使用して入力または設定されたパスワードのことです。 *番号*はパスワード定義の番号です。たとえば、以前に 2 つのパスワードをスクリプトで設定したり入力したりしている場合には、PASSWORD 2 により、2 番目のパスワードを使用することを指定するなどします。 パスワードの入力または設定の詳細については、次を参照してください。 • "PASSWORD コマンド" ページ 345 • "SET コマンド" ページ 404 • PASSWORD アナリティクス タグ パスワードの入力または設定の詳細については、次を参照してください。 • PASSWORD コマンド • SET コマンド • PASSWORD アナリティクス タグ このパスワードは、データベース プロファイルにパスワードを指定して保存していない場合にのみ指定する必要があります。SOURCE キーワードの後に PASSWORD を 2 回使用します。1 番目のパスワードでサーバーにログオンし、2 番目のパスワードでデータベースにログオンします。

SERVER *サーバープロファイル*	使用されなくなりました。 バージョン 10.0 より前の Analytics では、ACL Server Edition for z/OS へ接続するときに使用されていました。バージョン 10.0 以降の Analytics には、ACL Server Edition for z/OS が同梱されなくなりました。
FORMAT *フォーマット名* 省略可能	使用したいテーブル レイアウトを持つ、Analytics テーブルまたはテーブル レイアウト ファイル(.layout) の名前。
SCHEMA スキーマ	接続するスキーマ。スキーマ名は引用符で囲む必要があります。
TITLED *ACLテーブル名* 省略可能	作成する Analytics テーブルの名前。 *ACLテーブル名*は引用符で囲んだ文字列として指定する必要があります。TITLED を省略すると、データベースのテーブル名が使用されます。一度に複数のテーブルにアクセスした場合は、最初のテーブル名が使用されます。
PRIMARY | SECONDARY 省略可能	複数ファイル コマンド内でテーブルを主テーブルと副テーブルのどちらとして使用します。どちらのオプションも指定されない場合、デフォルト値の PRIMARY が使用されます。
DBTABLE *データベーステーブル名*	アクセスするデータベース テーブル。*データベーステーブル名*は引用符で囲んだ文字列として指定する必要があります。
FIELDS *フィールド名* | ALL	出力に含めるフィールド: ○ **FIELDS *フィールド名*** - 指定されたフィールド *フィールド名*は引用符で囲まれた文字列である必要があります。 ○ **ALL** - テーブルのすべてのフィールドを使用します。 複数のテーブルからフィールドを使用するには: a. 最初のテーブル名、そのテーブルのフィールドの順に入力します。 b. 次のテーブル名、そのテーブルのフィールドの順に入力します。 c. テーブルを追加するごとに手順 b を繰り返します。 `DBTABLE "DSN1310" FIELDS "Field_A Field_B Field_C" DBTABLE "DSN2516" FIELDS "Field_L Field_M Field_N"` **メモ** AX Connector を使用すると、無制限の関連したテーブルにアクセスすることができますが、5 つ以下のアクセスを推奨します。複数のテーブルにアクセスすると処理時間が増えます。
WHERE *条件* 省略可能	対象となるデータを、指定した条件を満たすレコードに限定する SQL WHERE 節。 引用符で囲まれた文字列として入力された有効な SQL 構文を使用する必要があります。 複数のテーブルを結合しようとすると、WHERE 句に結合条件が表示されます。

	"Table_1.First_name = Table_2.First_name"
ORDER *複数のフィールド名* 省略可能	データベース エンジンがレコードの並べ替えに使用するフィールド。*フィールド名*は引用符で囲んだ文字列として指定する必要があります。 レコードを並べ替えるときにはコマンドの実行に時間がかかります。並べ替えが重要であるときにのみ ORDER を使用します。

例

例

Microsoft SQL Server データベース内のデータにAX Connector 経由でアクセスしたいとします。それには、DEFINE TABLE DB コマンドを使用します。データベース プロファイルを使用してAX Connector に接続する SOURCE パラメーターも指定します。

```
DEFINE TABLE DB SOURCE "SQLServer_Audit" SCHEMA "HR" TITLED "Payroll" DBTABLE "HR.Employee" FIELDS "EmployeeID" DBTABLE "HR.EmployeePayHistory" FIELDS "Rate PayFrequency" WHERE "HR.Employee.EmployeeID=HR.EmployeePayHistory.EmployeeID"
```

備考

機能の仕組み

Analytics サーバー テーブルは、データベース プロファイルを使用してデータベース テーブルに接続するためのクエリとして定義されます。

日付時刻値の時刻部分を非表示にする

日付時刻値の時刻部分を非表示にするには、DEFINE TABLE DB コマンドの前にSET SUPPRESSTIME コマンドを置きます。

SET SUPPRESSTIME ON は、バージョン 10.0 より前の Analytics スクリプトで使用することを目的としています。それらのスクリプトでは、日付時刻値の時刻部分は切り捨てられることが前提となっています。それらのスクリプトは、SET SUPPRESSTIME ON を追加しない場合には、日付時刻型対応バージョンの Analytics では動作することができません。

詳細については、"SET コマンド" ページ 404の「SET SUPPRESSTIME」セクションを参照してください。

DEFINE VIEW コマンド

新しいビューの定義または既存のビューを上書きします。

構文

DEFINE VIEW *ビュー名* <RLINES *n*> <ALL> <SUPPRESS> <SUMMARIZED> <IF *テスト*> <WHILE *テスト*> <HEADER *ヘッダーテキスト*> <FOOTER *フッターテキスト*> <TO *レポート ファイル名* <HTML> > <OK>

パラメーター

名前	説明
ビュー名	作成または上書きするビューの名前。 **メモ:** ビューの名前は、64 文字までの英数字に制限されます。名前にはアンダースコア文字(_) を使用できますが、他の特殊文字やスペースは使用できません。名前の先頭を数字にすることはできません。
RLINES *n* 省略可能	ビューやレポートの詳細レコードの行間隔。デフォルトでは、詳細行の行間隔は 1 行です。
ALL 省略可能	アクティブな Analytics テーブル レイアウト内のすべてのフィールドがビューに追加されます。
SUPPRESS 省略可能	ビューから生成されるレポートで、空白の詳細行を表示させません。レポートの生成時、空白の詳細行は出力から除外されます。このオプションは、複数行ビューを基にしたレポートに適用されます。
SUMMARIZED 省略可能	ビューから生成されるレポートに小計と合計は含めるが、詳細行を含めないことを指定します。 小計は、ビューで定義されているブレークフィールドを基に生成されます。このオプションを指定しないと、レポートには、指定した各ブレークフィールドの小計と併せて詳細行も含まれます。
IF *テスト* 省略可能	各レコードを処理するために真である必要がある条件式。コマンドは、その条件を満たすレコードに対してのみ実行されます。 **メモ** IF パラメーターは、任意の範囲パラメーター(WHILE、FIRST、NEXT) が適用された後に、テーブルに残るレコードに対してのみ評価されます。
WHILE *テスト*	各レコードを処理するために真である必要がある条件式。条件が false と評価するか、テーブルの最後に達したら、コマンドは実行を中止します。

名前	説明
省略可能	**メモ** WHILE を FIRST または NEXT とともに使用する場合は、1 つの制限に達するとすぐに、レコードの処理が停止します。
HEADER *ヘッダーテキスト* 省略可能	レポートの各ページの最上部に挿入されるテキスト。 *ヘッダーテキスト*は引用符で囲んだ文字列として指定する必要があります。この値は、Analytics のシステム変数である HEADER の値よりも優先されます。
FOOTER *フッターテキスト* 省略可能	レポートの各ページの最下部に挿入されるテキスト。 *フッターテキスト*は引用符で囲んだ文字列として指定する必要があります。この値は、Analytics のシステム変数である FOOTER の値よりも優先されます。
TO *レポートファイル名* HTML 省略可能	このビューから作成されるレポートのファイル名と種類。 HTML キーワードを使用して、このビューから生成されたレポートを HTML ファイル(.htm) として保存します。デフォルトでは、生成されるレポートは ASCII テキスト ファイルとして出力されます。
OK 省略可能	アクションを確認せずに、項目を削除または上書きします。

例

ビューを作成する

次の例では、Ar テーブルを開き、そのテーブルレイアウトの全フィールドを含む AR_Report というビューを作成します。

```
OPEN Ar
DEFINE VIEW AR_Report HEADER "AR_Report" ALL OK
```

DELETE コマンド

Analytics プロジェクト項目、テーブル レイアウトのフィールド、変数、1 つ以上のテーブル履歴エントリ、テーブル間の関係、または Windows フォルダーのファイルを削除します。ビューの列も削除します。

構文

目的	構文
Analytics プロジェクト項目を削除するには	DELETE *項目の種類 項目名* <OK>
テーブル レイアウトからフィールドを削除するには	DELETE *フィールド名* <OK>
ビューから列を削除するには	DELETE COLUMN *ビュー名 フィールド名* <ALL> <OK>
1 つまたはすべての変数を削除するには	DELETE {*変数名*|ALL} <OK>
現在の Analytics テーブルの履歴を削除するには	DELETE HISTORY *<保持する履歴エントリ数>* <OK>
2 つのテーブル間の関係を削除するには	DELETE RELATION *<子テーブル名|リレーション名>* <OK>
ファイルを削除するには	DELETE *ファイル名* <OK>
開いているテーブルからすべてのレコード ノートと自動生成された **RecordNote** フィールドを削除するには	DELETE NOTES <OK>

パラメーター

名前	説明
項目の種類 項目名	削除対象となる項目の種類と名前。 次の項目タイプのいずれかを指定します。

名前	説明
	◦ **FOLDER**：-項目名に指定するプロジェクト フォルダーとその内容 ◦ **FORMAT**：-項目名に指定するテーブル レイアウトと、それに関連するビュー、インデックスおよび関連付け 関連するテーブルのその他のテーブル レイアウトは保持されます。 [**オプション**]ダイアログ ボックス([**ツール > オプション**]) の [**テーブル**]タブで、 [**テーブルと一緒にデータ ファイルを削除する**]オプションがオンになっていなければ、テーブル レイアウトに関連付けられたデータ ファイル(.fil) は削除されません。 このオプションのオン/オフは、スクリプトまたはコマンド ラインで SET DELETE_FILE {ON|OFF} コマンドを使用して切り替えることもできます。詳細については、"SET コマンド" ページ 404を参照してください。 **注意** [**テーブルと一緒にデータ ファイルを削除する**]オプションをオンにする際には慎重に行ってください。テーブル レイアウトと共に元のデータ ファイルが削除される場合もあります。 データ ファイルは完全に削除されます。Windows のごみ箱には送られません。 ◦ **REPORT** -指定されたビュー ビューは、現在アクティブである場合には削除できません。 ◦ **COLUMN** - 項目名に指定する列 ◦ **SCRIPT(または BATCH)** - 項目名に指定するスクリプト ◦ **WORKSPACE** - 項目名に指定するワークスペース ◦ **INDEX** - 項目名に指定するインデックス ◦ **NOTES** - 開いているテーブル内のすべてのレコード ノートと、テーブル レイアウト内の RecordNote フィールド
フィールド名 | ALL	**フィールドの削除** 現在の Analytics テーブル レイアウトから削除するフィールドの名前 フィールドは、現在のビューに含まれている場合でもテーブル レイアウトから削除できます。 **メモ** 演算フィールドから参照されているフィールドは、まずその演算フィールドを削除してからでないと削除できません。 **列の削除** 指定したビューから削除する列の名前 **メモ** 列表示名ではなく、物理フィールド名を使用します。 ◦ **ALL included(すべて含める)** - は、ビューにおける指定列のすべての出現箇所を削除します。 ◦ **ALL omitted(すべて省略)** -は、ビューにおける指定列の最初(一番左) の出現箇所を削除します。
ビュー名	列を削除するビューの名前。
変数名 | ALL	削除する変数の名前。すべての変数を削除する場合は、ALL を使用します。

名前	説明
	ALL を指定すると、プロジェクトから以下のタイプの変数の出現がすべて削除されます。 ◦ システム変数 ◦ 一時的なユーザー定義変数 ◦ 永続的なユーザー定義変数 **メモ** 演算フィールドから参照されている変数は、まずその演算フィールドを削除してからでないと削除できません。
HISTORY *保持する履歴エントリ数*	テーブルの全履歴エントリのうち、*保持する履歴エントリ数*で指定した数の直近のエントリより前の履歴エントリがすべて削除されます。 すべてのエントリを削除する場合は、*保持する履歴エントリ数*を省略します。
RELATION *子テーブル名* | *関連付け名*	依存関係がなく、アクティブなビューやアクティブな演算フィールドで参照される関連フィールドもない関係がすべて削除されます。 削除する関連付けを指定するには、以下のオプションを使用します。 ◦ ***子テーブル名*** - 関連付けに固有の名前が付けられていない場合に(関連付けを作成したときのデフォルト名を) 使用します。 ◦ ***関連付け名*** - 関連付けが固有の名前を付けて作成されている場合に使用します。そうでない場合は、*子テーブル名*を使用します。 どちらのオプションも使用しない場合は、最後に定義された関連付けが削除されます。
ファイル名	削除する物理ファイルの名前。 削除するファイルへの絶対パスまたは相対パスを指定することができます。パスにスペースが含まれている場合は、パスを二重引用符で囲みます。
OK 省略可能	確認ダイアログボックスを表示せずに項目を削除します。

例

日付フィールドの削除

次の例では、Ar テーブルに関連付けられているテーブルレイアウトから Date フィールドが削除されます。

```
OPEN Ar
DELETE Date
```

ビューから複数の列を削除する

Ar テーブルに関連付けられている AR_Report ビューから 2 つの列を削除するとします。次の例では、スクリプト実行時に確認プロンプトが表示されないよう、2 つの DELETE コマンドのどちらでも OK が指定されています。

```
OPEN Ar
DELETE COLUMN AR_Report Date OK
DELETE COLUMN AR_Report Invoice_Date OK
```

DIALOG コマンド

1 つ以上のスクリプト入力値をインタラクティブにユーザーに確認するカスタム ダイアログ ボックスを作成します。各入力値は名前付き変数に格納されます。

> **メモ**
>
> DIALOG コマンドを使用してパスワードを入力することは安全ではありません。代わりに "PASSWORD コマンド" ページ 345 を使用してください。
>
> DIALOG コマンドは、ACL Server 分析ではサポートされません。
>
> "ACCEPT コマンド" ページ 52では基本的なインタラクティブ ダイアログボックスを作成できます。

> **ヒント**
>
> カスタム ダイアログボックスを作成する最も簡単な方法は、**ダイアログ ビルダー**を使用することです。詳細については、カスタム ダイアログボックスの作成を参照してください。

構文

DIALOG (DIALOG TITLE *タイトルテキスト* WIDTH *ピクセル数* HEIGHT *ピクセル数*) (BUTTONSET TITLE "&OK;&Cancel" AT *x_pos y_pos* <WIDTH *ピクセル数*> <HEIGHT *ピクセル数*> DEFAULT *項目番号* <HORZ>) <[ラベル構文]|[テキストボックス構文]|[チェックボックス構文]|[ラジオボタン構文]|[ドロップダウンリスト構文]| [[プロジェクト項目リスト構文]> <...*n*>

ラベル構文 ::=
(TEXT TITLE *タイトルテキスト* AT *X座標 Y座標* <WIDTH *ピクセル数*> <HEIGHT *ピクセル数*> <CENTER|RIGHT>)

テキストボックス構文 ::=
(EDIT TO *変数名* AT *X座標 Y座標* <WIDTH *ピクセル数*> <HEIGHT *ピクセル数*> <DEFAULT *文字列*>)

チェックボックス構文 ::=
(CHECKBOX TITLE *タイトルテキスト* TO *変数名* AT *X座標 Y座標* <WIDTH *ピクセル数*> <HEIGHT *ピクセル数*> <CHECKED>)

ラジオボタン構文 ::=
(RADIOBUTTON TITLE *値リスト* TO *変数名* AT *X座標 Y座標* <WIDTH *ピクセル数*> <HEIGHT *ピクセル数*> <DEFAULT *項目番号*> <HORZ>)

ドロップダウンリスト構文 ::=
(DROPDOWN TITLE *値リスト* TO *変数名* AT *X座標 Y座標* <WIDTH *ピクセル数*> <HEIGHT *ピクセル数*> <DEFAULT *項目番号*>)

プロジェクト項目一覧構文 ::=
(ITEM TITLE *プロジェクト項目の種類* TO *変数名* AT *X座標 Y座標* <WIDTH *ピクセル数*> <HEIGHT *ピクセル数*> <DEFAULT *文字列*>)

パラメーター

一般パラメーター

名前	説明
DIALOG TITLE *タイトルテキスト*	メインダイアログボックスとダイアログボックスタイトルを作成します。 *タイトルテキスト*は引用符で囲んだ文字列として指定する必要があります。
BUTTONSET TITLE "&OK;&キャンセル"	ダイアログボックスの［**OK**］ボタンおよび［**キャンセル**］ボタンのラベル。 通常、この値を変更する必要はありませんが、変更する場合は、必ず肯定的な値が否定的な値の前に来るようにしてください。例：`"&はい;&いいえ"`
WIDTH *ピクセル数*	個々のコントロールの幅、またはダイアログボックスの幅（DIALOG コントロールに対して指定された場合）。 値はピクセル数で指定します。値がコントロールに対して指定されない場合は、幅はコントロールに含まれる最も長い値に基づいて計算されます。
HEIGHT *ピクセル数*	個々のコントロールの高さ、または DIALOG コントロールに対して指定された場合はダイアログボックスの高さ。 値はピクセル数で指定します。
AT *X座標 Y座標*	カスタムダイアログボックス上のコントロールの左上隅の場所。 ◦ *X座標*はダイアログボックスの左端からの水平距離（ピクセル）です。 ◦ *Y座標*はダイアログボックスの上端からの垂直距離（ピクセル）です。
DEFAULT *項目番号*	BUTTONSET 値で、デフォルトとして選択する項目に相当する数値。 たとえば、BUTTONSET 値が "&OK;&キャンセル" の場合、`DEFAULT 1` を指定し、デフォルトで OK を選択します。
HORZ 省略可能	BUTTONSET コントロールで値が左右に並べて表示されます。値はデフォルトで上下に並べて表示されます。

> **メモ**
>
> ほとんどのコントロールタイプでは、DIALOG コマンドは、ユーザー入力を保存するための変数を作成します。代入変数に使用される変数の名前に、é のような英語以外の文字は使用しないでください。変数名に英語以外の文字が含まれていると、スクリプト エラーになります。
>
> デフォルトでは、DIALOG 変数の一部は、文字変数として作成されます。文字変数を使用して、数値または日付時刻値を保存する場合は、スクリプトの後の処理で、変数を必要なデータ型に変換する必要があります。詳細については、"入力データ型" ページ 151を参照してください。

ラベル パラメーター

名前	説明
TEXT	テキスト ラベルを作成して、特定、通知、または指示します。
TITLE *タイトルテキスト*	コントロール ラベル。 *タイトルテキスト*は引用符で囲んだ文字列として指定する必要があります。
CENTER | RIGHT 省略可能	コントロール内のテキストの配置。 CENTER または RIGHT を省略する場合、左寄せがデフォルトで使用されます。

テキスト ボックス パラメーター

名前	説明
EDIT	ユーザー入力用のテキスト ボックスを作成します。
TO *変数名*	ユーザーによって指定された入力を格納する文字変数の名前。 変数が既に存在する場合は、指定された値が割り当てられます。変数が存在しない場合、変数が作成され、指定された値が割り当てられます。
DEFAULT *文字列* 省略可能	コントロールに表示するデフォルトのテキスト文字列。 *文字列*は引用符で囲んだ文字列として指定する必要があります。

チェックボックス パラメーター

名前	説明
CHECKBOX	チェックボックスを作成して、ユーザーへのオプションを表示します。
TITLE *タイトルテキスト*	コントロール ラベル。 *タイトルテキスト*は引用符で囲んだ文字列として指定する必要があります。

名前	説明
TO *変数名*	ユーザーによって指定された True または False 値を格納する論理変数の名前。 変数が既に存在する場合は、指定された値が割り当てられます。変数が存在しない場合、変数が作成され、指定された値が割り当てられます。
CHECKED 省略可能	コントロールがデフォルトでオンにされるように設定します。

ラジオボタン パラメーター

名前	説明
RADIOBUTTON	ラジオ ボタンを作成して、相互に排他的なオプションをユーザーに表示します
TITLE *値リスト*	コントロールに表示される値のリスト。 値は引用符で囲んだ文字列として指定する必要があります。各値はセミコロン(;) で区切ります。
TO *変数名*	ユーザーによって選択されたラジオボタン値の数値位置を格納する数値変数の名前。 変数が既に存在する場合は、指定された値が割り当てられます。変数が存在しない場合、変数が作成され、指定された値が割り当てられます。
DEFAULT *項目番号* 省略可能	項目リスト中で、デフォルトとして選択する項目に相当する数値。 たとえば、*値リスト*は "赤;緑;青" であるとします。フォルトで " 緑" が選択されるようにするには、DEFAULT 2 を指定します。
HORZ 省略可能	コントロールで値が左右に並べて表示されます。値はデフォルトで上下に並べて表示されます。

ドロップダウン リスト パラメーター

名前	説明
DROPDOWN	ドロップダウン リストを作成して、ユーザーへのオプションのリストを表示します
TITLE *値リスト*	コントロールに表示される値のリスト。 値は引用符で囲んだ文字列として指定する必要があります。各値はセミコロン(;) で区切ります。
TO *変数名*	ユーザーによって選択されたドロップダウン リスト値を格納する文字変数の名前。 変数が既に存在する場合は、指定された値が割り当てられます。変数が存在しない場合、変数が作成され、指定された値が割り当てられます。

名前	説明
DEFAULT *項目番号* 省略可能	項目リスト中で、デフォルトとして選択する項目に相当する数値。 たとえば、*値リスト*は"赤;緑;青"であるとします。ドロップダウンリストが表示されるときデフォルトで"緑"が選択されるようにするには、DEFAULT 2 を指定します。

プロジェクト項目一覧

名前	説明
ITEM	プロンプト項目リストを作成して、フィールドなどの Analytics プロジェクト項目のリストをユーザーに表示します。
TITLE *プロジェクト項目カテゴリ*	プロジェクト項目コントロールに含めるプロジェクト項目のカテゴリ。 1 つ以上のカテゴリを指定できます。ユーザーはプロジェクト項目リストから 1 つの値を選ぶことができます。 *project_item_category* を引用符で囲みます。カテゴリ間にはスペースまたは句読点を入れません。 カテゴリを指定するために使用するコードについては、"プロジェクト項目カテゴリーのコード"次のページを参照してください。 **メモ** 特に理由がない場合は、同じ ITEM コントロールに異なるカテゴリを混在させないでください。たとえば、テーブルとフィールドを混在させないでください。結果のプロジェクト項目一覧がユーザーにとって混乱するものになる可能性があります。
TO *変数名*	ユーザーによって選択されたプロジェクト項目の名前を格納する文字変数の名前。 変数が既に存在する場合は、指定された値が割り当てられます。変数が存在しない場合、変数が作成され、指定された値が割り当てられます。
DEFAULT *文字列* 省略可能	デフォルトとして選択するプロジェクト項目の正確な名前。 *文字列*は引用符で囲んだ文字列として指定する必要があります。

例

テーブルとスクリプトを選択するようユーザーに求める

スクリプトでは、解析の実行に使用する Analytics のテーブルおよびスクリプトを選択するようユーザーに求める必要があるとします。

`ACL_Demo.acl` プロジェクトの **Metaphor_Inventory_2012** テーブルを Analytics テーブルのデフォルトとして選択されるように指定しています(4 行目)。ただし、ユーザーはプロジェクト内で任意のテーブルを選択することもできます。

実行するスクリプトも、Analytics プロジェクト内でのスクリプトのリストから選択する必要があります。

```
DIALOG (DIALOG TITLE "在庫分析" WIDTH 500 HEIGHT 200 ) (BUTTONSET TITLE "OK(&O);キャンセル(&C)" AT 370 12 DEFAULT 1 ) (TEXT TITLE "分析するAnalyticsプロジェクトを選択してください。" AT 50 16 ) (TEXT TITLE "テーブル:" AT 50 50 ) (ITEM TITLE "f" TO "v_table" AT 50 70 DEFAULT "Metaphor_Inventory_2012" ) (TEXT TITLE "スクリプト:" AT 230 50 ) (ITEM TITLE "b" TO "v_script" AT 230 70 )
```

備考

> **メモ**
> このコマンドの動作の詳細については、Analytics のヘルプを参照してください。

インタラクティブ

インタラクティブ スクリプトを作成するには、DIALOG コマンドを使用します。DIALOG コマンドが処理されるときにはスクリプトが中断し、Analytics が次の処理で使用する入力をユーザーに促すダイアログ ボックスが表示されます。

項目ごとに個別のダイアログ ボックスを作成して、一度に 1 項目について入力を求めることもできますが、1 つのダイアログ ボックスで複数項目について入力を求めることもできます。

ACCEPT と DIALOG

ACCEPT コマンドでは、次の 1 つ以上のタイプのコントロールを使用できる基本的なインタラクティブ ダイアログ ボックスを作成できます。

- テキスト ボックス
- プロジェクト項目一覧

基本的な対話性については、ACCEPT で十分であると考えられます。詳細については、"ACCEPT コマンド" ページ 52を参照してください。

プロジェクト項目カテゴリーのコード

次のコードを使用して、プロジェクト項目リストで表示するプロジェクト項目のカテゴリーを指定します。

プロジェクト カテゴリー

コード	カテゴリー
f	テーブル
十億	スクリプト
i	インデックス

コード	カテゴリー
r	ビューとレポート
w	ワークスペース

フィールド カテゴリー

コード	カテゴリー
C	文字フィールド
N	数値フィールド
D	日付時刻フィールド
L	論理フィールド

変数カテゴリ

コード	カテゴリー
c	文字変数
n	数値変数
d	日付時刻変数
l	論理変数

入力データ型

DIALOG コマンドのコントロールの一部は、文字変数にユーザー入力を保存します。数値や日付時刻値として保存するには、VALUE( ) 関数やCTOD( ) 関数を使用します。文字変数の値がそれぞれ数値や日付時刻値に変換されます。

```
SET FILTER TO BETWEEN(%v_date_field%, CTOD(%v_start_date%), CTOD(%v_end_date%))
```

この例では、フィルターの開始日付および終了日付は文字値として保存されています。これらの日付は、日付時刻データ型を使用する日付フィールドで使用するには、日付型の値に変換する必要があります。

変数名をパーセント記号 (%) で囲むと、変数名の変数によって含まれる文字値を代入します。CTOD( ) 関数は文字値を日付値に変換します。

DIALOG コマンドの位置

可能な限り、DIALOG コマンドはすべてスクリプトの先頭部分に配置することをお勧めします。最初にすべての情報入力を要求し、必要な情報が入力されていれば、その後スクリプトを円滑に実行できます。

> **メモ**
>
> GROUP コマンド内では DIALOG コマンド コマンドを使用できません。

DIRECTORY コマンド

指定されたディレクトリ内のファイルとフォルダーの一覧を生成します。

構文

DIRECTORY <*ファイルスペック*> <SUPPRESS> <SUBDIRECTORY> <APPEND> <TO *テーブル名* | *ファイル名*>

パラメーター

名前	説明
ファイルスペック 省略可能	含まれている情報を一覧表示および表示する Windows フォルダーまたはファイル。 アスタリスクのワイルドカード (*) を使用して、特定の拡張子を持つすべてのファイルや、特定の文字列で始まるすべてのファイル、またはフォルダー内のすべてのファイルを選択できます。例: ◦ `*.fil` - ファイル名の拡張子が .fil であるすべてのファイル(Analytics データファイル) が一覧表示されます。 ◦ `Inv*.*` - ファイル拡張子が何であれ、名前が "Inv" で始まるすべてのファイルが一覧表示されます。 ◦ `Results\*` または `Results\*.*` - Results フォルダー内のすべてのファイルが一覧表示されます。 一覧表示されるファイルを特定のフォルダーに制限するには、Analytics プロジェクト フォルダーへの相対パスを指定するか、完全パスを指定できます。例: ◦ `Results\*.*` - Analytics プロジェクト フォルダーの Results サブフォルダーの内容が表示されます。 ◦ `C:\ACL Data\Results\*.*` - 指定したフォルダーの内容が表示されます。 **メモ** ワイルドカード文字は、指定されたファイル パスの中間レベルで使用できません。上記のように、パスの最終レベルでのみ使用できます。 スペースを含んでいるパスまたはファイル名は、二重引用符で囲む必要があります。 *ファイルスペック*を使用する場合は、ほかのパラメーターの前に配置する必要があります。*ファイルスペック*がほかの位置に表示されると、DIRECTORY コマンドが処理されず、エラーが発生します。 *ファイルスペック*を省略した場合は、Analytics プロジェクトが入っているフォルダー内のすべてのファイルが一覧表示されます。*ファイルスペック*を省略したら、ほかのパラメーターはどれも使用できません。
SUPPRESS 省略可能	出力にパス情報が表示されずに、ファイル名とプロパティのみが表示されます。

名前	説明
SUBDIRECTORY 省略可能	サブフォルダーの内容が、ディレクトリの一覧に含まれます。 たとえば、*ファイルスペック*が `Results\*.fil` の場合は、Results フォルダーと、Results フォルダー下のすべてのサブフォルダーで、`.fil` ファイルが検索されます。 サブフォルダーで検索が行われるとき、一覧表示する必要のあるサブフォルダーとファイルの数によっては、SUBDIRECTORY を使用することで遅延が生じる可能性があります。Analytics により、コマンド実行の進捗を示すダイアログボックスが表示されます。
APPEND 省略可能	コマンドの出力を既存ファイルに上書きしないで、そのファイルの末尾に追加します。 **メモ** コマンドの出力と既存のファイルの構造が同一であるようにする必要があります。 • 同じフィールド • 同じフィールド順序 • 一致するフィールドが同じ長さ • 一致するフィールドが同じデータ型 出力は、既存ファイルとの間でファイル構造が違っている場合でも、Analytics によって既存ファイルに追加されます。出力と既存のファイルの構造が一致しない場合は、データが混在、不足、不正確になります。
TO *テーブル名 | ファイル名* 省略可能	コマンドの結果を送信する場所： ◦ ***テーブル名*** -は、結果の保存先となる Analytics テーブルのことです。 *テーブル名*には、ファイル拡張子 .FIL を付けた文字列を引用符で囲んで指定する必要があります。例：`TO "Output.FIL"` デフォルトでは、テーブルデータファイル(.fil) は、Analytics プロジェクトが入っているフォルダーに保存されます。 既存の異なるフォルダーにデータファイルを保存するには、絶対または相対ファイルパスを指定します。 • `TO "C:\Output.FIL"` • `TO "Results\Output.FIL"` **メモ** テーブル名は 64 文字の英数字(.FIL 拡張子を含まない) に制限されています。名前にはアンダースコア文字(_) を使用できますが、他の特殊文字やスペースは使用できません。名前の先頭を数字にすることはできません。 ◦ ***ファイル名*** -は結果の保存先となるファイルです。 *ファイル名*には、適切なファイル拡張子を付けた文字列を引用符で囲んで指定します。例：`TO "Output.TXT"` デフォルトでは、テーブルファイルは、Analytics プロジェクトが入っているフォルダーに保存されます。 既存の異なるフォルダーにファイルを保存するには、絶対または相対ファイルパスを指定します。 • `TO "C:\Output.TXT"` • `TO "Results\Output.TXT"` TO を省略した場合は、ディレクトリの一覧は Analytics の表示領域に表示されます。

例

ファイルの様々な一覧表示方法

ファイルの一覧表示機能は、臨時の調査に有効であるほか、スクリプトへの組み込み対象として有効なものです。

DIRECTORY コマンドを使ってファイルを一覧表示する様々な方法を次に示します。

すべてのファイルを一覧表示

Analytics プロジェクトが入っているフォルダー内のすべてのファイルをリストアップする

```
DIRECTORY
```

特定のタイプのすべてのファイルを一覧表示

Analytics プロジェクトが入っているフォルダー内のすべての .fil ファイル(Analytics データファイル) をリストアップする

```
DIRECTORY *.fil
```

ワイルドカードを使用してファイルを一覧表示

Analytics プロジェクトが入っているフォルダー内の、"Inv" で始まるすべてのファイル名をリストアップする

```
DIRECTORY Inv*.*
```

Analytics プロジェクト フォルダーに相対的なサブフォルダーのすべてのファイルを一覧表示

Analytics プロジェクトが入っているフォルダー内の、`Results` サブフォルダーにあるすべてのファイルをリストアップする

```
DIRECTORY "Results\*"
```

指定したフォルダー内のすべてのファイルをリストアップする

`Results` サブフォルダーのすべてのファイルを一覧表示

```
DIRECTORY "C:\ACL Data\Results\*"
```

特定の場所の特定のタイプのすべてのファイルを一覧表示

指定したフォルダーとそのすべてのサブフォルダー内にある、すべての .fil ファイル(Analytics データファイル) を一覧

表示する

```
DIRECTORY "C:\ACL Data\Results\*.fil" SUBDIRECTORY
```

指定されたフォルダーのすべてのファイルを一覧表示し、リストを Analytics テーブルに出力

`Results`フォルダー内のすべてのファイルを一覧表示し、そのリストを、Analytics プロジェクトが入っているフォルダー内の Analytics テーブルに出力する

```
DIRECTORY "C:\ACL Data\Results\*" TO Results_Folder_Contents.fil
```

新しいテーブル`Results_Folder_Contents`が開いているプロジェクトに追加されます。

1 つのフォルダーのすべてのファイルを一覧表示し、リストを別のフォルダーの Analytics テーブルに出力

`ACL Data\Results`フォルダー内のすべてのファイルを一覧表示し、そのリストを、`GL Audit 2014\Results`フォルダーの Analytics テーブルに出力する

```
DIRECTORY "C:\ACL Data\Results\*" TO "C:\ACL Projects\GL Audit 2014\Results\Results_Folder_Contents.fil"
```

新しいテーブル`Results_Folder_Contents`が開いているプロジェクトに追加されます。関連するデータファイル(`Results_Folder_Contents.fil`)が指定した出力フォルダーに作成されます。このフォルダーは、Analytics プロジェクトが入っているフォルダーであってもなくてもかまいません。

備考

DIRECTORY によって表示されるプロパティ

DIRECTORY コマンドは、Windows の DIR コマンドに似ています。フォルダー内のファイルとサブフォルダーの一覧を表示することに加え、DIRECTORY コマンドでは、次のファイルプロパティおよびフォルダープロパティも表示します。

◦ ファイル サイズ ◦ 属性	◦ 作成日 ◦ 作成時刻	◦ アクセス日 ◦ アクセス時刻	◦ 更新日 ◦ 更新時刻 ◦ 指定された条件に一致するファイルおよびフォルダーの総数

スクリプト内での DIRECTORY の使用

スクリプト内で DIRECTORY コマンドを使用すると、ファイルシステムを調べることができます。たとえば、DIRECTORY をほかのコマンドと連携させて、ファイルの有無を検出したり、ファイルのサイズを調べたり、ほかのファイルプロパティに基づいて判断を下したりすることができます。

DIRECTORY の結果を出力する

コマンドラインからこのコマンドを実行する場合は、ディレクトリの一覧を画面に表示するか、一覧を Analytics テーブルまたは .txt ファイルに保存することができます。

DIRECTORY の結果を保存したテーブルを開く方法

DIRECTORY コマンドには OPEN パラメーターが含まれていません。スクリプトでコマンドを使用し、結果を Analytics テーブルに出力する場合、その結果テーブルを開きたいときは、DIRECTORY コマンドの後に OPEN コマンドを続けてください。例：

```
DIRECTORY "C:\ACL Data\Results\*" TO Results_Folder_Contents.fil
OPEN Results_Folder_Contents
```

DISPLAY コマンド

指定されたAnalytics 項目タイプの情報を表示します。式の結果、または関数の出力も表示できます。

構文とパラメーター

構文	目的
DISPLAY	フィールド定義を表示し、現在アクティブな Analytics テーブルの関連付けられた子テーブルを表示する
DISPLAY OPEN	開いている Analytics のテーブルおよびプロジェクト ファイルの一覧を表示する ◦ **Analytics のテーブル** -は、テーブル レイアウトの名前ではなく、ソース データ ファイルの名前として表示されます。 ◦ **複数テーブル モード** - PRIMARY として特定されたソース データ ファイルが、現在アクティブなテーブルに関連付けられます。 ◦ **関連テーブル** - 親テーブルが開いている場合、［ビュー］タブで子テーブルが開いていなくても、親と子の両方のテーブルのソース データ ファイルが表示されます。
DISPLAY {<PRIMARY>|SECONDARY}	主テーブルまたは副テーブルの名前およびテーブル レイアウト情報を表示する ◦ **PRIMARY(またはキーワードを指定しない場合)** - 現在アクティブなテーブルの情報が表示されます。 ◦ **SECONDARY** - 副テーブルの情報が表示されます。 複数テーブル モードでは、SECONDARY は現在アクティブなテーブルに関連付けられている副テーブルを参照します。 以下の情報が表示されます。 ◦ テーブル レイアウト名 ◦ ソース データ ファイル名 ◦ 現在アクティブなテーブルとその他のテーブルとの間にあるあらゆる関連付け ◦ テーブル レイアウト内のフィールド定義情報
DISPLAY HISTORY	現在のアクティブな Analytics テーブルのテーブル履歴を表示する **メモ** テーブルには、関連するテーブル履歴がある場合もない場合もあります。
DISPLAY RELATION	現在アクティブな Analytics テーブルの関連付け情報を表示する ◦ すべての子テーブルの名前 ◦ キー フィールド名 ◦ インデックス名
DISPLAY {*変数名*|VARIABLES}	1 つの変数あるいはすべての変数の値を表示する ◦ ***変数名*** - 値の表示対象となる単独の変数の名前

構文	目的
	◦ **VARIABLES** - すべてのシステム変数およびユーザー定義変数の値が表示されます。また、変数を格納するために利用できる残りのメモリ量も表示されます。
DISPLAY VERSION	インストールされた Analytics のバージョンに関する情報を次の書式で表示する ◦ **バージョン** - *メジャーバージョン番号.マイナーバージョン番号* ◦ **パッチ** - *パッチ番号* ◦ **種類** - Analytics の非 Unicode 版(000) または Unicode 版(001) ◦ **Build(ビルド)** - *ソフトウェアのビルド番号*
DISPLAY {DATE|TIME}	オペレーティングシステムの現在の日付と時刻を表示する **DATE | TIME** - いずれかのキーワードを指定します。どちらのキーワードを指定しても上記の機能が実行されます。
DISPLAY {FREE|SPACE}	Analytics で使用できる物理メモリ(RAM) の量を表示する 表示されるこの数値には、変数用に確保されたメモリは含まれません。デフォルトで、Analytics は変数を格納するために 60 KB の物理メモリを予約しますが、このサイズは必要に応じて自動的に増加します。 **FREE | SPACE** - いずれかのキーワードを指定します。どちらのキーワードを指定しても上記の機能が実行されます。
DISPLAY *式*	式の結果を表示する ***式*** - 結果の表示対象となる式
DISPLAY *関数*	関数の出力を表示する ***関数*** - 出力の表示対象となる関数

例

Analytics テーブルのレイアウトを表示する

テーブルレイアウトを表示する機能は、いろいろな場合に使用できます。たとえば、2 つ以上のテーブルを結合する場合や、フィールドの長さとデータ型を調べる場合に使用できます。

次の例では、Ap_Trans テーブルのレイアウトを表示しています。

```
OPEN AP_Trans
DISPLAY
```

DISPLAY コマンドにより行われる画面への出力は下記のとおりです。

> **メモ**
>
> Analytics コマンド ラインに直 接「DISPLAY」を入 力 する場 合 には、出 力 がすぐに表 示されます。
>
> これに対し、スクリプト内 で DISPLAY を実 行した場 合 には、コマンド ログ内 の該 当 する **DISPLAY** エントリをダブルクリックすれば出 力 が表 示されます。

画面への出力

関連

インデックス 'Vendor_on_Vendor_No' を使 用した 'Vendor' と 'Vendor_No' の関 連 付 け

ファイル

PRIMARY ファイルは 'Ap_Trans.fil'(書 式 'Ap_Trans') です。

レコード長 は 59 です。

フィールド

名前	種類	開始位置	長さ	小数点以下桁数	フィールドの説明
Vendor_No	ASCII	1	5		AS "業者;番号" WIDTH 7
Invoice_No	ASCII	6	15		AS "請求書;番号"
Invoice_Date	DATETIME	21	8		PICTURE "MM/DD/YY" AS "請求日" WIDTH 8
Invoice_ Amount	NUMERIC	29	12	2	PICTURE "(9,999,999.99)" AS "請求;金額" WIDTH 12
Prodno	ASCII	41	9		AS "製品番号"
Quantity	MICRO	50	4	0	SET PICTURE "(9,999,999.99)"
Unit_Cost	NUMERIC	54	6	2	PICTURE "(9,999,999)" AS "単価" SUPPRESS

Analytics プロジェクト内の全変数の値を表示する

DISPLAY VARIABLES は**ナビゲーター**の [**変数**] タブに表 示されるのと同じ情 報を出 力します。DISPLAY VARIABLES を使 用することの 1 つのメリットは、表 示された情 報をコピーして貼り付 けることができることです。

次 の例 は、2 つのユーザー定 義 変 数と 2 つのシステム変 数を作 成して、それらすべての変 数 の値を表 示しています。

```
ASSIGN v_table_name = "Ap_Trans"
ASSIGN v_field_name = "Invoice_Amount"
OPEN %v_table_name%
```

```
TOTAL FIELDS %v_field_name%
DISPLAY VARIABLES
```

DISPLAY コマンドにより行われる画面への出力は下記のとおりです。

> **メモ**
>
> Analytics コマンド ラインに直接「DISPLAY VARIABLES」を入力する場合には、出力がすぐに表示されます。
>
> これに対し、スクリプト内で DISPLAY VARIABLES を実行した場合には、コマンド ログ内の該当する **DISPLAY VARIABLES** エントリをダブルクリックすれば出力が表示されます。

画面への出力

名前	種類	値
TOTAL1	N	278,641.33
OUTPUTFOLDER	C	"/Tables/Accounts_Payable"
v_field_name	C	"Invoice_Amount"
v_table_name	C	"Ap_Trans"

式の結果を表示する

次の例により、選択したレコードにおける Sale_Price フィールドの値と Quantity_on_Hand フィールドの値を乗算した結果が表示されます。

```
DISPLAY Sale_Price * Quantity_on_Hand
```

関数の出力を表示する

次の例により、選択したレコードの Invoice_Date フィールドの日付以来経過した日数が表示されます。

```
DISPLAY AGE(Invoice_Date)
```

備考

コマンドの実行結果の場所

DISPLAY を Analytics のコマンド ラインから実行した場合 - 結果は画面に表示されます。

DISPLAY をスクリプト内で実行した場合 - 結果は Analytics コマンド ログに書き込まれます。結果を画面に表示するには、コマンド ログの当該エントリをダブルクリックします。

DO REPORT コマンド

指定されたAnalytics レポートを生成します。

構文

```
DO REPORT レポート名
```

パラメーター

名前	説明
レポート名	生成し、レポートとして出力するビューの名前。

例

デフォルトのビューを印刷する

AP_Trans テーブルを開いてデフォルトのビューを印刷するとします。

```
OPEN AP_Trans
DO REPORT デフォルト_ビュー
```

備考

コマンド ラインでの DO REPORT の実行とスクリプト内での DO REPORT の実行との比較

レポートを印刷するために使用される設定は、コマンドを実行する場所によって異なります。

- **コマンドラインから** - [**印刷**]ダイアログボックスが開き、ページを選択して、レポートの他のオプションを印刷および構成できます。
- **スクリプト内** -レポートのデフォルト設定を使用して直ちにレポートが印刷されます。

DO SCRIPT コマンド

ACL スクリプト内から第 2 のまたは外部スクリプトを実行します。

構文

DO <SCRIPT> *スクリプト名* {<IF *testテスト*>|<WHILE *テスト*>}

パラメーター

名前	説明
SCRIPT *スクリプト名*	実行対象となるスクリプトの名前。Analytics プロジェクトで副スクリプトを実行するか、.aclscript、.txt 、.bat などの拡張子があるテキスト ファイルに格納された外部スクリプトを実行できます。 外部スクリプトへのファイル パスを指定できます。スペースがある場合は、パスを引用符で囲む必要があります。 **メモ** 既に実行されているスクリプトを呼び出すことはできません。たとえば、ScriptA が ScriptB を呼び出す場合、ScriptB から ScriptA を呼び出すことはできません。ScriptA は、ScriptB が完了するのを待つ間も実行されています。
IF *テスト* 省略可能	スクリプトを実行するかどうかを判断するために 1 回評価される条件式。条件が True と評価された場合はスクリプトを実行します。そうでない場合は実行しません。 同じコマンド内で WHILE と一緒に使用することはできません。両方を使用する場合は、スクリプトが処理される際に WHILE が無視されます。コメントがログに入力されますが、スクリプトの実行は停止しません。
WHILE *テスト* 省略可能	スクリプトの実行後、再度スクリプトを実行するかどうかを判断するために評価される条件式。テストが True と評価された場合はスクリプトを再実行します。そうでない場合は実行しません。 **メモ** WHILE を使用する場合は、テストが最終的に false として評価されるようにしてください。そうしないと、スクリプトが無限ループになります。無限ループになった場合にスクリプト処理を取り消すには、**Esc** キーを押します。 同じコマンド内で IF と一緒に使用することはできません。両方を使用する場合は、スクリプトが処理される際に WHILE が無視されます。コメントがログに入力されますが、スクリプトの実行は停止しません。

例

入力が有効とされるまでサブスクリプトを繰り返し実行する

ダイアログ ボックスを使ってユーザー入 力を収 集 するサブスクリプトがあるとします。このサブスクリプトは以 下 を行います。

1. 必 要 な値 の入 力をユーザーに求 める
2. そのユーザー入 力を確 認 する
3. 入 力 値 が有 効とされたら、*v_validated* 変 数 を true に設 定 する

ユーザーが有 効 な入 力を行ったことを確 認 するには、DO SCRIPT を使 用 し、入 力 が有 効とされるまでスクリプトでこのコマンドが再 実 行 されるようにWHILE 条 件 を追 加 します。変 数 の値 が有 効になると、メインのスクリプトはその次 のコマンドに進 みます。

```
DO SCRIPT GetUserInput WHILE v_validated = F
```

共 有 ロケーションからのサブスクリプトの実 行

共 有 ロケーションで複 数 のユーティリティ サブスクリプトを管 理 することができます。分 析 時 にサブスクリプトのいずれかが必 要 な場 合 は、共 有 ロケーションへの絶 対 パスを使 ってサブスクリプトを参 照 します。

```
DO SCRIPT "C:\My utility scripts\GetUserInput.aclscript"  WHILE v_validated = F
```

備 考

関 連 コマンド

DO SCRIPT は、Analytics の旧 リリースで作 成 されたスクリプト内 で使 用 されている DO BATCH コマンドと同 等 です。

GROUP コマンド内 で DO SCRIPT コマンドを使 用 することはできません。

外 部 スクリプトの利 便 性

スクリプトを外 部 に保 存 すれば、Analytics スクリプト内 から呼 び出 せるので、同 じサブスクリプトを別 の Analytics スクリプトとプロジェクトで再 利 用 することができます。

スクリプトの単 一 のコピーを 1 つの場 所 に保 存 し、複 数 の場 所 で管 理 するのではなく 1 つの場 所 で更 新 できます。

DUMP コマンド

ファイルや現在のレコードの内容を 16 進、ASCII、および EBCDIC 文字エンコードで表示します。

> **メモ**
> このコマンドは、コマンド ラインでのみ入力できます。スクリプト内で使用することはできません。

構文

DUMP {RECORD|*ファイル名*} <SKIP *バイト*>} <COLUMN *バイト*> <HORIZONTAL>

パラメーター

名前	説明
RECORD	選択したレコードの内容を表示する *ファイル名*に値を指定しない場合は必須です。
ファイル名	表示するファイルの名前。 RECORD を指定しない場合は必須です。 **メモ** Analytics テーブルの文字エンコードを表示するには、ソース データ ファイルの名前とファイル拡張子を指定する必要があります。例：Ap_Trans.fil
SKIP *バイト数* 省略可能	ダンプを開始する前にバイパスするバイト数。デフォルトは 0 です。
COLUMN *バイト* 省略可能	出力における列の幅（バイト数）。 **メモ** *バイト*として指定された数値は、Analytics のレコードまたはテーブルに含まれるバイト数を示します。 出力におけるエンコードされた文字数は、ビューにおける文字数とは 1 対 1 で対応しない場合もあります。たとえば、数字 1 の 16 進エンコードは 31 です。 縦方向に表示した場合の各列のデフォルト数は 16 バイトであるのに対し、横方向に表示した場合の 1 列のデフォルト数は 64 バイトです。指定できる最大バイト数は 255 です。
HORIZONTAL 省略可能	文字エンコードが、縦方向である隣接するブロック（デフォルト）でなく、横方向である行に表示されます。

例

Inventory テーブルの文字エンコードを表示する

次の例により、Inventory テーブル内のデータの 16 進、ASCII、EBCDIC 文字エンコードが表示されます。出力は水平行で配置されます(16 進数エンコーディングは 2 行使用します) 。各行には、当該の Analytics テーブルからのデータが 97 バイト分、表現されています。

```
DUMP Inventory.fil COLUMN 97 HORIZONTAL
```

DUPLICATES コマンド

重複値または重複レコードがAnalyticsテーブル内に存在するかどうかを検出します。

構文

DUPLICATES <ON> {*キーフィールド* <D> <...*n*>|ALL} <OTHER *フィールド* <...*n*>|OTHER ALL> <UNFORMATTED> <TO {SCREEN|*テーブル名*|*ファイル名*|PRINT}> <APPEND> <IF *テスト*> <WHILE *テスト*> <FIRST *範囲*|NEXT *範囲*> <HEADER > <FOOTER *フッターテキスト*> <PRESORT> <OPEN> <LOCAL> <ISOLOCALE *ロケールコード*>

パラメーター

名前	説明
ON *キーフィールド* D <...*n*> | ALL	重複がないかどうかをテストする1つまたは複数のキーフィールド、あるいは式。 ◦ ***キーフィールド*** -には、指定された1つまたは複数のフィールドが使用されます。 複数のフィールドでテストする場合、レコードが重複として検出されるには、指定したすべてのフィールドの値が同一である必要があります。 キーフィールドを降順に並べ替えるDを含めます。デフォルトのソート順は昇順です。 ◦ **ALL** -では、テーブル内のすべてのフィールドが使用されます。 テーブル内のすべてのフィールドをテストする場合、レコードが重複として検出されるには、レコードが完全に同一である必要があります。 ALLでは昇順でしか並べ替えることができません。 **メモ** レコードの未定義部分はテストされません。
OTHER *フィールド* <...*n*> | OTHER ALL 省略可能	出力に含める1つ以上の追加フィールド。 ◦ ***フィールド <...n>*** - 指定した1つまたは複数のフィールドが含まれます。 ◦ **ALL** - キーフィールドとして指定しなかった、テーブル内のすべてのフィールドが含まれます。
UNFORMATTED 省略可能	結果をファイルに出力する場合、ページ見出しや改ページは除去されます。
TO SCREEN | *テーブル名* | *ファイル名* | PRINT 省略可能	コマンドの結果を送信する場所： ◦ **SCREEN** - はAnalyticsの表示領域に結果を表示します ◦ ***テーブル名*** -は、結果の保存先となるAnalyticsテーブルのことです。 *テーブル名*には、ファイル拡張子.FILを付けた文字列を引用符で囲んで指定する必要があります。例：`TO "Output.FIL"`

名前	説明
	デフォルトでは、テーブル データ ファイル(.fil) は、Analytics プロジェクトが入っているフォルダーに保存されます。 既存の異なるフォルダーにデータ ファイルを保存するには、絶対または相対ファイル パスを指定します。 • TO "C:\Output.FIL" • TO "Results\Output.FIL" **メモ** テーブル名は 64 文字の英数字(.FIL 拡張子を含まない) に制限されています。名前にはアンダースコア文字(_) を使用できますが、他の特殊文字やスペースは使用できません。名前の先頭を数字にすることはできません。 ◦ ***ファイル名*** -は結果の保存先となるファイルです。 *ファイル名*には、適切なファイル拡張子を付けた文字列を引用符で囲んで指定します。 例 : TO "Output.TXT" デフォルトでは、テーブル ファイルは、Analytics プロジェクトが入っているフォルダーに保存されます。 既存の異なるフォルダーにファイルを保存するには、絶対または相対ファイル パスを指定します。 • TO "C:\Output.TXT" • TO "Results\Output.TXT" ◦ **印刷** -- 通常使うプリンターに結果を送信します
APPEND 省略可能	コマンドの出力を既存ファイルに上書きしないで、そのファイルの末尾に追加します。 **メモ** コマンドの出力と既存のファイルの構造が同一であるようにする必要があります。 • 同じフィールド • 同じフィールド順序 • 一致するフィールドが同じ長さ • 一致するフィールドが同じデータ型 出力は、既存ファイルとの間でファイル構造が違っている場合でも、Analytics によって既存ファイルに追加されます。出力と既存のファイルの構造が一致しない場合は、データが混在、不足、不正確になります。
IF テスト 省略可能	各レコードを処理するために真である必要がある条件式。コマンドは、その条件を満たすレコードに対してのみ実行されます。 **メモ** IF パラメーターは、任意の範囲パラメーター(WHILE、FIRST、NEXT) が適用された後に、テーブルに残るレコードに対してのみ評価されます。
WHILE テスト 省略可能	各レコードを処理するために真である必要がある条件式。条件が false と評価するか、テーブルの最後に達したら、コマンドは実行を中止します。 **メモ** WHILE を FIRST または NEXT とともに使用する場合は、1 つの制限に達するとすぐに、レコードの処理が停止します。
FIRST *範囲* | NEXT *範囲*	処理するレコード数 :

名前	説明
省略可能	◦ **FIRST** - 指定されたレコード数に達するまで、最初のレコードから処理を開始します ◦ **NEXT** - 指定されたレコード数に達するまで、現在選択されているレコードから処理を開始します *範囲*は処理するレコード数を指定します。 FIRST とNEXT を省略すると、すべてのレコードがデフォルトで処理されます。
HEADER *ヘッダーテキスト* 省略可能	レポートの各ページの最上部に挿入されるテキスト。 *ヘッダーテキスト*は引用符で囲んだ文字列として指定する必要があります。この値は、Analytics のシステム変数である HEADER の値よりも優先されます。
FOOTER *フッターテキスト* 省略可能	レポートの各ページの最下部に挿入されるテキスト。 *フッターテキスト*は引用符で囲んだ文字列として指定する必要があります。この値は、Analytics のシステム変数である FOOTER の値よりも優先されます。
PRESORT 省略可能	コマンドを実行する前にキー フィールドでテーブルを並べ替えます。 **メモ** GROUP コマンドの内部では PRESORT を使用することができません。
OPEN 省略可能	コマンドを実行した後、コマンドによって作成されたテーブルを開きます。コマンドが出力テーブルを作成する場合にのみ有効です。
LOCAL 省略可能	Analytics プロジェクトと同じ場所に出力ファイルを保存します。 **メモ** Analytics テーブルである出力ファイルを含むサーバー テーブルに対してコマンドを実行するときにのみ適用されます。
ISOLOCALE *ロケールコード* 省略可能	**メモ** Analytics の Unicode 版にのみ適用されます。 システム ロケールは「*言語-国*」の形式で入力します。たとえば、カナダ フランス語はコード「fr_ca」を入力します。 次のコードを使用します。 ◦ **言語** - ISO 639 標準言語コード ◦ **国** - ISO 3166 標準国コード 国コードを指定しない場合は、言語のデフォルト国が使用されます。 ISOLOCALE を使用しない場合は、デフォルト システム ロケールが使用されます。

Analytics の出力変数

名前	含む
GAPDUP*n*	コマンドによって確認されたギャップ、重複、またはあいまい重複グループの合計数。

例

1 つのフィールドの重複値のテスト

次の例：

- **Invoice_Number** フィールドの重複値のテスト
- 重複する請求書番号を含むレコードを新しい Analytics テーブルに出力する

```
DUPLICATES ON Invoice_Number OTHER Vendor_Number Invoice_Date Invoice_Amount
PRESORT TO "Duplicate_Invoices.FIL"
```

2 つ以上のフィールドの組み合わせの重複値のテスト

次の例：

- **Invoice_Number** および **Vendor_Number** フィールドの値の重複した組み合わせのテスト
- 同じ請求書番号と同じ業者番号を含むレコードを新しい Analytics テーブルに出力する

このテストと前のテストの間の違いは、2 つの異なる業者番号からの同じ請求書番号が誤検出として報告されないことです。

```
DUPLICATES ON Invoice_Number Vendor_Number OTHER Invoice_Date Invoice_Amount
PRESORT TO "Duplicate_Invoices.FIL"
```

重複するレコードを検索する

下記の例では以下が行われます。

- Inventory テーブルのすべてのフィールドに重複値がないかどうかをテストする
- 完全に同じレコードを新しい Analytics テーブルに出力する

```
DUPLICATES ON ProdNum ProdClass Location ProdDesc ProdStatus UnitCost CostDate
SalePrice PriceDate PRESORT TO "Duplicate_Inventory_Items.FIL"
```

この構文は、ALL を使用すれば次のように簡素化できます。

```
DUPLICATES ON ALL PRESORT TO "Duplicate_Inventory_Items.FIL"
```

ESCAPE コマンド

Analytics を終了することなく、処理中のスクリプト、あるいはすべてのスクリプトを終了します。

構文

```
ESCAPE <ALL> <IF テスト>
```

パラメーター

名前	説明
ALL 省略可能	アクティブなすべてのスクリプトが強制終了されます。このパラメーターを省略した場合は、現在のスクリプトが強制終了されます。
IF テスト 省略可能	コマンドが実行されるのに True として評価される必要のあるテスト。テストが False と評価された場合、コマンドは実行されません。

例

スクリプトを条件付きで強制終了する

次の例では、テーブル内のレコード数をカウントし、カウントしたレコード数が 100 未満である場合は、ESCAPE コマンドを使ってスクリプトを強制終了します。

```
COUNT
ESCAPE IF COUNT1 < 100
```

備考

ESCAPE を使用する場面

ESCAPE を使用する場面は、条件に基づいてスクリプトまたはサブスクリプトの実行を停止する場合や、実行中のすべてのスクリプトの実行を停止する場合です。

サブスクリプト内で ESCAPE を使用する

サブスクリプト内で ESCAPE を実行すると、そのサブスクリプトが実行を停止し、メインのスクリプトが、そのサブスクリプトを呼び出した DO SCRIPT コマンドから処理を再開します。

サブスクリプトの ESCAPE コマンドに ALL オプションを追加した場合は、サブスクリプトとメインのスクリプトがどちらも処理を停止します。

```
ESCAPE ALL
```

EVALUATE コマンド

レコード サンプリングまたは金額単位サンプリングでは、サンプリングされたデータで決定された誤謬を基に母集団全体を予測し、逸脱率の上限または誤謬額を計算します。

レコードのサンプリング 金額単位のサンプリング

構文

EVALUATE RECORD CONFIDENCE *信頼度* SIZE *サンプルサイズ* ERRORLIMIT *誤謬の数* <TO SCREEN|*ファイル名*}>

パラメーター

> **メモ**
> 値を指定する際、3桁の区切り記号やパーセント記号は含めないでください。

名前	説明
RECORD	レコード サンプルで見つかった誤謬を評価する
CONFIDENCE *信頼度*	サンプル サイズの計算時に指定したのと同じ信頼度
SIZE *サンプル数*	サンプルのレコード数。 > **メモ** > 抽出した実際のサンプル サイズを指定します。これは、Analytics によって当初計算されたサンプル サイズと異なっていてもかまいません。
ERRORLIMIT *誤謬の数*	サンプルで見つかった誤謬または逸脱の総数。
TO SCREEN | *ファイル名*	コマンドの結果を送信する場所: ◦ **SCREEN** - は Analytics の表示領域に結果を表示します ◦ ***ファイル名*** -は結果の保存先となるファイルです。 *ファイル名*には、適切なファイル拡張子を付けた文字列を引用符で囲んで指定します。 例: TO "Output.TXT" デフォルトでは、テーブル ファイルは、Analytics プロジェクトが入っているフォルダーに保存されます。 既存の異なるフォルダーにファイルを保存するには、絶対または相対ファイル パスを指定します。 • TO "C:\Output.TXT" • TO "Results\Output.TXT"

Analytics の出力変数

名前	含む
MLE*n*	コマンドによって計算された最大誤謬率(計算された上限逸脱率)。

例

レコードの母集団全体に対して、サンプルされたデータで見つかった誤謬を推定する

サンプルデータのテストを完了し、見つかったコントロール逸脱を記録しました。検出したすべての誤謬を母集団に対して投影できます。

以下の例は、サンプリングされたデータで見つかった2つの誤謬を基に母集団全体を予測し、**最大誤謬率**(計算される上限逸脱率)を6.63%と算出しました。

```
EVALUATE RECORD CONFIDENCE 95 SIZE 297 ERRORLIMIT 7 TO SCREEN
```

Analytics が誤謬を評価する際に値を計算する方法の詳細については、レコード サンプルの誤謬の評価を参照してください。

備考

> **メモ**
>
> このコマンドの動作の詳細については、Analytics のヘルプを参照してください。

構文

```
EVALUATE MONETARY CONFIDENCE 信頼度 <ERRORLIMIT 簿価, 誤謬額 <,...n>>
INTERVAL 間隔値 <TO SCREEN|ファイル名>
```

パラメーター

> **メモ**
>
> 値を指定する際、3桁の区切り記号やパーセント記号は含めないでください。

名前	説明
MONETARY	金額単位サンプルで見つかった誤謬を評価する
CONFIDENCE *信頼度*	サンプル サイズの計算時に指定したのと同じ信頼度
ERRORLIMIT *簿価,誤謬額*	サンプルで見つかった虚偽表示の全誤謬。 虚偽表示額と金額の簿価をカンマで区切って指定します。たとえば、金額の簿価が $1,000 で、金額の監査額が $930 の場合は、1000,70 と指定します。 過剰計上は正の金額として、過少計上は負の金額として指定します。たとえば、金額の簿価が $1,250 で、金額の監査額が $1,450 の場合は、1250,-200 と指定します。 *簿価と誤謬額*のペアが複数組ある場合はカンマで区切ってください。 `1000,70,1250,-200`
INTERVAL *間隔値*	サンプルの抽出時に使用した間隔値。 **メモ** 使用した間隔値は、Analytics によって当初計算された間隔値と異なっていてもかまいません。
TO SCREEN | *ファイル名*	コマンドの結果を送信する場所: ◦ **SCREEN** - は Analytics の表示領域に結果を表示します ◦ ***ファイル名*** -は結果の保存先となるファイルです。 *ファイル名*には、適切なファイル拡張子を付けた文字列を引用符で囲んで指定します。例: `TO "Output.TXT"` デフォルトでは、テーブル ファイルは、Analytics プロジェクトが入っているフォルダーに保存されます。 既存の異なるフォルダーにファイルを保存するには、絶対または相対ファイル パスを指定します。 • `TO "C:\Output.TXT"` • `TO "Results\Output.TXT"`

Analytics の出力変数

名前	含む
MLE*n*	コマンドによって計算された推定誤謬額(推定による虚偽表示)。
UEL*n*	コマンドによって計算された最大誤謬額(虚偽表示上限)。

例

レコードの母集団全体に対して、サンプルされたデータで見つかった誤謬を推定する

サンプルデータのテストを完了し、見つかった虚偽表示を記録しました。検出したすべての誤謬を母集団に対して投影できます。

以下の例は、サンプリングされたデータで見つかった誤謬を基に母集団全体を予測し、以下を含む7つの値を計算しています。

- **基準となる精度** - サンプリングリスクの基本的な許容誤謬(18,850.00)
- **推定誤謬** - 推定による母集団全体の虚偽表示額(1,201.69)
- **最大誤謬率** - 母集団の虚偽表示上限(22,624.32)

```
EVALUATE MONETARY CONFIDENCE 95 ERRORLIMIT 1000,70,1250,-200,3200,900
INTERVAL 6283.33 TO SCREEN
```

Analytics が誤謬を評価する際に値を計算する方法の詳細については、金額単位サンプルの誤謬の評価を参照してください。

備考

> **メモ**
>
> このコマンドの動作の詳細については、Analytics のヘルプを参照してください。

EXECUTE コマンド

Analytics の外部にあるアプリケーションやプロセスを実行します。Windows の Run コマンドをエミュレートします。Windows のコマンド プロンプトを操作するために使用できます。

> **メモ**
>
> EXECUTE コマンドでは、Analytics 外部のオペレーティング システムやアプリケーションを操作することができるため、Analytics のネイティブ機能の範囲を超えた技術的な問題が発生する可能性があります。
>
> サポート チームでは、Analytics 内における EXECUTE コマンドの操作の手助けはできますが、Analytics 外部のプロセスやアプリケーションで発生する問題についてはサポートの対象外です。

構文

EXECUTE *Windows_Run_コマンド構文* <ASYNC>

パラメーター

名前	説明
Windows_Run_コマンド構文	実行するアプリケーションの名前、開くフォルダーまたはファイル、あるいは実行するコマンドと、任意の必要な引数やコマンド スイッチ。 引用符で囲まれた有効な Windows Run コマンドである必要があります。
ASYNC 省略可能	このコマンドを非同期モードで実行します。 非同期モードでは、Analytics スクリプトは、EXECUTE コマンドによって開始されたプロセスが完了するのを待たずに、実行を続行します。 ASYNC を省略した場合は、Analytics スクリプトが続行されるには、EXECUTE コマンドによって開始されたプロセスが完了する必要があります。外部プロセスが実行中の場合は、Analytics にはアクセスできません。 > **メモ** > Analytics のコマンド ラインから EXECUTE を使用する場合は、ASYNC を指定する必要があります。

Analytics の出力変数

名前	含む
RETURN_CODE	EXECUTE コマンドを使用した外部アプリケーションまたはプロセス実行によって返されるコード (リターン コード)。 **リターン コードの概要** リターン コードは外部アプリケーションまたはプロセスによって生成される数値であり、Analytics に返されることで外部プロセスの結果を示します。Analytics はリターン コードを出力せず、入力として受け付けるだけです。 **一般的なリターン コード** 通常、リターン コードは数値で、特定の通知やエラー メッセージが割り当てられています。たとえば、リターン コード "0" は "処理が正常終了した" ことを示します。リターン コード "2" は "指定されたファイルが見つからない" ことを示します。 **特定のリターン コードの意味** 特定のリターン コードとその意味は外部アプリケーションまたはプロセスによって異なります。リターン コードは "エラー コード" や "終了コード" とも呼ばれ、それらの意味は、関連する外部アプリケーションのドキュメントに記載されている可能性があります。リターン コードの一覧はインターネット上でも見ることができます。 **デフォルト モードでのみ作成された変数** RETURN_CODE 変数は、デフォルト モードでEXECUTE コマンドが実行されるときに、作成されます。変数は、コマンドが非同期モードで実行されるときには、作成されません。

例

アプリケーションを開く

Microsoft Excel を開く:

```
EXECUTE "Excel"
```

Adobe Acrobat Reader を開く

```
EXECUTE "AcroRd32.exe"
```

アプリケーションを閉じる

Microsoft Excel を閉じる:

```
EXECUTE "TASKKILL /f /im Excel.exe"
```

> **メモ**
>
> /f スイッチは慎重に使用してください。これは、変更の保存を確認するダイアログ ボックスなど、一切のダイアログ ボックスを表示しないで、アプリケーションを強制的に閉じます。

ファイルを開く

Excel ワークブック `AP_Trans.xlsx` を開く

```
EXECUTE '"C:\ACL Projects\Source Data\AP_Trans.xlsx"'
```

新しいフォルダーを作成する

新しいフォルダーのソース データを作成する

```
EXECUTE 'cmd /c MD "C:\ACL Projects\Source Data"'
```

外部スクリプト、または Analytics の外部のバッチ ファイル(.bat) を実行する

スクリプト `My_Batch.bat` を実行する

```
EXECUTE '"C:\ACL Projects\Batch Files\My_Batch.bat"'
```

Analytics の外部のバッチ ファイルにパラメーターを渡す

2 つのパラメーターを `My_Batch.bat` に渡します。パラメーターには、リテラルまたは Analytics 変数を指定できます。

```
EXECUTE '"C:\ACL Projects\Batch Files\My_Batch.bat" param1%v_param2%'
```

他の Analytics スクリプトで Analytics スクリプトを実行する

`AP Trans Tests.acl` で "AP_Trans_script" を実行する"

```
EXECUTE 'aclwin.exe "C:\ACL Projects\AP Trans Tests.acl" /b AP_Trans_script'
```

> **メモ**
>
> 別のプロジェクト内で Analytics スクリプトを実行すると、もう 1 つの Analytics インスタンスが起動されます。そのもう 1 つの Analytics インスタンスを閉じたら制御が 1 番目のインスタンスに戻るように、2 番目のプロジェクト内のスクリプトの末尾に QUIT コマンドを記述しておく必要があります。

Analytics スクリプトに待ち時間を組み込む

どちらの例も、30 秒の待ち時間を設けます。

```
EXECUTE "TIMEOUT /t 30"
```

```
EXECUTE "cmd /c PING -n 31 127.0.0.1 > nul"
```

備考

EXECUTE を使用して便利なタスクを実行する

EXECUTE コマンドは、Windows コマンドおよび DOS コマンドを Analytics のコマンド ラインから、または Analytics スクリプトから実行できるようにします。

この機能を使用すると、以下に挙げる多様で便利なタスクを実行することにより、Analytics スクリプトの自動化を向上させることができます。これらのタスクは、ACLScript 構文だけを使用しては不可能です。

EXECUTE 使用して開始できるタスクの例

他のプログラムやアプリケーションを開いて、Analytics スクリプトで必要なタスクを実行する	バッチ ファイルにパラメーターを渡す	ネットワークの場所からデータにアクセスする	Active Directory アカウント一覧を組み込む
任意のファイルをその既定のアプリケーションで開く	他の Analytics スクリプトで Analytics スクリプトを実行する	FTP を使用して、リモートの場所からデータにアクセスする	VBScript と統合する
Analytics の外に存在するファイルやフォルダーのコピー、移動、作成、削除、または比較などのファイルおよびフォルダーの管理タスクを実行する	Analytics スクリプトに待ち時間を組み込む	データを圧縮または解凍する	SQL データベースと統合する
外部スクリプト、または Analytics の外部のバッチ ファイル(.bat) を実行する	Analytics スクリプトに Windows のタスク スケジュールを組み込む	データを暗号化または復号する	Web ページを開く

> **メモ**
>
> これらのタスクを実行する方法の詳細は、Galvanize のヘルプ ドキュメントには記載されていません。サポートが必要な場合は、適切な Windows オペレーティング システムのドキュメント、またはその他サード パーティのドキュメントを参照してください。

デフォルト モードと非同期モード

デフォルト モードまたは非同期モードで、EXECUTE コマンドを実行できます。

- **デフォルト モード** - EXECUTE で開始するプロセスは、Analytics スクリプトを続行する前に、完了する必要があります。

  外部プロセスが実行中の場合は、Analytics にはアクセスできません。
- **非同期モード** - Analytics スクリプトは、EXECUTE コマンドによって開始されたプロセスが完了するのを待たずに、実行を続行します。

  外部プロセスが実行中の場合は、Analytics にはアクセスし続けることができます。

  ASYNC 指定する場合、EXECUTE コマンドは非同期モードで実行されます。

使用するモード

EXECUTE コマンドを使用する Analytics スクリプトを作成する場合は、どちらのモードの操作が適切であるかを考慮する必要があります。

デフォルト モードを使用する	非同期モードを使用する / ASYNC
◦ ファイルおよびフォルダー管理タスク ◦ 待機期間の指定 ◦ 後続のタスクが依存するタスク ◦ すぐ後のスクリプトの実行が RETURN_CODE 変数の結果に依存します	◦ 外部タスクにより、アプリケーション インターフェイスまたはポップアップ ダイアログボックスが開きます

自動実行される Analytics スクリプト

EXECUTE コマンドを含む Analytics スクリプトを自動で実行する場合は、次のいずれかの方法を使用します。

- タスクで非同期モードを使用することにより、アプリケーション インターフェイスまたはポップアップ ダイアログボックスが開きます
- 自動スクリプトでインターフェイス要素を開かないようにします

> **メモ**
>
> インターフェイス要素が閉じるまで、実行中のプロセスを表します。これらのインターフェイス要素がデフォルト モードで EXECUTE とともに開く場合、Analytics スクリプトの後続の行は実行されず、スクリプトがハングします。

アナリティクス スクリプトの EXECUTE コマンド

ロボットまたは AX Server のアナリティクス スクリプトで、EXECUTE コマンドを使用する場合は、特に実行するコマンドを構成する必要があります。詳細については、次を参照してください。

- **ロボット** -ロボット エージェントの構成
- **AX Server** -AX Server 設定

二重引用符

EXECUTE コマンドで使用する Windows の Run コマンドの構文は、一重引用符または二重引用符で囲む必要があります。

次の例は、Windows `MD` コマンドを使用して新しいフォルダーを作成しています。

```
EXECUTE 'cmd /c md C:\New_Data_Folder'
```

ネストされた二重引用符

Run コマンド構文内のパスにスペースが含まれている場合は、そのパスも引用符で囲む必要があります。

引用符でパスを囲むときには、2 つのオプションがあります。

- **単一引用符内の二重引用符** - 単一引用符を使用して、Run コマンド文字列全体を囲み、二重引用符を内部で使用し、パスを囲みます

  ```
  EXECUTE 'cmd /c md "C:\New Data Folder"'
  ```

  2 番目の方法よりも、この方法の方が容易である場合があります。

  > **メモ**
  >
  > 入れ子の順序を逆にする、つまり、二重引用符で文字列全体を囲み、一重引用符でパスを囲むと、動作しません。

- **2 つの二重引用符** - 二重引用符を使用して、Run コマンド文字列全体を囲み、2 つの二重引用符を内部で使用してパスを囲みます。

  ```
  EXECUTE "cmd /c md ""C:\New Data Folder"""
  ```

  この 2 番目の方法を使用する場合、内部的に使用される 2 つの二重引用符は隣接する必要があり、スペースを含めることはできません。

内部コマンドおよび外部コマンド

Windows コマンドは、**内部**または**外部**のいずれかです。

- **内部コマンド** -は、コマンド プロンプトからのみ実行できます。これはつまり、コマンドを指定する前に、cmd /c または cmd /k を用いて、コマンド シェルを開いておく必要があるということです。
- **外部コマンド** -は、コマンド プロンプトからでも、直接 EXECUTE コマンドを用いても実行できます。これはつまり、コマンド シェルを開くことは任意であり、必須ではないということです。

次の例では、違いを説明するために、内部 Windows DIR コマンド(ディレクトリの内容を表示する) と外部 Windows COMP コマンド(2 つのファイルを比較する) を使用しています。

```
EXECUTE 'cmd /k dir "C:\ACL DATA\Sample Data Files"'
EXECUTE 'comp C:\File_1.txt C:\File_2.txt'
```

Windows コマンドを含んでいる外部スクリプトまたはバッチ ファイルを作成し、バッチ ファイルを起動するためだけにEXECUTE コマンドを使用することによって、この複雑な事態を避けることができます。例：

```
EXECUTE 'C:\My_Batch.bat'
```

複数行の Run コマンド構文

EXECUTE コマンドは、複数行の Run コマンド構文をサポートしていません。Analytics スクリプトに複数行の Run コマンドを組み込むには、以下の方法のいずれかを使用します。

方法	例
Run コマンドごとに EXECUTE コマンドを繰り返します。	EXECUTE 'cmd /c md "C:\New Data Folder"' EXECUTE 'cmd /c copy C:\File_1.txt "C:\New Data Folder"'
'&' を使用して、複数の Run コマンドを結合します。	EXECUTE 'cmd /c md "C:\New Data Folder" & copy C:\File_1.txt "C:\New Data Folder"'
複数行の Run コマンドを含んでいる外部スクリプトまたはバッチファイルを作成し、バッチファイルを起動するためだけに EXECUTE コマンドを使用します。	EXECUTE 'C:\My_Batch.bat'

EXPORT コマンド

指定されたファイル形式に、または HighBond 内のリザルトに、Analytics からデータをエクスポートします。

構文

EXPORT <FIELDS> *フィールド名* <AS *エクスポート名*> <...*n*>|<FIELDS> ALL} <UNICODE> *エクスポート タイプ*<SCHEMA> PASSWORD *番号* TO {*ファイル名*|*aclgrc_id*} <OVERWRITE> <IF *テスト*> <WHILE *テスト*> <{FIRST *範囲*|NEXT *範囲*}> <APPEND> <KEEPTITLE> <SEPARATOR *文字*> <QUALIFIER *文字*> <WORKSHEET *ワークシート名*> <DISPLAYNAME>

パラメーター

名前	説明
FIELDS *フィールド名* AS *エクスポート名* <...*n*> | FIELDS ALL	エクスポートするフィールド。 ◦ ***フィールド名*** - 指定されたフィールドをエクスポートします フィールド名はスペースで区切ります。 任意で AS *エクスポート名*を使って、エクスポート ファイル内のフィールドとは異なる名前を追加することもできます。*エクスポート名*は引用符で囲みます。 ACLGRC(HighBond) へのエクスポートを行う場合は、AS を DISPLAYNAME パラメーターと結合することもできます。詳細については、"HighBond のリザルトにエクスポートを行う際の DISPLAYNAME と AS の相互作用" ページ 192を参照してください。 ◦ **ALL** - テーブルのすべてのフィールドをエクスポートします
UNICODE 省略可能	Analytics の Unicode 版でのみ利用可能です。テキスト (ASCII)、区切られたテキスト (DELIMITED)、および XML ファイルのみ、および Windows クリップボード CLIPBOARD) 出力に適用されます。 Unicode UTF-16 LE 文字エンコードが適用されている Analytics データをエクスポートします。 ◦ **UNICODE を指定** -エクスポートしているデータが拡張 ASCII(ANSI) によってサポートされていない場合 ◦ エクスポートしているデータのすべての文字が拡張 ASCII(ANSI) によってサポートされている場合は、**Unicode** - を指定しないでください エクスポートされたデータは拡張 ASCII(ANSI) としてエンコードされます。 **メモ** サポートされていない文字はエクスポートされたファイルから省略されます。 詳細については、ACL Unicode 製品を参照してください。
エクスポート形式	以下のオプションのいずれかを使用して、ファイルの出力形式または宛先を指定します。 ◦ **ACCESS** - Microsoft Access データベース ファイル (.mdb)

名前	説明
	デフォルトでは、データは Unicode としてエクスポートされます。 ◦ **ACLGRC** - HighBond のリザルト ◦ **ASCII** - ASCII プレーン テキスト (.txt) ◦ **CLIPBOARD** - Windows クリップボード ◦ **DBASE** - dBASE 互換ファイル (.dbf) ◦ **DELIMITED** - 区切り文字付きテキスト ファイル (.del) ◦ **EXCEL** - - Microsoft Excel 1997 ～ 2003 と互換性のある Excel ファイル(.xls) ◦ **JSON** - JSON ファイル (.json) ◦ **LOTUS** - Lotus 123 ファイル ◦ **WDPF6** - Wordperfect 6 ファイル ◦ **WORD** - MS Word ファイル (.doc) ◦ **WP** - Wordperfect ファイル ◦ **XLS21** - Microsoft Excel バージョン 2.1 のファイル ◦ **XLSX** - Microsoft Excel の .xlsx ファイル デフォルトでは、データは Unicode としてエクスポートされます。 ◦ **XML** - XML ファイル (.xml)
SCHEMA 省略可能	XML ファイルへの出力のみに適用されます。 エクスポートされた XML ファイルに XML スキーマを含めます。XML スキーマには、フィールドのデータ型など、XML ファイルの構造を記述するメタデータが含まれています。 ファイルがエクスポートされたら、スキーマに対してファイルを検証することができます。
PASSWORD *番号*	**メモ** HighBond のリザルトに適用されます。 使用するパスワード定義。 実際のパスワードを入力することを求めたり指定したりするのには、PASSWORD *番号*の構文は使用しません。パスワード定義とは、以前に PASSWORD コマンドか SET PASSWORD コマンドを使用して入力または設定されたパスワードのことです。 *番号*はパスワード定義の番号です。たとえば、以前に 2 つのパスワードをスクリプトで設定したり入力したりしている場合には、PASSWORD 2 により、2 番目のパスワードを使用することを指定するなどします。 パスワードの入力または設定の詳細については、次を参照してください。 • "PASSWORD コマンド" ページ 345 • "SET コマンド" ページ 404 • PASSWORD アナリティクス タグ パスワードの入力または設定の詳細については、次を参照してください。 • PASSWORD コマンド • SET コマンド • PASSWORD アナリティクス タグ PASSWORD *番号* は、TO の直前またはコマンド構文の文字列の最後に配置する必要があります。このパスワードは、HighBond のアクセストークンになります。詳細については、"HighBond のリザルトへのエクスポート" ページ 191を参照してください。

<table>
<tr><th>名前</th><th>説明</th></tr>
<tr><td></td><td>
メモ
PASSWORD は必要な場合と不要な場合があります。スクリプトを実行する環境によって異なります。
<table>
<tr><td>Analytics
(オンライン アクティベーション)</td><td>PASSWORD は必要ではありません。
現在のユーザーの HighBond アクセストークンが自動的に使用されます</td></tr>
<tr><td>Analytics
(オフライン アクティベーション)</td><td rowspan="4">PASSWORD が必要です。</td></tr>
<tr><td>ロボット</td></tr>
<tr><td>Analytics Exchange</td></tr>
<tr><td>分析アプリウィンドウ</td></tr>
</table>
</td></tr>
<tr><td>TO ファイル名 | ACL GRC ID</td><td>
エクスポート先：
◦ TO ファイル名 -データをファイルにエクスポート
必要に応じて、絶対ファイル パスまたは相対ファイル パスのいずれかを含めることができますが、その Windows フォルダーは既に存在している必要があります。ファイル名 の値は引用符で囲まれた文字列として指定する必要があります。
◦ TO ACL GRC ID - データを HighBond のリザルト
にエクスポート ACL GRC ID 値には、統制テストの ID 番号が含まれている必要があるほか、北米以外のデータ センターへエクスポートする場合には、そのデータ センターのコードも含まれている必要があります。ACL GRC ID 値は引用符で囲む必要があります。
統制テスト ID 番号とデータ センター コードは、記号 (@) で区切られる必要があります。例：TO "99@eu".
統制テスト ID 番号がわからない場合は、Analytics のユーザー インターフェイスを使用して、リザルトへのエクスポートを開始します。統制テスト ID 番号を特定したら、エクスポートをキャンセルします。詳細については、ACL GRC への例外のエクスポートを参照してください。
データ センター コードは、どの地域の HighBond サーバーにデータをエクスポートするのかを指定します。
• ap - Asia Pacific(アジア太平洋)
• au - Australia(オーストラリア)
• ca - Canada(カナダ)
• eu - Europe(ヨーロッパ)
• us - North America(北米)
組織にインストールされた HighBond に対して承認されているデータ センター コードのみを使用できます。北米のデータ センターがデフォルトであるため、@us を指定するのはオプションです。
</td></tr>
<tr><td>OVERWRITE</td><td>HighBond のリザルトへのエクスポートにのみ適用されます。</td></tr>
</table>

名前	説明
省略可能	対象統制テスト(テーブル) のすべての既存のデータが、エクスポートされたデータによって上書きされます。データを上書きするには、ターゲット コレクションで、Professional 部門長ロールが必要です。 OVERWRITE を省略した場合、統制テスト(テーブル) に既にデータが存在する場合は、エクスポートされたデータが既存のデータの最後に追加されます。リザルトでの追加の詳細については、以下の「備考」を参照してください。 対象とする統制テスト(テーブル) の解釈は、上書きか追加かには関係なく、インポートされたデータを反映するために動的に更新されます。
IF *テスト* 省略可能	各レコードを処理するために真である必要がある条件式。コマンドは、その条件を満たすレコードに対してのみ実行されます。 **メモ** IF パラメーターは、任意の範囲パラメーター(WHILE、FIRST、NEXT) が適用された後に、テーブルに残るレコードに対してのみ評価されます。
WHILE *テスト* 省略可能	各レコードを処理するために真である必要がある条件式。条件が false と評価するか、テーブルの最後に達したら、コマンドは実行を中止します。 **メモ** WHILE を FIRST または NEXT とともに使用する場合は、1 つの制限に達するとすぐに、レコードの処理が停止します。
FIRST *範囲* | NEXT *範囲* 省略可能	処理するレコード数: ◦ **FIRST** - 指定されたレコード数に達するまで、最初のレコードから処理を開始します ◦ **NEXT** - 指定されたレコード数に達するまで、現在選択されているレコードから処理を開始します *範囲*は処理するレコード数を指定します。 FIRST とNEXT を省略すると、すべてのレコードがデフォルトで処理されます。
APPEND 省略可能	テキスト(ASCII) および区切り文字付きテキスト(DELIMITED) ファイルのみに適用されます。 コマンドの出力を既存ファイルに上書きしないで、そのファイルの末尾に追加します。 **メモ** コマンドの出力と既存のファイルの構造が同一であるようにする必要があります。 • 同じフィールド • 同じフィールド順序 • 一致するフィールドが同じ長さ • 一致するフィールドが同じデータ型 出力は、既存ファイルとの間でファイル構造が違っている場合でも、Analytics によって既存ファイルに追加されます。出力と既存のファイルの構造が一致しない場合は、データが混在、不足、不正確になります。
KEEPTITLE 省略可能	テキスト(ASCII) および区切り文字付きテキスト(DELIMITED) ファイルのみに適用されます。 Analytics フィールド名とエクスポートされたデータを含めます。省略された場合、フィールド名は出力ファイルに表示されません。

名前	説明
SEPARATOR *文字* 省略可能	区切り文字付きテキスト ファイル(DELIMITED) への出力のみに適用されます。 フィールド間の区切りとして使用する文字。文字は引用符で囲まれた文字列として指定する必要があります。 デフォルトでは、カンマが使用されます。
QUALIFIER *文字* 省略可能	区切り文字付きテキスト ファイル(DELIMITED) への出力のみに適用されます。 フィールド値を折り返すため と識別するためにテキスト修飾子として使用する文字。文字は引用符で囲まれた文字列として指定する必要があります。 デフォルトでは、二重引用符が使用されます。
WORKSHEET *ワークシート名* 省略可能	Microsoft Excel (.xlsx) ファイルにのみ適用されます。 新規または既存の Excel ファイルに作成される Excel ワークシートの名前。 デフォルトでは、ワークシート名としてエクスポートする Analytics テーブルの名前が使用されます。 *ワークシート名*には、英数文字とアンダースコア文字(_) のみを含めることができます。名前に特殊文字や空白を使用したり、名前を数字で始めることはできません。値を引用符で囲むのは任意です。 エクスポートするときに Excel ワークブックとワークシートを上書きする詳細については、"WORKSHEET パラメーターと上書き" ページ 190を参照してください。
DISPLAYNAME 省略可能	ACLGRC(HighBond) にのみ適用されます。 フィールド名自体に影響を与えることなく表示名がリザルトの列見出しに表示されるように、フィールド名をフィールド名として、表示名を表示名としてエクスポートします。 DISPLAYNAME を AS と組み合わせて使用することもできます。詳細については、"HighBond のリザルトにエクスポートを行う際の DISPLAYNAME と AS の相互作用" ページ 192を参照してください。

例

Excel .xlsx ファイルにデータをエクスポートします。

Vendor テーブル内の特定のフィールドを Excel .xlsx ファイルにエクスポートするには、次のようにします。

```
OPEN Vendor
EXPORT FIELDS Vendor_No Vendor_Name Vendor_City XLSX TO "VendorExport"
```

ワークシート名を指定して、データを Excel .xlsx ファイルへエクスポートする

Vendor テーブル内の特定のフィールドを Excel .xlsx ファイル内での **Vendors_US** というワークシートにエクスポートするには、次のようにします。

```
OPEN Vendor
EXPORT FIELDS Vendor_No Vendor_Name Vendor_City XLSX TO "VendorExport" WORKSHEET
Vendors_US
```

すべてのフィールドを区切りファイルにエクスポートする

Vendor テーブル内のフィールドを以下の 2 つの区切り文字付きテキスト ファイルにエクスポートします。

```
OPEN Vendor
EXPORT FIELDS ALL DELIMITED TO "VendorExport"
```

GROUP を使用して、データを複数の区切り文字付きテキスト ファイルへエクスポートする

Vendor テーブル内の特定のフィールドを以下の 2 つの区切り文字付きテキスト ファイルにエクスポートします。

- "A" から "M" の業者名の 1 つのファイル
- "N" から "Z" の業者名の 1 つのファイル

GROUP コマンドと IF 条件を使って、各レコードの業者名をテストします。

```
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
```

HighBond のリザルトへのデータのエクスポート

AR_Exceptions テーブル内の特定のフィールドを HighBond のリザルトにエクスポートするとします。対象統制テスト (テーブル) 内の既存のデータを上書きします。

```
OPEN AR_Exceptions
EXPORT FIELDS No Due Date Ref Amount Type ACLGRC PASSWORD 1 TO "10926@us"
OVERWRITE
```

備考

> **メモ**
>
> このコマンドの動作の詳細については、Analytics のヘルプを参照してください。

EXPORT と GROUP コマンドの使用

ほとんどのエクスポート形式は、GROUP コマンドを使ってデータを複数ファイルへ同時にエクスポートすることができます。

Microsoft Excel および Microsoft Access にデータをエクスポートする場合は、同時に 1 つのファイルのみ作成することができます。

Excel へのエクスポート

データを Excel ファイルにエクスポートするときには、以下の制限事項があります。

レコード数	◦ **Excel 2007 以降 (*.xlsx)** -最大レコード件数は 1,048,576 件 ◦ **Excel 97 および 2003** -最大レコード件数は 65,536 件 この最大件数を超える Analytics テーブルでもエクスポートは行えますが、上限を超えた分のレコードは無視されエクスポートされません。
フィールドの長さ	◦ 特定のフィールド長上限なし ◦ フィールド長の合計が、レコード全体の長さ上限である 32 KB (非 Unicode 版 Analytics では 32,765 文字、Unicode 版 Analytics では 16,382 文字) を超えることはできません。 ◦ Excel 2.1 の場合は、最大 247 文字
フィールド名の長さ	◦ 最大 64 文字 ◦ Excel 2.1 の場合は、最大 248 文字

WORKSHEET パラメーターと上書き

Analytics テーブルから Excel ファイルにエクスポートするときに WORKSHEET コマンドを使用する場合としない場合の結果は、以下で説明します。

一致	説明	使用される WORKSHEET パラメーター	使用されない WORKSHEET パラメーター
一致する Excel ファイル名がない	◦ TO に指定されたファイル名の値が既存のどの Excel ファイルの名前とも一致しない	指定された名前のワークシートで新しい Excel ファイルが作成される	エクスポートされた Analytics テーブルの名前を使用するワークシートで、新しい Excel ファイルが作成される
一致する Excel ファイル名がある 一致するワークシート名がある	◦ TO に指定されたファイル名の値と既存の Excel ファイルの名前が一致する ◦ WORKSHEET に指定されたワークシート名が Excel ファイルのワークシート名と一致しない	指定された名前のワークシートが既存の Excel ファイルに追加される	エクスポートされた Analytics テーブルの名前を使用するワークシートで、既存の Excel ファイルが新しい Excel ファイルによって上書きされる
Excel ファイル名とワークシート名が一致する	◦ TO に指定されたファイル名の値と既存の Excel	最初に Analytics から作成された場合は、指定された名	エクスポートされた Analytics テーブルの名前を使用する

一致	説明	使用される WORKSHEET パラメーター	使用されない WORKSHEET パラメーター
	ファイルの名前が一致する ◦ WORKSHEET に指定されたワークシート名が Excel ファイルのワークシート名と一致する	前のワークシートが既存のワークシートを上書きします。 既存のワークシートが最初に直接 Excel で作成された場合は、エラー メッセージが表示され、エクスポート処理がキャンセルされます。	ワークシートで、既存の Excel ファイルが新しい Excel ファイルによって上書きされる

HighBond のリザルトへのエクスポート

以下の表では、リザルトで統制テストにエクスポートする方法の詳細について説明しています。

項目	詳細
必要なアクセス許可	リザルトの統制テストへ結果をエクスポートする機能は、特定の HighBond 役割の割り当て、または管理者特権を必要とします。 ◦ リザルト コレクションのプロフェッショナル ユーザーまたはプロフェッショナル部長の役割を担うユーザーは、そのコレクション内のあらゆる統制テストに結果をエクスポートすることができます。 **メモ** 監査部門長の役割のユーザーのみが統制テストの既存のデータをエクスポートおよび上書きできます。 ◦ HighBond システム管理者およびリザルト管理者は、HighBond 組織、または管理する組織のすべてのコレクションで自動的にプロフェッショナル部門長の役割を取得します。
エクスポートの制限事項	統制テストにエクスポートする際、次の制限が適用されます。 ◦ エクスポートごとに最大 100,000 レコード ◦ 統制テストごとに最大 100,000 レコード ◦ レコードごとに最大 500 フィールド ◦ フィールドごとに最大 256 文字 同じ統制テストへは、複数回～全体の上限回数までエクスポートを行うことができます。
フィールドの追加	既存のフィールドと一致する物理フィールド名がある場合は、Analytics テーブル内の順序に関係なく、エクスポートされたフィールドは統制テストの既存のフィールドの最後に追加されます。 Analytics では、物理フィールド名はテーブル レイアウトの名前です。既存のフィールド名に一致しないエクスポートされるフィールドは、リザルトのテーブルに追加列として追加されます。 Analytics とリザルトのフィールドの表示名は、考慮されません。ただし、オプションの AS *エクスポート名* パラメーターを使用する場合に、*DISPLAYNAME* を使用しないときは、*エクスポート名* 値が物理フィールド名として使用されます。 アンケート フィールドの末尾にデータを追加するときには、リザルトの列の表示名は、アンケート構成で指定された名前のままです。

項目	詳細
	メモ リザルトとAnalytics の間でデータを往復し、データがリザルトで不一致になる場合は、不一致のフィールド名がある可能性があります。 詳細については、"リザルト データをインポートおよびエクスポートするときのフィールド名の考慮事項" ページ 254を参照してください。
パスワード定義の作成とパスワード値の指定	PASSWORD コマンド PASSWORD コマンドを使用して、HighBond に接続するための番号付けされたパスワード定義を作成した場合、パスワードの値が指定されていないと、スクリプトを接続しようとするときにパスワード プロンプトが表示されます。 詳細については、"PASSWORD コマンド" ページ 345を参照してください。 SET PASSWORD コマンド SET PASSWORD コマンドを使用して、HighBond に接続するための番号付けされたパスワード定義を作成した場合、パスワードの値が指定されていれば、パスワード プロンプトは表示されません。これは、無人で実行するように設計されたスクリプトに適しています。 詳細については、SET PASSWORD コマンドを参照してください。 HighBond アクセストークン どの方法を用いてパスワード定義を作成したかにかかわらず、必要なパスワードの値は、HighBond アクセストークンです。 ◦ **PASSWORD による方法** - ユーザーは、[**ツール > HighBond アクセストークン**]を選択し、HighBond にサインインして、アクセストークンを取得できます。アクセストークンが返されるので、ユーザーはこれをコピーして、パスワード プロンプトに貼り付けることができます。 ◦ **SET PASSWORD** による方法 - Analytics スクリプト内の SET PASSWORD コマンド構文にアクセストークンを挿入するには、**スクリプト エディター**で右クリックして [**挿入 > HighBond トークン**]を選択し、次に HighBond にサインインします。スクリプト内のカーソル位置にアクセストークンが挿入されます。 **注意** 返されるアクセストークンは HighBond にサインインするために使用されるアカウントと一致します。他のユーザーが使用するスクリプトを作成している場合は、スクリプト作成者として、独自のアクセストークンを使用することは適切ではない場合があります。

HighBond のリザルトにエクスポートを行う際の DISPLAYNAME と AS の相互作用

下記のマトリクスは、Analytics からリザルトにフィールド名をエクスポートする際の DISPLAYNAME パラメーターと AS の相互作用を示しています。

	AS を指定しない場合	AS を指定した場合
DISPLAYNAME を指定しない場合	Analytics のフィールド名が、リザルトのフィールド名と表示名になります。	リザルトのフィールド名と表示名が、AS パラメーターの表示名になります。
DISPLAYNAME を指定した場合	Analytics のフィールド名が、リザルトのフィールド名になります。Analytics の表示名が、リザルトの表示名になります。	Analytics のフィールド名が、リザルトのフィールド名になります。リザルトの表示名が、AS パラメーターの表示名になります。

EXTRACT コマンド

Analytics テーブルからデータを抽出し、それを新しい Analytics テーブルに出力するか、または既存の Analytics テーブルへ追加します。レコード全体または選択したフィールドを抽出することができます。

構文

EXTRACT {RECORD|FIELDS *フィールド名* <AS *表示名*> <...*n*>|FIELDS ALL} TO *テーブル名* <IF *テスト*> <WHILE *テスト*> <FIRST *範囲*|NEXT *範囲*> <EOF> <APPEND> <OPEN> <LOCAL>

パラメーター

名前	説明
RECORD | FIELDS *フィールド名* | FIELDS ALL	出力に含めるフィールド： ◦ **RECORD** - ソースデータファイル内のレコード全体、つまり、テーブル内のすべてのフィールド、およびレコード内の未定義部分すべてが使用されます。 フィールドは、テーブルレイアウトに現れる順序と同じ並びで使用されます。 演算フィールドを保持します。 ◦ **FIELDS *フィールド名*** - 指定されたフィールドを使用します フィールドは一覧の順序で使用されます。 演算フィールドを出力先テーブル内の適切なデータ型(ASCII 型または Unicode 型(Analytics のエディションによる)、ACL 型(ネイティブの数値データ型)、日付時刻型、あるいは論理型)の物理フィールドに変換します。演算された実際の値を物理フィールドに設定します。 ◦ **FIELDS ALL** - テーブルのすべてのフィールドを使用します。 フィールドは、テーブルレイアウトに現れる順序と同じ並びで使用されます。 演算フィールドを出力先テーブル内の適切なデータ型(ASCII 型または Unicode 型(Analytics のエディションによる)、ACL 型(ネイティブの数値データ型)、日付時刻型、あるいは論理型)の物理フィールドに変換します。演算された実際の値を物理フィールドに設定します。
AS *表示名* 省略可能	FIELDS *フィールド名*を使って抽出を行う場合にのみ使用します。 新しい Analytics テーブルのビューにおけるフィールドの表示名(代替列見出し)。表示名をフィールド名、またはソーステーブル内の既存の表示名と同じにしたい場合は、AS を使用しないでください。 *表示名*の値は引用符で囲まれた文字列。列見出しを改行したい場合は、語句の間にセミコロン(;)を入れます。

名前	説明
	> **メモ** AS は新しいテーブルに抽出を行う場合にのみ使用します。既存のテーブルに追加している場合は、既存テーブルの代替列見出しが優先されます。
TO *テーブル名*	コマンドの結果を送信する場所： ◦ ***テーブル名*** -は、結果の保存先となる Analytics テーブルのことです。 *テーブル名*には、ファイル拡張子 .FIL を付けた文字列を引用符で囲んで指定する必要があります。例：`TO "Output.FIL"` デフォルトでは、テーブル データ ファイル(.fil) は、Analytics プロジェクトが入っているフォルダーに保存されます。 既存の異なるフォルダーにデータ ファイルを保存するには、絶対または相対ファイル パスを指定します。 • `TO "C:\Output.FIL"` • `TO "Results\Output.FIL"` > **メモ** テーブル名は 64 文字の英数字(.FIL 拡張子を含まない) に制限されています。名前にはアンダースコア文字(_) を使用できますが、他の特殊文字やスペースは使用できません。名前の先頭を数字にすることはできません。
IF *テスト* 省略可能	各レコードを処理するために真である必要がある条件式。コマンドは、その条件を満たすレコードに対してのみ実行されます。 > **メモ** IF パラメーターは、任意の範囲パラメーター(WHILE、FIRST、NEXT) が適用された後に、テーブルに残るレコードに対してのみ評価されます。
WHILE *テスト* 省略可能	各レコードを処理するために真である必要がある条件式。条件が false と評価するか、テーブルの最後に達したら、コマンドは実行を中止します。 > **メモ** WHILE を FIRST または NEXT とともに使用する場合は、1 つの制限に達するとすぐに、レコードの処理が停止します。
FIRST *範囲* | NEXT *範囲* 省略可能	処理するレコード数： ◦ **FIRST** - 指定されたレコード数に達するまで、最初のレコードから処理を開始します ◦ **NEXT** - 指定されたレコード数に達するまで、現在選択されているレコードから処理を開始します *範囲*は処理するレコード数を指定します。 FIRST と NEXT を省略すると、すべてのレコードがデフォルトで処理されます。
EOF 省略可能	ファイルの終わりに達した後、コマンドをもう一度実行します。 これにより、GROUP コマンド内でテーブルの最後のレコードが処理されることが保証されます。すべてのフィールドが以前のレコードを参照する演算フィールドである場合にのみ使用してください。
APPEND	コマンドの出力を既存ファイルに上書きしないで、そのファイルの末尾に追加します。

名前	説明
省略可能	**メモ** コマンドの出力と既存のファイルの構造が同一であるようにする必要があります。 • 同じフィールド • 同じフィールド順序 • 一致するフィールドが同じ長さ • 一致するフィールドが同じデータ型 出力は、既存ファイルとの間でファイル構造が違っている場合でも、Analytics によって既存ファイルに追加されます。出力と既存のファイルの構造が一致しない場合は、データが混在、不足、不正確になります。
OPEN 省略可能	コマンドを実行した後、コマンドによって作成されたテーブルを開きます。コマンドが出力テーブルを作成する場合にのみ有効です。
LOCAL 省略可能	Analytics プロジェクトと同じ場所に出力ファイルを保存します。 **メモ** Analytics テーブルである出力ファイルを含むサーバーテーブルに対してコマンドを実行するときにのみ適用されます。

例

テーブル内のすべてのレコードを新しいテーブルへ抽出する

次の例では、既存のすべてのレコードを新しい Analytics テーブルへ抽出することにより、**AR_Customer** テーブルの正確な複製を作成します。すべての演算フィールドは演算フィールドとして保持されます。

```
OPEN AR_Customer
EXTRACT RECORD TO "AR_Customer_2"
```

テーブル内のすべてのフィールドを新しいテーブルへ抽出する

AR_Customer テーブル内の定義済みのすべてのフィールドを新しい Analytics テーブルへ抽出するとします。すべての演算フィールドが、物理フィールドに変換され、実際に計算された値が格納されます。

```
OPEN AR_Customer
EXTRACT FIELDS ALL TO "AR_Customer_2"
```

テーブルのすべてのレコードの抽出と、既存のテーブルへの追加

AR_Customer テーブルのすべてのレコードを抽出し、**AR_Customer_Master** テーブルの最後にグループとして追加します。

```
OPEN AR_Customer
EXTRACT RECORD TO "AR_Customer_Master" APPEND
```

テーブルのすべてのレコードの抽出と、別のフォルダーの既存のテーブルへの追加

AR_Customer テーブルのすべてのレコードを抽出し、**AR_Customer_Master** テーブルの最後にグループとして追加します。これは、Analytics プロジェクト フォルダー以外のフォルダーです。

```
OPEN AR_Customer
EXTRACT RECORD TO "C:\Users\Customer Data\AR_Customer_Master" APPEND
```

テーブルから新しいテーブルへフィールドのサブセットを抽出する

次の例では、**AR_Customer** テーブルから 3 つのフィールドを新しい Analytics テーブルへ抽出します。

```
OPEN AR_Customer
EXTRACT FIELDS Name Due Date TO "AR_Customer_Dates.fil"
```

抽出されたフィールドの表示名を作成する

次の例では、**AR_Customer** テーブルから 3 つのフィールドを抽出し、新しい Analytics テーブルにおけるそれらのフィールドの表示名を作成します。

```
OPEN AR_Customer
EXTRACT FIELDS Name AS "Customer;Name" Due AS "Due;Date" Date AS "Invoice;Date" TO
"AR_Customer_Dates.fil"
```

条件に基づいてフィールドを抽出する

次の例では、**Due** フィールドの日付が 2014 年 7 月 1 日より前の日付の場合に、**AR_Customer** テーブルから 3 つのフィールドを新しい Analytics テーブルへ抽出します。

```
OPEN AR_Customer
EXTRACT FIELDS Name Due Date IF Due < `20140701` TO "Overdue.fil"
```

備考

> **メモ**
> このコマンドの動作の詳細については、Analytics のヘルプを参照してください。

テーブルの抽出 (EXTRACT) とテーブルのコピーの比較

EXTRACT では、新しいテーブル レイアウトだけでなく、新しいソース データ ファイル(.fil) も作成されます。

これに対し、**ナビゲーター**(**編集 > コピー**) を使ってテーブルを作成すると、元のソースデータファイルとも関連付けられた新しいテーブルレイアウトが作成されます。この場合、新しいデータファイルが作成されるわけではありません。

FIELDSHIFT コマンド

テーブルレイアウトでフィールド定義の開始位置を変更します。

構文

FIELDSHIFT START *開始位置* COLUMNS *シフトバイト数* <FILTER *データフィルター名*> <OK>

パラメーター

名前	説明
START *開始位置*	シフトする最初のフィールド定義の開始バイト位置。 すべてのフィールド定義が、指定されたフィールド定義の右にシフトされます。 非開始バイト位置を指定する場合は、次の開始バイト位置が使用されます。 **メモ** 非 Unicode 版 Analytics: 1 バイト = 1 文字 Unicode 版 Analytics、拡張 ASCII (ANSI) データ: 1 バイト = 1 文字 Unicode 版 Analytics、Unicode データ: 2 バイト = 1 文字 Unicode データでは、一般的に、奇数で開始するバイト位置を指定してください。偶数の開始位置を指定すると、文字が正しく表示されない可能性があります。
COLUMNS *シフトバイト数*	フィールド定義をシフトするバイト数。 フィールド定義を右にシフトする正の数値を入力します。フィールド定義を左にシフトする負の数値を入力します。 **メモ** 非 Unicode 版 Analytics: 1 バイト = 1 文字 Unicode 版 Analytics、拡張 ASCII (ANSI) データ: 1 バイト = 1 文字 Unicode 版 Analytics、Unicode データ: 2 バイト = 1 文字 Unicode データでは、偶数バイトのみを指定します。奇数バイトを指定すると、文字が正しく表示されない可能性があります。

名前	説明
FILTER *データ フィルター名* 省略可能	特定のレコード定義に関連するフィールド定義を識別するためのフィルターの名前。
OK 省略可能	アクションを確認せずに、項目を削除または上書きします。

例

フィールド定義をシフトする

バイト 11 でフィールド定義と後続のフィールド定義 4 バイトを右にシフトします。

```
FIELDSHIFT START 11 COLUMNS 4
```

備考

> **メモ**
>
> このコマンドの動作の詳細については、Analytics のヘルプを参照してください。

シフトされたフィールド定義は、レコード長内にある必要があります。

1 つ以上のフィールド定義を左右にシフトするときには、フィールドがいずれの方向でもレコード長を超えることができません。

FIELDSHIFT は、指定されたフィールド定義と、任意のフィールド定義の両方を、指定された定義の右に移動します。定義のシフトされたブロックがいずれかの方向でレコード長を超える場合、エラー メッセージが表示され、コマンドは実行されません。

> **ヒント**
>
> レコードの最後を超過しているためにエラー メッセージが表示される場合は、最終フィールド定義を削除し、シフト対象のフィールド定義の場所を作ってください。

FIND コマンド

インデックス付き文字フィールドで、指定した文字列と一致する最初の値を検索します。

> **メモ**
>
> FIND コマンドと FIND( ) 関数は Analytics の 2 つの別個の機能であり、大きな違いがあります。関数の詳細については、"FIND( ) 関数" ページ 557を参照してください。

構文

```
FIND 検索値
```

パラメーター

名前	説明
検索値	検索する文字列。 *検索値*は大文字と小文字を区別し、先頭のスペースを含めることができません。 引用符が検索されるデータの一部でない限り、値を引用符で囲まないでください。

例

特定の値の検索

文字型フィールド **Card_Number** の値のうち、"8590124" と完全に一致するか "8590124" で始まる最初の値を検索したいとします。

まず、**Card_Number** フィールドにインデックスを昇順に作成します。次に、FIND を実行します。これらを実行する例は次のようになります。

```
INDEX ON Card_Number TO "CardNum" OPEN
SET INDEX TO "CardNum"
FIND 8590124
```

備考

> **メモ**
> このコマンドの動作の詳細については、Analytics のヘルプを参照してください。

FIND の用途

FIND コマンドを使用すると、テーブル内で、インデックス付きの文字フィールドに指定した*検索値*を含んでいる最初のレコードに直接移動できます。

INDEX を使用するための要件

このコマンドを使用するには、検索対象のテーブルに、昇順の文字フィールドに基づくインデックスが作成されている必要があります。

複数の昇順の文字フィールドを基にインデックスが作成されている場合は、そのインデックスに指定されている最初のフィールドだけが検索されます。このコマンドで、文字以外のインデックスフィールドや、降順でインデックス付けされた文字フィールドを検索することはできません。

部分一致

部分一致がサポートされます。インデックス付きフィールドに含まれる長い値の一部を検索値に指定できるのです。ただし、検索値は、一致を成すフィールドの先頭に現れる必要があります。

一致に応じた FIND の出力

FIND コマンドの結果は、検索値が見つかったかどうかに応じて、以下のいずれかになります。

- **検索値が見つかった場合** - テーブル内で最初の一致するレコードが選択されます。
- **検索値が見つからない場合** - テーブル内で、検索値よりも大きい値を持つ最初のレコードが、選択されます。

  検索値よりも大きい値がインデックス付きフィールドに存在しない場合は、テーブルは最初のレコードに位置付けられます。どちらの場合も、"キーと一致するインデックスがありません" というメッセージが表示されます。

FIND コマンドは、［**正確な文字比較を行う**］オプション(SET EXACT ON/OFF) の影響を受けません。

FUZZYDUP コマンド

文字フィールドでほぼ同一の値(あいまい重複)を検出します。

> **メモ**
>
> 曖昧一致を使用して、2 つの Analytics テーブルのフィールドを組み合わせて 1 つの新しい Analytics テーブルにする手順については、"FUZZYJOIN コマンド" ページ 207を参照してください。

構文

FUZZYDUP ON *キーフィールド* <OTHER *フィールド*> LEVDISTANCE *値* <DIFFPCT *割合*> <RESULTSIZE *割合*> <EXACT> <IF *テスト*> TO *テーブル名* <LOCAL> <OPEN>

パラメーター

名前	説明
ON *キーフィールド*	あいまい重複をテストする文字フィールドまたは式。
OTHER *フィールド* 省略可能	出力に含めるフィールドや式の一覧。
LEVDISTANCE *値*	2 つの文字列があいまい重複と認定されて結果に含まれるために、それらの文字列間で許容される最大のレーベンシュタイン距離。 LEVDISTANCE の値は 1 未満にすることや、10 を超えることはできません。LEVDISTANCE の値を大きくすると、あいまい度が高い値(相互の関連性が低い値)が含まれるため、結果の件数が多くなります。 詳細については、"FUZZYDUP 動作" ページ 205を参照してください。
DIFFPCT *割合* 省略可能	「相違のパーセント」、つまり、文字列のうち異なっていてもよい割合を制限するしきい値。 あいまい重複の可能性のあるペアが結果に含まれるためには、そのペアに対して実行される Analytics 内部での計算によって出力されるパーセントは、DIFFPCT の値以下である必要があります。DIFFPCT の値は 1 未満にすることや、99 を超えることはできません。 DIFFPCT が省略された場合、しきい値はオフになり、FUZZYDUP コマンドの処理時に相違のパーセントは考慮されません。 詳細については、"FUZZYDUP 動作" ページ 205を参照してください。
RESULTSIZE *割合* 省略可能	キーフィールドのレコード数の割合として、出力結果のセットの最大サイズ。 たとえば、50,000 個の値を持つキーフィールドの場合、RESULTSIZE に 3 を設定すると、結果のあいまい重複数が 1500 を超えた場合(50,000 x 0.03)に処理が終了します。処理が終了した場合、出力テーブルは作成されません。

名前	説明
	RESULTSIZE の値は 1 未満にすることや、1000 パーセントを超えることはできません。1000% の上限は多対多一致の本質に対応するためのものです。多対多一致では、元の検査データセットより大きい結果が生成されるかもしれません。 RESULTSIZE が省略された場合、しきい値はオフになり、FUZZYDUP コマンドの処理時に結果のサイズは考慮されません。 **注意** RESULTSIZE を省略すると、過度に大きな結果のセットが生成され、処理時間が非常に長くなったり、使用可能なメモリの超過を引き起こして、処理が終了したりする可能性があります。結果が管理可能なサイズになると確信している場合にのみ、RESULTSIZE を省略してください。
EXACT 省略可能	あいまい重複だけでなく完全な重複も結果に含まれます。
IF テスト 省略可能	各レコードを処理するために真である必要がある条件式。コマンドは、その条件を満たすレコードに対してのみ実行されます。 **メモ** IF パラメーターは、任意の範囲パラメーター(WHILE、FIRST、NEXT) が適用された後に、テーブルに残るレコードに対してのみ評価されます。
TO *テーブル名*	コマンドの結果を送信する場所: ◦ ***テーブル名*** -は、結果の保存先となる Analytics テーブルのことです。 *テーブル名*には、ファイル拡張子 .FIL を付けた文字列を引用符で囲んで指定する必要があります。例: TO "Output.FIL" デフォルトでは、テーブル データ ファイル(.fil) は、Analytics プロジェクトが入っているフォルダーに保存されます。 既存の異なるフォルダーにデータ ファイルを保存するには、絶対または相対ファイル パスを指定します。 • TO "C:\Output.FIL" • TO "Results\Output.FIL" **メモ** テーブル名は 64 文字の英数字(.FIL 拡張子を含まない) に制限されています。名前にはアンダースコア文字(_) を使用できますが、他の特殊文字やスペースは使用できません。名前の先頭を数字にすることはできません。
LOCAL 省略可能	Analytics プロジェクトと同じ場所に出力ファイルを保存します。 **メモ** Analytics テーブルである出力ファイルを含むサーバー テーブルに対してコマンドを実行するときにのみ適用されます。
OPEN 省略可能	コマンドを実行した後、コマンドによって作成されたテーブルを開きます。コマンドが出力テーブルを作成する場合にのみ有効です。

Analytics の出力変数

名前	含む
GAPDUP*n*	コマンドによって確認されたギャップ、重複、またはあいまい重複グループの合計数。

例

あいまい重複の姓フィールドのテスト

姓フィールドであいまい重複を検査(**Last_Name** フィールドは、`ACL DATA\Sample Data Files\Metaphor_Employee_Data.ACL` の **Employee_List** テーブルにあります)します。結果は新しい Analytics テーブルに出力されます。

- 検査フィールドに加え、他のフィールドも結果に含まれます。
- 許容される最大のレーベンシュタイン距離は 1 です。
- 異なってもよい文字列の割合は 50% に制限されています。
- 結果のサイズは、検査フィールドのサイズの 20% に制限されています。
- あいまい重複に加え、完全な重複も含まれます。

```
FUZZYDUP ON Last_Name OTHER First_Name EmpNo LEVDISTANCE 1 DIFFPCT 50
RESULTSIZE 20 EXACT TO "Fuzzy_Last_Name" OPEN
```

備考

> **メモ**
>
> このコマンドの動作の詳細については、Analytics のヘルプを参照してください。

機能の仕組み

FUZZYDUP コマンドは、ほぼ同一の値(あいまい重複)の検出や、手作業で入力されたデータで一貫性のないつづりを見つけます。

ISFUZZYDUP( ) 関数は、単一の文字値に対するあいまい重複の完全な一覧を識別します。それとは異なり、FUZZYDUP コマンドでは、フィールド内のすべてのあいまい重複を識別し、それらをグループにまとめて、完全でない結果を出力します。

「網羅的でない」とは

「完全でない」とは、結果のあいまい重複の個々のグループには、そのグループ所有者の指定された相違の度合いの範囲内にある、検査フィールドのすべてのあいまい重複が含まれない可能性があることを意味しま

す。ただし、グループ所有者が検査フィールド内の別の値のあいまい重複である場合は、2 つの値が結果のどこかのグループに一緒に現れます。

分析において、検査フィールド内の特定の値に対するあいまい重複の完全な一覧を生成することが重要な場合は、この目的のためにISFUZZYDUP( ) 関数を使用することができます。

FUZZYDUP 動作

FUZZYDUP コマンドには、2 つのパラメーターがあります。これらを指定して、あいまい重複間の相違の度合いや結果のサイズを制御することができます。

- LEVDISTANCE
- DIFFPCT

これらの 2 つのパラメーターの設定の組み合わせをいろいろ試してみて、特定のデータセットでどれが最も良い状態で機能するかを調べる必要があるかもしれません。

LEVDISTANCE (レーベンシュタイン距離)

データの処理中に、FUZZYDUP コマンドは、検査フィールドで評価される文字列のペアごとにレーベンシュタイン距離を計算し、相違のパーセントを求めます。レーベンシュタイン距離は、ある文字列を別の文字列にするために必要な、1 文字の編集の最小回数を示す値です。詳細については、"LEVDIST( ) 関数" ページ 614を参照してください。

DIFFPCT (相違のパーセント)

相違のパーセントは、評価される 2 つの文字列の長さが異なるとき、それらのうち短い方に対する割合であり、2 つの文字列間のレーベンシュタイン距離を使用する、次のような Analytics 内部での計算の結果です。

レーベンシュタイン距離 / 短い文字列内の文字数 × 100 = 相違のパーセント

詳しい情報

あいまい重複のさまざまな設定、結果サイズの制御、およびあいまい重複のグループに関する詳細については、『あいまい重複の概要』を参照してください。

大文字と小文字の区別

FUZZYDUPコマンドでは大文字と小文字が区別されないため、"SMITH" は "smith" と同じであると判断されます。

最後の空白は自動的に削除されます

FUZZYDUP コマンドは、*キーフィールド*内の末尾にあるスペースを自動的に除去するため、*キーフィールド*の単一フィールドを指定するときにTRIM( ) またはALLTRIM( ) 関数を使う必要はありません。

*キーフィールド*のフィールドを連結する場合は、以下のように、ALLTRIM() を使用してください。

FUZZYDUP の効果の改善

フィールドの連結

2 つ以上の検査フィールドを連結すると、検査値の一意性の度合いが増すことによって、FUZZYDUP コマンドの有効性を高めることができます。

例：

```
FUZZYDUP ON ALLTRIM(First_Name)+ALLTRIM(Last_Name) OTHER First_Name Last_Name EmpNo LEVDISTANCE 4 DIFFPCT 50 RESULTSIZE 20 EXACT TO "Fuzzy_First_Name_Last_Name" OPEN
```

OMIT( ) 関数

OMIT( ) 関数もまた、フィールド値から "Corporation" や "Inc." などの総称要素を除去することによって、コマンドの有効性を高めることができます 。

総称要素の除去により、FUZZYDUP の文字列比較は、意味のある違いが発生する可能性のある文字列の部分だけに集中されます。

詳細については、"OMIT( ) 関数" ページ 664を参照してください。

その他の文字列比較方法

- **DICECOEFFICIENT( ) 関数** -は、文字または文字ブロックの相対位置を重視しない、または完全に無視して文字列を比較するための方法を提供します。
- **SOUNDSLIKE( ) 関数および SOUNDEX( ) 関数** -は、正字法の比較 (綴り) ではなく、発音記号の比較 (発音) に基づいて文字列を比較するための方法を提供します。

FUZZYJOIN コマンド

曖昧一致を使用して、2 つの Analytics テーブルのフィールドを組み合わせて 1 つの新しい Analytics テーブルにします。

> **メモ**
>
> 単一の文字フィールドでほぼ同一の値 (あいまい重複) を検出するには、"FUZZYDUP コマンド" ページ 202を参照してください。
>
> 完全に一致するキーフィールド値を使用してテーブルを結合するときのさまざまなオプションについては、"JOIN コマンド" ページ 312を参照してください。

構文

FUZZYJOIN {DICE PERCENT *割合* NGRAM *n-gram 長*|LEVDISTANCE DISTANCE *値*} PKEY *主キーフィールド* SKEY *副キーフィールド* FIELDS *主フィールド*|FIELDS ALL} <WITH *副フィールド*|WITH ALL> <IF *テスト*> <OPEN> <FIRSTMATCH> TO *テーブル名* <WHILE *テスト*> <FIRST *範囲*|NEXT *範囲*> <APPEND>

> **メモ**
>
> サーバーのテーブルに対してローカルで FUZZYJOIN コマンドを実行することはできません。
>
> FUZZYJOIN コマンドはその全体を指定する必要があります。簡略化することはできません。

パラメーター

名前	説明
DICE PERCENT *範囲* NGRAM *n-gram 長* | LEVDISTANCE DISTANCE *値*	使用するあいまい一致アルゴリズム **DICE** - ダイス係数アルゴリズムを使用します ◦ **PERCENT *割合*** - あいまい一致と見なされるための、2 つの文字列の許容可能な最低ダイス係数 0.0000 ～ 1.0000 の小数を指定します(例 : 0.7500) 。小数点 4 桁の最大値を使用します。 値を小さくすると、あいまい度が高い値 (相互の関連性が低い文字列) が含まれるため、一致の件数が多くなります。 ◦ **NGRAM *n-gram 長*** - 使用する *n*-gram の長さ 1 以上の整数を指定します。 *n*-gram の長さを大きくすると、2 つ文字列の間の類似度の基準が厳しくなります。 *n*-gram は、ダイス係数計算の構成要素であり、比較対象となる 2 つの文字列にとって、構成要素であると同時に重なり合う、部分文字列 (文字ブロック) です。

名前	説明
	メモ DICE を指定するときには、FUZZYJOIN コマンドは、IF 文で DICECOEFFICIENT( ) 関数を使用し、キーフィールド値を条件付きで結合します。関数の詳細については、"DICECOEFFICIENT( ) 関数" ページ 530を参照してください。 **LEVDISTANCE** - レーベンシュタイン距離アルゴリズムを使用します ◦ **DISTANCE *値*** -あいまい一致であると見なされるための、2 つの文字列間の許容可能な最低レーベンシュタイン距離 1 以上の整数で指定します。 値を大きくすると、あいまい度が高い値(相互の関連性が低い文字列) が含まれるため、一致の件数が多くなります。 **メモ** LEVDISTANCE を指定するときには、FUZZYJOIN コマンドは、IF 文で LEVDIST( ) 関数を使用し、キーフィールド値を条件付きで結合します。関数の詳細については、"LEVDIST( ) 関数" ページ 614を参照してください。 関数とは異なり、FUZZYJOIN コマンドのレーベンシュタイン距離アルゴリズムは、自動的に先頭と末尾の空白を取り除きます。大文字と小文字は区別されません。
PKEY *主キーフィールド* 主テーブルの文字キーフィールドまたは式。 主キーフィールドは 1 つだけ指定できます。	
SKEY *副キーフィールド* 副テーブルの文字キーフィールドまたは式。 副キーフィールドは 1 つだけ指定できます。	
FIELDS *主フィールド* | FIELDS ALL	結合先の出力テーブルに含める、主テーブル内のフィールドまたは式。 ◦ ***主フィールド*** - 指定した 1 つまたは複数のフィールドが含まれます。 ◦ **ALL** - テーブルのすべてのフィールドを含めます **メモ** 結合テーブルに含める場合は、主キーフィールドを明示的に指定する必要があります。ALL を指定すると含まれます。
WITH *副フィールド* | WITH ALL 省略可能	結合先の出力テーブルに含める、副テーブル内のフィールドまたは式。 ◦ ***副フィールド*** - 指定した 1 つまたは複数のフィールドが含まれます。 ◦ **ALL** - テーブルのすべてのフィールドを含めます **メモ** 結合テーブルに含める場合は、副キーフィールドを明示的に指定する必要があります。ALL を指定すると含まれます。
IF *テスト* 省略可能	各レコードを処理するために真である必要がある条件式。コマンドは、その条件を満たすレコードに対してのみ実行されます。 **メモ** IF パラメーターは、任意の範囲パラメーター(WHILE、FIRST、NEXT) が適用された後に、テーブルに残るレコードに対してのみ評価されます。

名前	説明
	メモ IF 条件は、主テーブル、副テーブル、または両方を参照できます。
OPEN 省略可能	コマンドを実行した後、コマンドによって作成されたテーブルを開きます。コマンドが出力テーブルを作成する場合にのみ有効です。
FIRSTMATCH 省略可能	各主キー値が、最初に発生した副キーと一致にのみ結合されることを指定します。 最初の発生が完全一致であった場合は、後続の主キー値のあいまい一致は、結合された出力テーブルに含まれません。 FIRSTMATCH を省略する場合、FUZZYJOIN のデフォルト動作では、各主キー値をすべての副キーと一致に結合します。 FIRSTMATCH は、完全一致またはあいまい一致の一致が 2 つのテーブル間に存在するかどうかを確認し、すべての一致を特定するために必要な処理時間を減らしたい場合にのみ役立ちます。 各主キー値に対する一致が副テーブルで多くても 1 つだけであることが確実な場合は、FIRSTMATCH を使用することもできます。 **メモ** FIRSTMATCH は、ACLスクリプト パラメーターとしてのみ使用できます。このオプションは、Analytics のユーザー インターフェイスでは使用できません。
TO *テーブル名*	コマンドの結果を送信する場所： ◦ ***テーブル名*** -は、結果の保存先となる Analytics テーブルのことです。 *テーブル名*には、ファイル拡張子 .FIL を付けた文字列を引用符で囲んで指定する必要があります。例：TO "Output.FIL" デフォルトでは、テーブル データ ファイル(.fil) は、Analytics プロジェクトが入っているフォルダーに保存されます。 既存の異なるフォルダーにデータ ファイルを保存するには、絶対または相対ファイル パスを指定します。 • TO "C:\Output.FIL" • TO "Results\Output.FIL" **メモ** テーブル名は 64 文字の英数字(.FIL 拡張子を含まない) に制限されています。名前にはアンダースコア文字(_) を使用できますが、他の特殊文字やスペースは使用できません。名前の先頭を数字にすることはできません。
WHILE *テスト* 省略可能	各レコードを処理するために真である必要がある条件式。条件が false と評価するか、テーブルの最後に達したら、コマンドは実行を中止します。 **メモ** WHILE を FIRST または NEXT とともに使用する場合は、1 つの制限に達するとすぐに、レコードの処理が停止します。
FIRST *範囲* | NEXT *範囲* 省略可能	処理するレコード数： ◦ **FIRST** - 指定されたレコード数に達するまで、最初のレコードから処理を開始します ◦ **NEXT** - 指定されたレコード数に達するまで、現在選択されているレコードから処理を開始し

名前	説明
	ます *範囲*は処理するレコード数を指定します。 FIRST とNEXT を省略すると、すべてのレコードがデフォルトで処理されます。
APPEND 省略可能	コマンドの出力を既存ファイルに上書きしないで、そのファイルの末尾に追加します。 **メモ** コマンドの出力と既存のファイルの構造が同一であるようにする必要があります。 • 同じフィールド • 同じフィールド順序 • 一致するフィールドが同じ長さ • 一致するフィールドが同じデータ型 出力は、既存ファイルとの間でファイル構造が違っている場合でも、Analytics によって既存ファイルに追加されます。出力と既存のファイルの構造が一致しない場合は、データが混在、不足、不正確になります。
ISOLOCALE *ロケールコード* 省略可能	**メモ** Analytics の Unicode 版にのみ適用されます。 システムロケールは「*言語-国*」の形式で入力します。たとえば、カナダフランス語はコード「fr_ca」を入力します。 次のコードを使用します。 ◦ **言語** - ISO 639 標準言語コード ◦ **国** - ISO 3166 標準国コード 国コードを指定しない場合は、言語のデフォルト国が使用されます。 ISOLOCALE を使用しない場合は、デフォルト システム ロケールが使用されます。

例

あいまい一致を使用して、業者である可能性がある従業員を検索するための方法として、2 つのテーブルを結合する

次の例は、共通キーフィールドとして住所(Address および Vendor_Street フィールド) を使用し、Empmast および Vendor テーブルを結合します。

FUZZYJOIN コマンドは、完全一致またはあいまい一致の主および副レコードを含む新しいテーブルを作成します。結果は、同一の住所または類似した住所の従業員と業者のリストです。

FUZZYJOIN とダイス係数アルゴリズム

```
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 とレーベンシュタイン距離アルゴリズム

```
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"
```

すべてのフィールドを含める

このバージョンの FUZZYJOIN コマンドは、主テーブルと副テーブルのすべてのフィールドを結合された出力テーブルに含めます。

```
OPEN Empmast PRIMARY
OPEN Vendor SECONDARY
FUZZYJOIN LEVDISTANCE DISTANCE 5 PKEY Address SKEY Vendor_Street FIELDS ALL WITH
ALL OPEN TO "Employee_Vendor_Match"
```

備考

> **メモ**
> このコマンドの動作の詳細については、Analytics のヘルプを参照してください。

大文字と小文字の区別

FUZZYJOIN コマンドは、使用するあいまい一致アルゴリズムに関係なく、大文字と小文字を区別しません。このため、"SMITH" は "smith" と同じです。

先頭と末尾の空白

FUZZYJOIN コマンドは、使用するあいまい一致アルゴリズムに関係なく、フィールドの先頭と末尾の空白を自動的に取り除きます。主および副キーフィールドを指定するときに、TRIM( ) または ALLTRIM( ) 関数を使用する必要はありません。

GAPS コマンド

Analytics テーブルの数値または日付時刻フィールドに1つ以上のギャップが連続して含まれるかどうかを検出します。

構文

GAPS <ON> *キーフィールド* <D> <UNFORMATTED> <PRESORT> <MISSING *制限*> <HEADER *ヘッダーテキスト*> <FOOTER *フッターテキスト*> <IF *テスト*> <WHILE *テスト*> <FIRST*範囲*|NEXT *範囲*> <TO {SCREEN|*テーブル名*|*ファイル名*|PRINT}> <APPEND> <LOCAL> <OPEN>

パラメーター

名前	説明
ON *キーフィールド*D	欠落を検査するフィールドや式。 キーフィールドを降順に並べ替えるDを含めます。デフォルトのソート順は昇順です。
UNFORMATTED 省略可能	結果をファイルに出力する場合、ページ見出しや改ページは除去されます。
PRESORT 省略可能	コマンドを実行する前にキーフィールドでテーブルを並べ替えます。 **メモ** GROUP コマンドの内部では PRESORT を使用することができません。
MISSING *制限* 省略可能	ギャップの幅ではなく、欠落している個々の項目を出力結果に含めません。 *制限*値は、識別された各ギャップでレポートする欠落項目の最大数を指定します。デフォルト値は5です。制限を超えると、欠落項目はその特定のギャップの範囲としてレポートされます。 *制限*値は、レポートされる欠落項目の総数を制限するものではなく、特定のギャップの範囲内でレポートされる欠落項目の数を制限するだけです。
HEADER *ヘッダーテキスト* 省略可能	レポートの各ページの最上部に挿入されるテキスト。 *ヘッダーテキスト*は引用符で囲んだ文字列として指定する必要があります。この値は、Analytics のシステム変数である HEADER の値よりも優先されます。
FOOTER *フッターテキスト* 省略可能	レポートの各ページの最下部に挿入されるテキスト。 *フッターテキスト*は引用符で囲んだ文字列として指定する必要があります。この値は、Analytics のシステム変数である FOOTER の値よりも優先されます。
IF *テスト* 省略可能	各レコードを処理するために真である必要がある条件式。コマンドは、その条件を満たすレコードに対してのみ実行されます。

名前	説明
	メモ IF パラメーターは、任意の範囲パラメーター(WHILE、FIRST、NEXT) が適用された後に、テーブルに残るレコードに対してのみ評価されます。
WHILE *テスト* 省略可能	各レコードを処理するために真である必要がある条件式。条件が false と評価するか、テーブルの最後に達したら、コマンドは実行を中止します。 **メモ** WHILE を FIRST または NEXT とともに使用する場合は、1 つの制限に達するとすぐに、レコードの処理が停止します。
FIRST *範囲* | NEXT *範囲* 省略可能	処理するレコード数: ◦ **FIRST** - 指定されたレコード数に達するまで、最初のレコードから処理を開始します ◦ **NEXT** - 指定されたレコード数に達するまで、現在選択されているレコードから処理を開始します *範囲*は処理するレコード数を指定します。 FIRST と NEXT を省略すると、すべてのレコードがデフォルトで処理されます。
TO SCREEN | *テーブル名* | *ファイル名* | PRINT 省略可能	コマンドの結果を送信する場所: ◦ **SCREEN** - は Analytics の表示領域に結果を表示します ◦ ***テーブル名*** -は、結果の保存先となる Analytics テーブルのことです。 *テーブル名*には、ファイル拡張子 .FIL を付けた文字列を引用符で囲んで指定する必要があります。例: `TO "Output.FIL"` デフォルトでは、テーブル データ ファイル(.fil) は、Analytics プロジェクトが入っているフォルダーに保存されます。 既存の異なるフォルダーにデータ ファイルを保存するには、絶対または相対ファイル パスを指定します。 • `TO "C:\Output.FIL"` • `TO "Results\Output.FIL"` **メモ** テーブル名は 64 文字の英数字(.FIL 拡張子を含まない) に制限されています。名前にはアンダースコア文字(_) を使用できますが、他の特殊文字やスペースは使用できません。名前の先頭を数字にすることはできません。 ◦ ***ファイル名*** -は結果の保存先となるファイルです。 *ファイル名*には、適切なファイル拡張子を付けた文字列を引用符で囲んで指定します。例: `TO "Output.TXT"` デフォルトでは、テーブル ファイルは、Analytics プロジェクトが入っているフォルダーに保存されます。 既存の異なるフォルダーにファイルを保存するには、絶対または相対ファイル パスを指定します。 • `TO "C:\Output.TXT"` • `TO "Results\Output.TXT"` ◦ **印刷** -- 通常使うプリンターに結果を送信します
APPEND	コマンドの出力を既存ファイルに上書きしないで、そのファイルの末尾に追加します。

名前	説明
省略可能	**メモ** コマンドの出力と既存のファイルの構造が同一であるようにする必要があります。 • 同じフィールド • 同じフィールド順序 • 一致するフィールドが同じ長さ • 一致するフィールドが同じデータ型 出力は、既存ファイルとの間でファイル構造が違っている場合でも、Analyticsによって既存ファイルに追加されます。出力と既存のファイルの構造が一致しない場合は、データが混在、不足、不正確になります。
LOCAL 省略可能	Analyticsプロジェクトと同じ場所に出力ファイルを保存します。 **メモ** Analyticsテーブルである出力ファイルを含むサーバーテーブルに対してコマンドを実行するときにのみ適用されます。
OPEN 省略可能	コマンドを実行した後、コマンドによって作成されたテーブルを開きます。コマンドが出力テーブルを作成する場合にのみ有効です。

Analyticsの出力変数

名前	含む
GAPDUP*n*	コマンドによって確認されたギャップ、重複、またはあいまい重複グループの合計数。

例

欠落した請求書番号がないかどうかをテストする

次の例では、GAPSを使用して、**Invoices**テーブルから欠落している請求書番号がないことを確認しています。

```
OPEN Invoices
GAPS ON Inv_Num PRESORT TO "Invoices_Gaps.fil"
```

備考

文字フィールドに対してGAPSを使用する

数値または日付時刻フィールドをテストする他に、文字フィールドに存在する数値データに対してもギャップがないかどうかをテストすることができます。たとえば、通常、文字データとして書式設定される小切手番号などに対してテストを行うことができます。

文字フィールドに文字と数字が続けて現れる場合は、数字のみが検査され、文字は無視されます。

GROUP コマンド

テーブルの次のレコードに移動する前に、テーブルを 1 回通過するだけで、レコードに対して 1 つ以上の ACLScript コマンドを実行します。コマンド実行は条件によって制御できます。

構文

```
GROUP <IF テスト> <WHILE テスト> <FIRST 範囲|NEXT 範囲>
 コマンド
 <...n>
<ELSE IF テスト>
 コマンド
 <...n>
<ELSE>
 コマンド
 <...n>
END
```

> **メモ**
>
> 一部の Analytics コマンドは、GROUP コマンドとともに使用できません。詳細については、"GROUP コマンドの内部で使用することができるコマンド" ページ 220を参照してください。

パラメーター

名前	説明
IF *テスト* 省略可能	各レコードを処理するために真である必要がある条件式。コマンドは、その条件を満たすレコードに対してのみ実行されます。 **メモ** IF パラメーターは、任意の範囲パラメーター(WHILE、FIRST、NEXT) が適用された後に、テーブルに残るレコードに対してのみ評価されます。
WHILE *テスト* 省略可能	各レコードを処理するために真である必要がある条件式。条件が false と評価するか、テーブルの最後に達したら、コマンドは実行を中止します。 **メモ** WHILE を FIRST または NEXT とともに使用する場合は、1 つの制限に達するとすぐに、レコードの処理が停止します。
FIRST *範囲* | NEXT *範囲* 省略可能	処理するレコード数: ◦ **FIRST** - 指定されたレコード数に達するまで、最初のレコードから処理を開始します ◦ **NEXT** - 指定されたレコード数に達するまで、現在選択されているレコードから処理を開始

名前	説明
	します *範囲*は処理するレコード数を指定します。 FIRST とNEXT を省略すると、すべてのレコードがデフォルトで処理されます。
コマンド <...n>	GROUP 内で実行する 1 つまたは複数の ACLScript コマンド。GROUP 内で使用できるコマンドの詳細な一覧については、"GROUP コマンドの内部で使用することができるコマンド" ページ 220を参照してください。 これらの各コマンドの前に IF または ELSE IF がある場合には、テストの評価結果が true になる必要があります。 これらの各コマンドが ELSE より後に記述されている場合は、そのコマンドより前にあるどのコマンドでも処理されていないレコードが存在するときに、そのコマンドが実行されます。各コマンドは別々の行で開始して、複数のコマンドを含めることができます。
ELSE IF *テスト* 省略可能	GROUP コマンドの ELSE IF ブロックを開きます。この条件は、GROUP コマンド テスト、およびその条件より前にあるどの ELSE IF テストにも適合しなかったレコードをテストするものです。 複数の ELSE IF テストを含めることができ、それらは上から下へ、レコードが True と評価されて、その ELSE IF ステートメントに続くコマンドが実行されるまで評価されます。
ELSE 省略可能	GROUP コマンドの ELSE ブロックを開きます。このブロックより前にあるすべてのテストが False と評価されたレコードに対し、このブロックに入力したコマンドが実行されます。
END	GROUP コマンドの終わり。

例

単純な GROUP

単純なグループは、GROUP コマンドで始まり、その後に一連のコマンド群が続き、END コマンドで終わります。

```
GROUP
 COUNT
 HISTOGRAM ON Quantity MINIMUM 0 MAXIMUM 100 INTERVALS 10
 CLASSIFY ON Location SUBTOTAL Quantity
END
```

GROUP IF

条件付きグループでは、条件が True または False のどちらであるかに基づいてコマンドを実行します。たとえば、次の GROUP コマンドは、**Product_class** フィールドの値が 5 未満であるレコードに対してのみ実行されます。

```
GROUP IF Product_class < "05"
 COUNT
 HISTOGRAM ON Quantity MINIMUM 0 MAXIMUM 100 INTERVALS 10
 CLASSIFY ON Location SUBTOTAL Quantity
END
```

GROUP IF ...ELSE

ELSE ブロックを指定しない限り、条件を満たしていないレコードは無視されます。

ELSE ステートメントに続くコマンドの数に制限はありません。次の例では、条件を満たしていないレコードすべての **Quantity** フィールドを合計します。

```
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

ELSE IF ブロックは、各ブロックが別々の条件式であれば、グループ内に複数指定できます。次の例では、ELSE IF および ELSE ブロックは 4 つの合計を生成します。

```
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
```

入れ子の GROUP コマンド

入れ子のグループとは、別のグループの中に入っているグループのことをいいます。入れ子のグループを使用すると、レコードに対してどのコマンドを実行するかを強力に制御できます。ほとんどのアプリケーションではこうした高度な機能は必要ありませんが、必要に応じて使用することができます。

ほかの種類のグループと同様、入れ子のグループの終了にはENDコマンドを使用します。Analyticsのデータ処理は、すべてのグループコマンドが終了してから開始されます。

```
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
```

この例では、**Product_class** が05未満である場合にのみ、COUNTから次のGROUPまでに含まれるコマンドすべてが実行されます。

STATISTICSコマンドとHISTOGRAMコマンドは、**Quantity** が0より大きい場合にのみ実行されます。ただし、2番目のGROUPコマンドは入れ子になっているため、STATISTICSコマンドとHISTOGRAMコマンドは、**Product_class** < "05" の条件と**Quantity** > 0の条件の両方を満たすレコードに対してのみ実行されます。

GROUPブロック内でシステム変数を生成する

1つのGROUPコマンドで、複数のシステム変数を作成できます。

通常、TOTAL、COUNT、STATISTICSなどのコマンドを実行すると、ただ1つのシステム変数が生成されます。コマンドを実行するたびに、前回のコマンド実行によって生成されたシステム変数の値は上書きされます。GROUPブロック内で実行されるコマンドのインスタンスごとに、固有の変数が作成されます。

次の例では、TOTALコマンドにより、**Metaphor_Trans_2002** テーブル内の各製品クラスの**Amount** フィールドの合計が計算されます。下記のコードを実行すると、以下の変数が生成され、GROUPブロックの後続のコマンド内で使用できるようになります。

- **TOTAL2** - 製品クラス03の**Amount** フィールドの合計
- **TOTAL3** - 製品クラス05の**Amount** フィールドの合計
- **TOTAL4** - 製品クラス08の**Amount** フィールドの合計
- **TOTAL5** - 製品クラス09の**Amount** フィールドの合計

```
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
```

備考

> **ヒント**
>
> GROUP および LOOP コマンドの詳細なチュートリアルについては、"グループ化とループ処理" ページ 32を参照してください。

GROUP コマンドの内部で使用することができるコマンド

以下の表は Analytics コマンドの内部で使用することができるコマンドを示しています。

コマンドが以下の一覧にない場合は、GROUP 内で使用できません。

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		

グループ化とループ処理

GROUP コマンドを使用すると、テーブル内の次のレコードに移動する前にレコードにいくつかのコマンドを実行することができます。これにより、処理時間を大幅に減らすことができます。

レコードに対して一連のコマンドを複数回実行する必要がある場合には、GROUP コマンド内で LOOP コマンドを使用することができます。

GROUP での変数の使用

ユーザー定義変数

GROUP コマンド内で変数を使用するには、その GROUP ブロックを入力する前に、その変数を定義しておきます。

> **メモ**
>
> GROUP のブロック内で変数を定義して初期化することもできますが、お勧めしません。GROUP 内で初期化された変数を使用すると、予期しない結果が生成される場合があります。

グループで変数を評価するには、代入変数を使用します。変数の値は、GROUP 内に入る前の値から変わらなくなります。

GROUP 内で変数を定義した場合に、代入変数を使ってその変数を参照することはできません。

```
ASSIGN v_test = "hello"
GROUP
 ASSIGN v_test2 = "%v_test% world"
 COMMENT これは無効です: v_test3 = "%v_test2% again"
END
```

システム定義変数

TOTAL やSTATISTICS など特定のコマンドでは、実行する計算に基づくシステム変数を生成します。これらのコマンドを実行するのにGROUP を使用すると、生成されるすべてのシステム変数には連続番号が付けられます。この連続番号は、GROUP ブロック内のコマンドの行番号で始まり、*n* で終わります(空行は除く) 。*n* の値は GROUP ブロック内の行番号 1 つごとに 1 ずつ増加します。

> **メモ**
>
> GROUP ブロック内で作成されたシステム生成変数を使用するには、GROUP ブロックが完了するまで待つ必要があります。変数が使用可能になるには、コマンドがテーブル内のすべてのレコードに対して実行される必要があります。これらの変数は、GROUP ブロックを終了する END キーワードの後に使用してください。

次の例では、最初のTOTAL コマンドにより変数 TOTAL2 が生成され、2 番目のTOTAL コマンドにより変数 TOTAL4 が生成されます。これらの変数は両方とも、GROUP ブロックの完了後に後続のコマンドで使用できます。

```
GROUP
 TOTAL Discount IF Order_Priority = "Low"
 ASSIGN v_var = "test"
 TOTAL Discount IF Order_Priority = "High"
END
```

構文に関するメモ

- GROUP コマンドには複数行の構文の記述が必要なため、コマンド ラインでこのコマンドを入力することはできません。
- 各 GROUP コマンドは END コマンドで終了する必要があります。
- スクリプト内で GROUP コマンドを使用するときは、グループ内に記載されたコマンドをインデントすることにより、コマンド ブロックの読みやすさを向上させることができます。

HELP コマンド

Analytics ヘルプ文 書をブラウザーで起 動します。

構 文

```
HELP
```

HISTOGRAM コマンド

文字フィールドまたは数値フィールドの値に基づいてレコードをグループ化し、各グループ内のレコード数をカウントして、グループとカウントを棒グラフで表します。

構文

HISTOGRAM {<ON> *文字フィールド*|<ON> *数値フィールド* MINIMUM *値* MAXIMUM *値* {<INTERVALS *数値*>|FREE *間隔値 <...n> 最後の間隔*}} <TO SCREEN|*ファイル名* |GRAPH|PRINT}> <IF *テスト*> <WHILE *テスト*> <FIRST *範囲*|NEXT *範囲*> <HEADER *ヘッダーテキスト*> <FOOTER *フッターテキスト*> <KEY *内訳フィールド*> <SUPPRESS> <COLUMNS *数値*> <APPEND> <LOCAL> <OPEN>

パラメーター

名前	説明
ON *文字フィールド*	ヒストグラムに使用する文字フィールドまたは式。
ON *数値フィールド*	ヒストグラムに使用する数値フィールドまたは式。
MINIMUM *値*	数値型のフィールドにのみ適用されます。最初の数値間隔の最小値。 FREE を使用している場合、MINIMUM は省略可能です。それ以外の場合は必要になります。
MAXIMUM *値*	数値型のフィールドにのみ適用されます。最後の数値間隔の最大値。 FREE を使用している場合、MAXIMUM は省略可能です。それ以外の場合は必要になります。
INTERVALS *数* 省略可能	数値型のフィールドにのみ適用されます。 MINIMUM 値と MAXIMUM 値によって指定された範囲の間に Analytics が生成する、均等な間隔の数。間隔の数を指定しない場合は、デフォルトの数が使用されます。 デフォルトは、[**オプション**]ダイアログボックスの [**コマンド**]タブの [**間隔の数**]によって決定されます。
FREE *間隔値 <...n> 最終間隔* 省略可能	数値型のフィールドにのみ適用されます。 各間隔の開始点と最後の間隔の終了点を指定することにより、カスタムサイズの間隔を作成することができます。 MINIMUM 値と MAXIMUM 値を指定した場合は、これらの値がそれぞれ最初の間隔の開始点と最後の間隔の終了点となり、各 *間隔値* が範囲内に追加の間隔を生成します。指定する間隔値は、MINIMUM 値より大きく、かつ MAXIMUM 値以下である必要があります。 間隔値は、数値順でなければならず、重複値を含めることはできません。次に例を示します。

名前	説明
	たとえば、FREE -1000, 0, 1000, 2000, 3000 のように指定します。 FREE とINTERVALS の両方を指定する場合は、INTERVALS が無視されます。
TO SCREEN | *ファイル名* | GRAPH | PRINT	コマンドの結果を送信する場所： ◦ **SCREEN** - はAnalytics の表示領域に結果を表示します ◦ ***ファイル名*** -は結果の保存先となるファイルです。 *ファイル名*には、適切なファイル拡張子を付けた文字列を引用符で囲んで指定します。 例：TO "Output.TXT" デフォルトでは、テーブル ファイルは、Analytics プロジェクトが入っているフォルダーに保存されます。 既存の異なるフォルダーにファイルを保存するには、絶対または相対ファイル パスを指定します。 • TO "C:\Output.TXT" • TO "Results\Output.TXT" ◦ **GRAPH** - は結果をグラフに表示し、それを Analytics の表示領域に表示します ◦ **印刷** -- 通常使うプリンターに結果を送信します **メモ** ファイルに出力されたヒストグラムの結果は、棒グラフのテキスト表現として表示されます。
IF *テスト* 省略可能	各レコードを処理するために真である必要がある条件式。コマンドは、その条件を満たすレコードに対してのみ実行されます。 **メモ** IF パラメーターは、任意の範囲パラメーター(WHILE、FIRST、NEXT) が適用された後に、テーブルに残るレコードに対してのみ評価されます。
WHILE *テスト* 省略可能	各レコードを処理するために真である必要がある条件式。条件が false と評価するか、テーブルの最後に達したら、コマンドは実行を中止します。 **メモ** WHILE を FIRST または NEXT とともに使用する場合は、1 つの制限に達するとすぐに、レコードの処理が停止します。
FIRST *範囲* | NEXT *範囲* 省略可能	処理するレコード数： ◦ **FIRST** - 指定されたレコード数に達するまで、最初のレコードから処理を開始します ◦ **NEXT** - 指定されたレコード数に達するまで、現在選択されているレコードから処理を開始します *範囲*は処理するレコード数を指定します。 FIRST とNEXT を省略すると、すべてのレコードがデフォルトで処理されます。
HEADER *ヘッダーテキスト* 省略可能	レポートの各ページの最上部に挿入されるテキスト。 *ヘッダーテキスト*は引用符で囲んだ文字列として指定する必要があります。この値は、Analytics のシステム変数である HEADER の値よりも優先されます。

名前	説明
FOOTER *フッターテキスト* 省略可能	レポートの各ページの最下部に挿入されるテキスト。 *フッターテキスト*は引用符で囲んだ文字列として指定する必要があります。この値は、Analytics のシステム変数である FOOTER の値よりも優先されます。
KEY *ブレークフィールド* 省略可能	小計計算をグループ化するフィールドまたは式。*ブレークフィールド*の値が変わるたびに、小計が計算されます。 *ブレークフィールド*は、文字フィールドか式である必要があります。指定できるフィールドは 1 つだけですが、1 つ以上のフィールドを含んでいる式を使用することができます。
SUPPRESS 省略可能	MAXIMUM 値より大きい値と MINIMUM 値より小さい値をコマンド出力から除外します。
COLUMNS *数* 省略可能	ヒストグラムの結果をテキスト ファイルに出力する場合は、棒グラフをテキストで表示するときの x 軸の長さを指定します。 数の値は、x 軸(および y 軸のラベル) に使用する文字スペース(テキスト列) の数です。COLUMNS を省略した場合は、デフォルトの文字スペース数の 78 が使用されます。
APPEND 省略可能	コマンドの出力を既存ファイルに上書きしないで、そのファイルの末尾に追加します。 **メモ** コマンドの出力と既存のファイルの構造が同一であるようにする必要があります。 • 同じフィールド • 同じフィールド順序 • 一致するフィールドが同じ長さ • 一致するフィールドが同じデータ型 出力は、既存ファイルとの間でファイル構造が違っている場合でも、Analytics によって既存ファイルに追加されます。出力と既存のファイルの構造が一致しない場合は、データが混在、不足、不正確になります。
LOCAL 省略可能	Analytics プロジェクトと同じ場所に出力ファイルを保存します。 **メモ** Analytics テーブルである出力ファイルを含むサーバー テーブルに対してコマンドを実行するときにのみ適用されます。
OPEN 省略可能	コマンドを実行した後、コマンドによって作成されたテーブルを開きます。コマンドが出力テーブルを作成する場合にのみ有効です。

例

時給の基本的なヒストグラム

次の例は、HISTOGRAM を使用して、時給 0～100 ドルの間の賃金の分布を示すグラフを作成しています。

```
HISTOGRAM ON Rate MINIMUM 0 MAXIMUM 100 TO GRAPH
```

時給用に定義された間隔を使用したヒストグラム

直前の例に続けてHISTOGRAMを使用し、直前の例より意味のあるやり方でグラフの範囲を指定します。

賃金の大部分は時給20～50ドルの範囲に含まれているため、グラフに含まれる間隔の数は以下のとおりです。

- 20～50の範囲に3つ
- 0～20に1つ
- 50～100に1つ
- 100より大きい範囲に1つ

```
HISTOGRAM ON Rate MINIMUM 0 MAXIMUM 100 FREE 20,30,40,50,100 TO GRAPH
```

備考

> **メモ**
> このコマンドの動作の詳細については、Analytics のヘルプを参照してください。

最小値と最大値を設定する

HISTOGRAMコマンドを実行する前に、数値フィールドでSTATISTICSコマンドまたはPROFILEコマンドを実行して、フィールド内の最小値と最大値をMINIMUMパラメーターとMAXIMUMパラメーターに自動的に設定することができます。

関連コマンド

文字フィールドを使用してヒストグラムを作成するのは、分類に似ています。数値フィールドを使用してヒストグラムを作成するのは、階層化に似ています。

Analytics内の他のグループ化操作とは異なり、ヒストグラムは数値フィールドの小計をサポートしていません。

IF コマンド

コマンドを実行するためにtrueと評価する必要のある条件を指定します。

構文

```
IF テスト コマンド
```

パラメーター

名前	説明
検査	コマンドを実行するために満たす必要のある条件。
コマンド	テストの評価結果がtrueの場合に実行する、任意の有効なACLScriptコマンド。

例

スクリプトを条件付きで実行する

*v_counter*変数が10より大きい場合にのみ、テーブルに対してCLASSIFYを実行するには、次のように指定します。

```
IF v_counter > 10 CLASSIFY ON Location TO "Count_by_Location.fil" OPEN
```

ユーザーの決定に基づいてコマンドを実行する

テーブルを分類化するかどうかをスクリプトのユーザーに決定させたいとします。

スクリプトで、選択した場合には、CLASSIFYを実行できるというチェックボックスとともに、ダイアログボックスを含めます。チェックボックスには、論理変数 *v_classify_checkbox* にTrueまたはFalse入力値が保存されます。

IFテストを使用して、*v_classify_checkbox*の値を決定します。値がTrueの場合は、CLASSIFYは次を実行します。

```
IF v_classify_checkbox=T CLASSIFY ON Location TO "Count_by_Location.fil" OPEN
```

備考

IF コマンドと IF パラメーターの比較

IF コマンドのロジックは、ほとんどのコマンドでサポートされている IF パラメーターとは異なります。

- **IF コマンド：** -テスト式の値に基づいて、関連付けられたコマンドを実行するかどうかを判断します。
- **IF パラメーター**-: Analytics テーブル内の各レコードに対し、テスト式の値に基づいて、コマンドを実行するかどうかを判断します。

スクリプト内で判断を行う

スクリプトでは、一連の IF コマンド条件を入力して、その結果を基に異なるコマンドを実行できます。さらに処理を行うかどうかを判断するために、IF コマンドを使用して変数の値をテストすることができます。

IMPORT ACCESS コマンド

Microsoft Access データベースファイルを定義およびインポートして、Analytics テーブルを作成します。

構文

IMPORT ACCESS TO *テーブル*<PASSWORD *数値*> *インポートファイル名* FROM *ソースファイル名* TABLE *入力テーブル名* CHARMAX *最大フィールド長* MEMOMAX *最大フィールド長*

パラメーター

名前	説明
TO *テーブル*	データをインポートする Analytics テーブルの名前。 **メモ** テーブルの名前は、64 文字までの英数字に制限されます。名前にはアンダースコア文字 (_) を使用できますが、他の特殊文字やスペースは使用できません。名前の先頭を数字にすることはできません。
PASSWORD *番号* 省略可能	パスワード保護された Access ファイルでのみ使用します。 使用するパスワード定義。 実際のパスワードを入力することを求めたり指定したりするのには、PASSWORD *番号*の構文は使用しません。パスワード定義とは、以前に PASSWORD コマンドか SET PASSWORD コマンドを使用して入力または設定されたパスワードのことです。 *番号*はパスワード定義の番号です。たとえば、以前に 2 つのパスワードをスクリプトで設定したり入力したりしている場合には、PASSWORD 2 により、2 番目のパスワードを使用することを指定するなどします。 パスワードの入力または設定の詳細については、次を参照してください。 • "PASSWORD コマンド" ページ 345 • "SET コマンド" ページ 404 • PASSWORD アナリティクス タグ パスワードの入力または設定の詳細については、次を参照してください。 • PASSWORD コマンド • SET コマンド • PASSWORD アナリティクス タグ
インポートファイル名	作成する Analytics データファイルの名前。 *インポートファイル名*、.FIL ファイル拡張子を持つ引用符で囲まれた文字列として指定します。例 : "Invoices.FIL". デフォルトでは、データファイル(.FIL) は、Analytics プロジェクトが入っているフォルダーに保存されます。

名前	説明
	既存の異なるフォルダーにデータファイルを保存するには、絶対または相対ファイルパスを指定します。 ◦ "C:\data\Invoices.FIL" ◦ "data\Invoices.FIL"
FROM *ソースファイル名*	ソースデータファイルの名前。*ソースファイル名*は引用符で囲む必要があります。 ソースデータファイルがAnalyticsプロジェクトと同じフォルダーに位置しない場合、ファイルの位置を指定するために絶対パスまたは相対パスを使用する必要があります。 ◦ "C:\data*ソースファイル名*" ◦ "data*ソースファイル名*"
TABLE *入力テーブル名*	インポートするMicrosoft Accessデータベースファイルのテーブルの名前。
CHARMAX *最大フィールド長*	インポートしているソースの文字データから発生するAnalyticsテーブルの任意のフィールドの文字の最大長。 1 ～ 255文字を指定できます。
MEMOMAX *最大フィールド長*	インポートするテキスト、ノート、またはメモフィールドの文字の最大長。 1 ～ 32767文字(非Unicode版のAnalytics)、または16383文字(Unicode版のAnalytics)を指定できます。

例

> **メモ**
> このコマンドの動作の詳細については、Analyticsのヘルプを参照してください。

テーブルへのインポート

Microsoft Accessファイル`Acceptable_Codes.mdb`があります。**[Acceptable_Codes]**テーブルをAnalyticsからインポートする必要があります。このためには、次のコマンドを使用して、**acc_codes**テーブルをAnalyticsで作成します。

インポートされた文字またはメモフィールドの長さが、フィールドの最も長い値の長さまたは指定された文字の最大数の短い方に設定されます。

```
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 コマンド

区切り文字付きテキスト ファイルを定義およびインポートして、Analytics テーブルを作成します。

構文

IMPORT DELIMITED TO *テーブル インポート ファイル名* FROM *ソース ファイル名* <SERVER *プロファイル名*> *ソースの文字エンコード* SEPARATOR {*文字*|TAB|SPACE} QUALIFIER {*文字*|NONE} <CONSECUTIVE> STARTLINE *行番号* <KEEPTITLE> <CRCLEAR> <LFCLEAR> <REPLACENULL> <ALLCHAR> {ALLFIELDS | [*フィールド構文*] <...*n*> <IGNORE *フィールド番号*> <...*n*>}

フィールド構文 ::=
FIELD *名前 型* AT *開始位置* DEC *値* WID *バイト* PIC *書式* AS *表示名*

パラメーター

名前	説明
TO *テーブル*	データをインポートする Analytics テーブルの名前。 **メモ** テーブルの名前は、64 文字までの英数字に制限されます。名前にはアンダースコア文字 (_) を使用できますが、他の特殊文字やスペースは使用できません。名前の先頭を数字にすることはできません。
インポート ファイル名	作成する Analytics データ ファイルの名前。 *インポート ファイル名*、.FIL ファイル拡張子を持つ引用符で囲まれた文字列として指定します。例: "Invoices.FIL". デフォルトでは、データ ファイル(.FIL) は、Analytics プロジェクトが入っているフォルダーに保存されます。 既存の異なるフォルダーにデータ ファイルを保存するには、絶対または相対ファイル パスを指定します。 ◦ "C:\data\Invoices.FIL" ◦ "data\Invoices.FIL"
FROM *ソースファイル名*	ソース データ ファイルの名前。*ソース ファイル名*は引用符で囲む必要があります。 ソース データ ファイルが Analytics プロジェクトと同じフォルダーに位置しない場合、ファイルの位置を指定するために絶対パスまたは相対パスを使用する必要があります。 ◦ "C:\data*ソース ファイル名*" ◦ "data*ソース ファイル名*"

<table>
<tr><th>名前</th><th>説明</th></tr>
<tr><td>SERVER プロファイル名
省略可能</td><td>インポートしたいデータが置かれている AX Server のサーバー プロファイル名。</td></tr>
<tr><td>ソースの文字エンコード</td><td>ソース データの文字セットおよび文字エンコード。
お使いの Analytics エディションとソース データのエンコードに応じて、次のうち該当するコードを指定してください。
<table>
<tr><th>コード</th><th>Analytics のエディション</th><th>ソース データのエンコード</th></tr>
<tr><td>0</td><td>非 Unicode 版</td><td>すべてのデータ</td></tr>
<tr><td>0</td><td>Unicode 版</td><td>ASCII データ</td></tr>
<tr><td>2</td><td>Unicode 版</td><td>Unicode データ、UTF-16 LE エンコード</td></tr>
<tr><td>3 数値コード</td><td>Unicode 版</td><td>Unicode データ、UTF-16 LE エンコード
ソース データのエンコに適合する数値コードを決定するには、データ定義ウィザードを使用してインポートを実行し、 [エンコードされたテキスト]オプションを選択し、付属するドロップダウンリストを使って適合するエンコードを見つけます。
コードを指定するには、3、1 つのスペース、数値コードを順に指定します。</td></tr>
</table></td></tr>
<tr><td>SEPARATOR 文字 | TAB | SPACE</td><td>ソース データのフィールド間で使用される区切り文字。文字は引用符で囲まれた文字列として指定する必要があります。
タブまたは空白を区切り文字として指定するには、それらを二重引用符で囲んで入力するか、またはそれらのキーワードを使用します。
◦ SEPARATOR " " または SEPARATOR SPACE
◦ SEPARATOR "," または SEPARATOR SPACE</td></tr>
<tr><td>QUALIFIER 文字 | NONE</td><td>ソース データ内のフィールド値を折り返すため と識別するために使用するテキスト修飾子文字。文字は引用符で囲まれた文字列として指定する必要があります。
文字をテキスト修飾子として指定するには、二重引用符を一重引用符で囲む必要があります。QUALIFIER '"'
テキスト修飾子がないことを指定するには、次のいずれかの方法を使用します。
◦ QUALIFIER ""
◦ QUALIFIER NONE</td></tr>
<tr><td>CONSECUTIVE
省略可能</td><td>連続したテキスト修飾子を単一の修飾子として扱います。</td></tr>
<tr><td>STARTLINE 行番号</td><td>データが始まる行。
たとえば、データの最初の 4 行にヘッダー情報が含まれており、それらをインポートの対象外とす</td></tr>
</table>

名前	説明
	る場合は、行番号「*5*」を指定します。
KEEPTITLE 省略可能	STARTLINE に指定する行番号をデータでなくフィールド名と見なします。KEEPTITLE を省略すると、汎用フィールド名が使用されます。 FIELD 構文を個別に指定する場合は、FIELD の直後に指定した*名前*が区切り文字付きファイル内の最初の行内の値より優先されます。このような場合に KEEPTITLE を使用することで、最初の行内の値がインポートされないようにすることができます。
CRCLEAR 省略可能	テキスト修飾子間で発生するすべての CR 文字(キャリッジリターン)をスペース文字で置換します。CRCLEAR を使用するには、QUALIFIER と文字値を指定する必要があります。 CRCLEAR と LFCLEAR の両方を使用する場合は、CRCLEAR を最初に記述する必要があります。
LFCLEAR 省略可能	テキスト修飾子間で発生するすべての LF 文字(ラインフィード)をスペース文字で置換します。LFCLEAR を使用するには、QUALIFIER と文字値を指定する必要があります。 CRCLEAR と LFCLEAR の両方を使用する場合は、CRCLEAR を最初に記述する必要があります。
REPLACENULL 省略可能	スペースで区切られたファイルで発生するすべての NUL 文字を置換します。置き換えられたすべての NULL 文字の数がログに記録されます。
ALLCHAR 省略可能	インポートされたすべてのフィールドには、自動的に文字のデータ型が割り当てられます。 **ヒント** インポートされたすべてのフィールドに文字のデータ型を割り当てると、区切り文字付きテキスト ファイルのインポート処理が容易になります。Analytics にインポートされたデータのフィールドには、数値や日付時刻などのさまざまなデータ型を割り当て、書式の詳細を指定することができます。 Analytics によって識別子のフィールドに数値のデータ型が自動的に割り当てられたテーブルをインポートする際、実際には文字のデータ型を使用する必要がある場合には、ALLCHAR を使用することができます。
ALLFIELDS	ソース データ ファイルの全フィールドがインポートされます。 ALLFIELDS を使用する際に Analytics によってデータ型が割り当てられる方法の詳細については、"備考" ページ 236を参照してください。 **メモ** ALLFIELDS を指定する場合、個別の FIELD 構文や IGNORE 指定しないでください。
FIELD *名前 型*	インポートするソース データ ファイル内の個別フィールドの名前およびデータ型。フィールドをインポート対象から除外する場合は、そのフィールドを指定しないでください。 *型*については、"フィールド データ型の識別子" ページ 237を参照してください。 **メモ** ALLCHAR を指定した場合には、*型*は無視されます。
AT *開始位置*	Analytics データ ファイル内のフィールドの開始バイトを指定します。

名前	説明
	メモ 非 Unicode 版 Analytics: 1 バイト = 1 文字 Unicode 版 Analytics: 2 バイト = 1 文字 Unicode 版 Analytics では、一般的に、奇数で開始するバイト位置を指定してください。偶数の開始位置を指定すると、文字が正しく表示されない可能性があります。
DEC *値*	数値フィールドの小数点以下の桁数 **メモ** DEC を指定した場合には、PIC は無視されます。
WID *バイト*	Analytics テーブルレイアウトにおけるフィールドの長さ(バイト数) **メモ** 非 Unicode 版 Analytics: 1 バイト = 1 文字 Unicode 版 Analytics: 2 バイト = 1 文字 Unicode 版 Analytics では、偶数バイトのみを指定します。奇数バイトを指定すると、文字が正しく表示されない可能性があります。
PIC *書式*	**メモ** 数値フィールドまたは日付時刻フィールドにのみ適用されます。 ◦ **数値フィールド** - Analytics のビューとレポートに含まれる数値の表示形式。 ◦ **日付時刻フィールド** -ソースデータの日付時刻値の物理形式(日付時刻文字、区切り文字の順など) **メモ** 日付時刻フィールドの場合、*形式*はソースデータの物理形式と正確に一致する必要があります。たとえば、ソースデータが 12/31/2014 である場合は、書式を "MM/DD/YYYY" として入力します。 *書式*は引用符で囲む必要があります。 **メモ** ALLCHAR を指定した場合には、PIC は無視されます。
AS *表示名*	新しい Analytics テーブルのビューにおけるフィールドの表示名(代替列見出し)。 *表示名*の値は引用符で囲まれた文字列。列見出しを改行したい場合は、語句の間にセミコロン(;) を入れます。 フィールドの定義時には、AS は必須です。表示名をフィールド名と同じにしたい場合は、空の表示名を入力します。つまり、次の構文を使用します。AS "" 2 つの二重引用符の間にスペースがないことを確認してください。
IGNORE *フィールド番号* <...*n*>	テーブルレイアウトからフィールドを除外します。

名前	説明
省略可能	*フィールド番号*は、ソース データにおけるフィールドの位置を指定します。たとえば、IGNORE 5 は、ソース データの 5 番目のフィールドを Analytics テーブル レイアウトから除外します。 **メモ** このフィールドのデータは、インポートは行われますが、定義されないため、新しい Analytics テーブルには表示されません。必要に応じ、データを後で定義して、テーブルに追加することができます。 特定のフィールドをインポート対象から完全に除外するには、個別に各フィールドを指定する際にそのフィールドを指定しないでください。

例

すべてのフィールドをインポートする

カンマ区切りファイルから **Employees** という名前の Analytics テーブルにすべてのフィールドをインポートするとします。このファイルでは、テキスト修飾子として二重引用符が使用されています。データ型は、"備考" 次のページに示した規則セットに基づいて自動的に割り当てられます。

```
IMPORT DELIMITED TO Employees "Employees.fil" FROM "Employees.csv" 0 SEPARATOR "," QUALIFIER "" CONSECUTIVE STARTLINE 1 KEEPTITLE ALLFIELDS
```

すべてのフィールドをインポートし、文字のデータ型を自動的に割り当てる

カンマ区切りファイルから **Employees** という名前の Analytics テーブルにすべてのフィールドをインポートするとします。このファイルでは、テキスト修飾子として二重引用符が使用されています。インポートされたすべてのフィールドには、自動的に文字のデータ型が割り当てられます。

```
IMPORT DELIMITED TO Employees "Employees.fil" FROM "Employees.csv" 0 SEPARATOR "," QUALIFIER "" CONSECUTIVE STARTLINE 1 KEEPTITLE ALLFIELDS
```

指定したフィールドをインポートし、文字のデータ型を自動的に割り当てる

タブ区切りファイルから **Employees** という名前の Analytics テーブルに指定のフィールドをインポートするとします。このファイルでは、テキスト修飾子として二重引用符が使用されています。インポートされたすべてのフィールドには、自動的に文字のデータ型が割り当てられます。

```
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"
```

指定したフィールドをインポートし、フィールドごとにデータ型を割り当てる

セミコロン区切りファイルから **Employees** という名前の Analytics テーブルに指定のフィールドをインポートするとします。このファイルではテキスト修飾子は使用されていません。インポートするフィールドごとにデータ型を指定します。

```
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"
```

備考

> **メモ**
>
> このコマンドの動作の詳細については、Analytics のヘルプを参照してください。

ALLFIELDS を使用する際に Analytics によってデータ型が割り当てられる仕組み

個別にフィールドを定義する代わりに ALLFIELDS パラメーターを使用すると、区切り文字付きファイルの先頭にあるレコードのサブセットが Analytics によって調べられ、次に示す規則セットに基づいてデータ型が割り当てられます。

Analytics にインポートされたデータのフィールドには、数値や日付時刻などのさまざまなデータ型を割り当て、書式の詳細を指定することができます。

区切り文字付きファイル内のフィールド値の説明	例	割り当てられるデータ型
テキスト修飾子で囲んだ値	"ABC 社のサプライヤー" "6,990.75"	文字
フィールド内の任意の位置の値には、数値の区切り文字として使用されるカンマとピリオド、および負号 (-) を除く、数字以外の文字を 1 つ使用できます。	$995 (995)	文字
値には数値、数値の区切り文字、および負号 (-) のみを使用できます。	6,990.75	数値

区切り文字付きファイル内のフィールド値の説明	例	割り当てられるデータ型
	-6,990.75 995	
フィールドには 1 つまたは複数の空白値が出現します。		文字
区切り文字や英字の月名を含む複数の日付時刻値	2016/12/31 2016 年 12 月 31 日	文字
数字のみから成る日付時刻値	20161231	数値

フィールド データ型の識別子

以下の表は、FIELD でデータ*型*を指定するときに使用する必要がある文字の一覧を示します。各文字はデータ型に対応します。

たとえば、文字データ型を使用する姓フィールドを定義する場合は、"C": FIELD "Last_Name" C と指定します。

> **メモ**
>
> **データ定義ウィザード**を使用して EBCDIC、Unicode、または ASCII フィールドを含むテーブルを定義する場合、それらのフィールドには自動的に文字 "C"(CHARACTER 型) が割り当てられます。
>
> IMPORT ステートメントを手作業で入力するか、既存の IMPORT ステートメントを編集する場合、EBCDIC または Unicode フィールドに対して、より特有の文字 "E" または "U" に置き換えることができます。

文字	データ型
A	ACL
B	BINARY
C	CHARACTER
D	DATETIME
E	EBCDIC
F	FLOAT
G	ACCPAC
I	IBMFLOAT

文字	データ型
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 EXCEL コマンド

Microsoft Excel ワークシートまたは名前付き範囲を定義およびインポートして、Analytics テーブルを作成します。

構文

IMPORT EXCEL TO *table インポートファイル名* FROM *ソースファイル名* TABLE *入力ワークシートまたは名前付き範囲*<KEEPTITLE> {ALLFIELDS|CHARMAX *最大フィールド長*|[フィールド構文] <...*n*> <IGNORE *フィールド番号*> <...*n*>} <OPEN>

フィールド構文 ::=
FIELD *名前 型* {PIC *書式*|WID *文字* DEC *値*} AS *表示名*

パラメーター

名前	説明
TO *テーブル*	データをインポートする Analytics テーブルの名前。 **メモ** テーブルの名前は、64 文字までの英数字に制限されます。名前にはアンダースコア文字 (_) を使用できますが、他の特殊文字やスペースは使用できません。名前の先頭を数字にすることはできません。
インポートファイル名	作成する Analytics データファイルの名前。 *インポートファイル名*、.FIL ファイル拡張子を持つ引用符で囲まれた文字列として指定します。例："Invoices.FIL". デフォルトでは、データファイル(.FIL) は、Analytics プロジェクトが入っているフォルダーに保存されます。 既存の異なるフォルダーにデータファイルを保存するには、絶対または相対ファイルパスを指定します。 ◦ "C:\data\Invoices.FIL" ◦ "data\Invoices.FIL"
FROM *ソースファイル名*	ソースデータファイルの名前。*ソースファイル名*は引用符で囲む必要があります。 ソースデータファイルが Analytics プロジェクトと同じフォルダーに位置しない場合、ファイルの位置を指定するために絶対パスまたは相対パスを使用する必要があります。 ◦ "C:\data*ソースファイル名*" ◦ "data*ソースファイル名*"

名前	説明
TABLE *入力ワークシートまたは名前付き範囲*	インポートする Microsoft Excel ワークシート、あるいはソース データ ファイル内の名前付き範囲。 ◦ ワークシート名の最後に "$" 記号を入力する必要があります ◦ *入力ワークシートまたは名前付き範囲*は、引用符で囲んだ文字列として指定する必要があります。
KEEPTITLE 省略可能	データの代わりに、フィールド名としてデータの最初の行を処理します。省略すると、汎用フィールド名が使用されます。 フィールドを個別に定義する場合、KEEPTITLE は最初の FIELD よりも先に現れる必要があります。
ALLFIELDS	ソース データ ファイルの全フィールドがインポートされます。 **メモ** ALLFIELDS を指定する場合、個別の FIELD 構文、CHARMAX、IGNORE 指定しないでください。
CHARMAX *最大フィールド長*	フィールドを個別に定義していないときにのみ適用されます。 ソース データ ファイル内の文字データから発生する Analytics テーブルの任意のフィールドの文字の最大長。
FIELD *名前 型*	インポートするソース データ ファイル内の個別フィールドの名前およびデータ型。フィールドをインポート対象から除外する場合は、そのフィールドを指定しないでください。 *型*については、"フィールド データ型の識別子" ページ 242を参照してください。
PIC *書式*	**メモ** 数値フィールドまたは日付時刻フィールドにのみ適用されます。 ◦ **数値フィールド** - Analytics のビューとレポートに含まれる数値の表示形式。 ◦ **日付時刻フィールド** -ソース データの日付時刻値の物理形式(日付時刻文字、区切り文字の順など) **メモ** 日付時刻フィールドの場合、*形式*はソース データの物理形式と正確に一致する必要があります。たとえば、ソース データが 12/31/2014 である場合は、書式を "MM/DD/YYYY" として入力します。 *書式*は引用符で囲む必要があります。
WID *文字*	Analytics テーブル レイアウトにおけるフィールドの長さ(文字)。
DEC *値*	数値フィールドの小数点以下の桁数
AS *表示名*	新しい Analytics テーブルのビューにおけるフィールドの表示名(代替列見出し)。 *表示名*の値は引用符で囲まれた文字列。列見出しを改行したい場合は、語句の間にセミコロン(;)を入れます。 フィールドの定義時には、AS は必須です。表示名をフィールド名と同じにしたい場合は、空の表示名を入力します。つまり、次の構文を使用します。AS "" 2 つの二重引用符の間にスペースがないことを確認してください。

名前	説明
IGNORE *フィールド番号* <...*n*> 省略可能	テーブル レイアウトからフィールドを除外します。 *フィールド番号*は、ソース データにおけるフィールドの位置を指定します。たとえば、IGNORE 5 は、ソース データの 5 番目のフィールドを Analytics テーブル レイアウトから除外します。
OPEN 省略可能	コマンドを実行した後、コマンドによって作成されたテーブルを開きます。コマンドが出力テーブルを作成する場合にのみ有効です。

例

指定されたフィールドのインポート

Credit_Cards という名前の新しい Analytics テーブルを定義するインポートを行うとします。このインポートでは、Excel データの 1 行目をフィールド名として使用します。

この Analytics テーブルにはソース テーブルのうち、3 つのフィールドを定義、追加し、残りのフィールドは除外します。

```
IMPORT EXCEL TO Credit_Cards "Credit Cards.fil" FROM "Credit_Cards_Metaphor.xls" TABLE "Corp_Credit_Cards$" KEEPTITLEFIELD "CARDNUM" N WID 16 DEC 0 AS "カード番号"FIELD "EXPDT" D WID 10 PIC "YYYY-MM-DD"  AS "有効期限"FIELD "PASTDUEAMT" N WID 6 DEC 2 AS "支払期日"IGNORE 2 IGNORE 3 IGNORE 5 IGNORE 6 IGNORE 7 IGNORE 9 IGNORE 10 IGNORE 11 IGNORE 12
```

すべてのフィールドをインポートする

May_Transactions という名前の新しい Analytics テーブルを定義するインポートを行うとします。このインポートでは、Excel データの 1 行目をフィールド名として使用します。

Analytics テーブルにはソース テーブルのすべてのフィールドを含め、デフォルトのフィールド定義を使用します。

最も長い値に設定されたフィールド長

最初の例では、ソース データ ファイルの文字データから生成されるフィールドの長さは、そのフィールド内で最長の値の長さに設定されます。

```
IMPORT EXCEL TO May_Transactions "May_Transactions.fil" FROM "Trans_May.xls" TABLE "Trans1_May$" KEEPTITLE ALLFIELDS
```

制約されたフィールド長

2 番目の例では、ソース データ ファイルの文字データから生成されるフィールドの長さは、そのフィールド内で最長の値の長さの 100 文字の CHARMAX 値のうち、短い方に設定されます。

```
IMPORT EXCEL TO May_Transactions "May_Transactions.fil" FROM "Trans_May.xls" TABLE
"Trans1_May$" KEEPTITLE CHARMAX 100
```

備考

> **メモ**
>
> このコマンドの動作の詳細については、Analytics のヘルプを参照してください。

個別にフィールドを定義するか、デフォルト定義を使用してすべてのフィールドをインポートします。

Analytics テーブルに Excel ファイルをインポートするとき、FIELD パラメーターを使用して各フィールドを個別に定義するか、あるいは ALLFIELDS パラメーターまたは CHARMAX パラメーターを使用して、デフォルトの Analytics フィールド定義を基にすべてのフィールドをインポートすることができます。

データ インポートの最大サイズ

ファイル形式 .xlsx または .xlsm

.xlsx または .xlsm ファイルからインポートできる Excel の列の最大数、およびフィールド内の最大文字数は特定の数に制限されません。

これらの Excel ファイルの種類からのインポートは、Analytics データ ファイル(.fil) における 32 KB のレコード長の制限によって制御されます。ソース Excel ファイル内のいずれかのレコードによって 32 KB より長い Analytics レコードが作成される場合、インポートは失敗します。

ファイル形式 .xls

.xls(Excel 97 - 2003) ファイルのインポートは違う種類の処理を用いており、次を最大数としています。

- 255 列
- フィールドにつき 255 文字
- レコードにつき 32 KB
- 65,000 行

フィールド データ型の識別子

以下の表は、FIELD でデータ*型*を指定するときに使用する必要がある文字の一覧を示します。各文字はデータ型に対応します。

たとえば、文字データ型を使用する姓フィールドを定義する場合は、"C": FIELD "Last_Name" C と指定します。

メモ

データ定義ウィザードを使用してEBCDIC、Unicode、またはASCIIフィールドを含むテーブルを定義する場合、それらのフィールドには自動的に文字 "C"(CHARACTER 型) が割り当てられます。

IMPORTステートメントを手作業で入力するか、既存のIMPORTステートメントを編集する場合、EBCDICまたはUnicodeフィールドに対して、より特有の文字 "E" または "U" に置き換えることができます。

文字	データ型
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

文字	データ型
Y	UNISYS
Z	ZONED

IMPORT GRCPROJECT コマンド

HighBond プロジェクト テーブルをインポートして、Analytics テーブルを作成します。

構文

IMPORT GRCPROJECT TO *テーブル インポート ファイル名* PASSWORD *数値* FROM *組織 ID タイプ ID* <FIELD *名前* AS *表示名* <...*n*>>

パラメーター

名前	説明
TO *テーブル*	データをインポートする Analytics テーブルの名前。 **メモ** テーブルの名前は、64 文字までの英数字に制限されます。名前にはアンダースコア文字 (_) を使用できますが、他の特殊文字やスペースは使用できません。名前の先頭を数字にすることはできません。
インポート ファイル名	作成する Analytics データ ファイルの名前。 *インポート ファイル名*、.FIL ファイル拡張子を持つ引用符で囲まれた文字列として指定します。例 : "Invoices.FIL". デフォルトでは、データ ファイル(.FIL) は、Analytics プロジェクトが入っているフォルダーに保存されます。 既存の異なるフォルダーにデータ ファイルを保存するには、絶対または相対ファイル パスを指定します。 ◦ "C:\data\Invoices.FIL" ◦ "data\Invoices.FIL"
PASSWORD *番号*	使用するパスワード定義。 実際のパスワードを入力することを求めたり指定したりするのには、PASSWORD *番号*の構文は使用しません。パスワード定義とは、以前に PASSWORD コマンドか SET PASSWORD コマンドを使用して入力または設定されたパスワードのことです。 *番号*はパスワード定義の番号です。たとえば、以前に 2 つのパスワードをスクリプトで設定したり入力したりしている場合には、PASSWORD 2 により、2 番目のパスワードを使用することを指定するなどします。 パスワードの入力または設定の詳細については、次を参照してください。 • "PASSWORD コマンド" ページ 345 • "SET コマンド" ページ 404 • PASSWORD アナリティクス タグ パスワードの入力または設定の詳細については、次を参照してください。

<table>
<tr><th>名前</th><th>説明</th></tr>
<tr><td></td><td>

PASSWORD コマンド
SET コマンド
PASSWORD アナリティクス タグ

メモ
PASSWORD は必要な場合と不要な場合があります。スクリプトを実行する環境によって異なります。
<table>
<tr><td>Analytics
(オンライン アクティベーション)</td><td>PASSWORD は必要ではありません。
現在のユーザーの HighBond アクセストークンが自動的に使用されます</td></tr>
<tr><td>Analytics
(オフライン アクティベーション)</td><td rowspan="4">PASSWORD が必要です。</td></tr>
<tr><td>ロボット</td></tr>
<tr><td>Analytics Exchange</td></tr>
<tr><td>分析アプリウィンドウ</td></tr>
</table>
</td></tr>
<tr><td>FROM 組織ID タイプID</td><td>
組織とインポート中のデータを定義する情報のタイプ。

組織 ID - データをインポートしているプロジェクト組織
タイプID - インポートする情報のタイプ。

組織ID 値と タイプID 値はスペースを入れずにスラッシュで区切る必要があります: FROM "125@eu/audits" とします。
文字列全体を引用符で囲む必要があります。
<h3>組織 ID</h3>
組織 ID には、組織の ID 番号が含まれている必要があるほか、北米以外のデータ センターへインポートする場合には、そのデータ センターのコードも含まれている必要があります。組織 ID 番号とデータ センターのコードは、アット マーク(@) で区切る必要があります: FROM "125@eu" と指定します。
データ センター コードは、どの地域の HighBond サーバーからデータをインポートするのかを指定します。

ap - Asia Pacific(アジア太平洋)
au - Australia(オーストラリア)
ca - Canada(カナダ)
eu - Europe(ヨーロッパ)
us - North America(北米)

組織にインストールされた HighBond に対して承認されているデータ センター コードのみを使用できます。北米のデータ センターがデフォルトであるため、@us を指定するのはオプションです。
組織 ID 番号がわからない場合は、Analytics のユーザー インターフェイスを使用して、プロジェクトからテーブルをインポートします。組織 ID 番号はログのコマンドに含まれます。詳細については、ACL GRC データの定義を参照してください。
</td></tr>
</table>

名前	説明
	タイプ ID *タイプID* はインポートする情報のタイプを指定します。プロジェクトの情報は一連の関連するテーブルに含まれます。 *タイプ ID* では、次のリストの値のいずれかを使用します。必要に応じて、表示される値を正確認入力します(アンダースコアを含む) 。 ◦ 監査 - プロジェクト ◦ 統制テスト計画 - 統制テスト計画 ◦ 統制テスト - 統制テスト ◦ 統制 - 統制 ◦ - Actions - アクション ◦ 調査結果 - 問題 ◦ 軽減 - リスクコントロールの関連付け ◦ 説明文 - 説明文 ◦ 目標- 目標 ◦ リスク - リスク ◦ ウォークスルー - ウォークスルー **ヒント** プロジェクトのテーブルが関連付けられる方法および Analytics にインポートした後にテーブルを結合するために使用できるキーフィールドについては、ACL GRC データの定義を参照してください。
FIELD *名前* AS *表示名* <...*n*> 省略可能	インポートするソースデータの個別のフィールド。名前を指定します。 FIELD を省略すると、すべてのフィールドがインポートされます。 ◦ *名前*は、プロジェクトのテーブルの物理フィールド名と正確に一致する必要があります。大文字小文字も区別されます。 ◦ *表示名*は、新しい Analytics テーブルのビューにおけるフィールドの表示名(代替列見出し) です。各 FIELD 名の表示名を指定する必要があります. *表示名*の値は引用符で囲まれた文字列。 列見出しを改行したい場合は、語句の間にセミコロン(;) を入れます。 Analytics の他のいくつかの IMPORT コマンドとは異なり、FIELD 名を表示名として使用する方法として、空の*表示名*を指定することはできません。 **ヒント** 物理フィールド名を取得するには、Analytics のユーザーインターフェイスを使用して、プロジェクトから該当するテーブルをインポートします。物理フィールド名はログのコマンドに含まれます。 後続のインポートをスクリプト化できます。

例

プロジェクト テーブルからすべてのフィールドをインポートする

組織 286 に属するすべてのアクティブなプロジェクトに関する**プロジェクト** テーブル内のすべてのフィールドを、Analytics テーブル **All_Projects** にインポートするとします。接続を認証するため、番号付けされたパスワード定

義を追加します。以上を行うコマンドは次のようになります。

```
IMPORT GRCPROJECT TO All_Projects "C:\ACL GRC Project Data\All_Projects.fil"
PASSWORD 1 FROM "286@us/audits"
```

指定したフィールドをプロジェクト テーブルからインポートする

組織 286 に属するすべてのアクティブなプロジェクトに関する**プロジェクト** テーブル内の指定したフィールドを、Analytics テーブル **All_Projects** にインポートするとします。

```
IMPORT GRCPROJECT TO All_Projects "C:\ACL GRC Project Data\All_Projects.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"
```

すべてのフィールドを問題テーブルからインポートする

この例では、組織 286 に属するすべてのアクティブなプロジェクトに関する**問題**テーブルのすべてのフィールドを、Analytics テーブル **All_Issues** にインポートします。

```
IMPORT GRCPROJECT TO All_Issues "C:\ACL GRC Project Data\All_Issues.fil" FROM
"286@us/findings"
```

備考

> **メモ**
>
> このコマンドの動作の詳細については、Analytics のヘルプを参照してください。

パスワード定義の作成とパスワード値の指定

PASSWORD コマンド

PASSWORD コマンドを使用して、HighBond に接続するための番号付けされたパスワード定義を作成した場合、パスワードの値が指定されていないと、スクリプトを接続しようとするときにパスワード プロンプトが表示されます。

詳細については、"PASSWORD コマンド" ページ 345を参照してください。

SET PASSWORD コマンド

SET PASSWORD コマンドを使用して、HighBond に接続するための番号付けされたパスワード定義を作成した場合、パスワードの値が指定されていれば、パスワード プロンプトは表示されません。これは、無人で実行するように設計されたスクリプトに適しています。

詳細については、SET PASSWORD コマンドを参照してください。

HighBond アクセストークン

どの方法を用いてパスワード定義を作成したかにかかわらず、必要なパスワードの値は、HighBond アクセストークンです。

- **PASSWORD による方法** - ユーザーは、 [**ツール > HighBond アクセストークン**]を選択し、HighBond にサインインして、アクセストークンを取得できます。アクセストークンが返されるので、ユーザーはこれをコピーして、パスワード プロンプトに貼り付けることができます。
- **SET PASSWORD** による方法 - Analytics スクリプト内の SET PASSWORD コマンド構文にアクセストークンを挿入するには、**スクリプト エディター**で右クリックして [**挿入 > HighBond トークン**]を選択し、次に HighBond にサインインします。スクリプト内のカーソル位置にアクセストークンが挿入されます。

> **注意**
>
> 返されるアクセストークンは HighBond にサインインするために使用されるアカウントと一致します。他のユーザーが使用するスクリプトを作成している場合は、スクリプト作成者として、独自のアクセストークンを使用することは適切ではない場合があります。

インポート デバッグ機能

HighBond からのインポートには、簡易デバッグ機能があります。

インポートされたデータは、対象 Analytics プロジェクトを含むフォルダーの JSON 中間ファイルに一時的に格納されます。Analytics プロジェクトを含むフォルダーでは、データが Analytics にインポートされた後に削除する代わりに、JSON ファイルを保持するテキスト ファイルを作成できます。

- **JSON ファイルが存在する** -HighBond からのインポートが失敗し、JSON ファイルがコンピューターにある場合は、問題が HighBond 側ではなく、Analytics 側にあることがわかっています。
- **JSON ファイルが存在しない** - HighBond からのインポートが失敗し、JSON ファイルがコンピューターにない場合は、問題が HighBond 側にあることがわかっています。

この情報はトラブルシューティングで役立ちます。

JSON 中間ファイルの保持を設定する

ターゲット Analytics プロジェクトを含むフォルダーで、`_grc_import_debug.txt` という名前の空のテキストファイルを作成します。

HighBond のリザルトまたはプロジェクトからインポートするときには、JSON 中間ファイルは `results.json` という名前で保持されます。ファイルは後続の各 HighBond からのインポートするたびに上書きされます。

IMPORT GRCRESULTS コマンド

HighBond のリザルト テーブルまたは解釈をインポートして、Analytics テーブルを作成します。

構文

IMPORT GRCRESULTS TO *テーブル* *インポート ファイル名* PASSWORD *数値* FROM *リザルト リソース パス*<FIELD *名前* AS *表示名* <...*n*>>

パラメーター

名前	説明
TO *テーブル*	データをインポートする Analytics テーブルの名前。 **メモ** テーブルの名前は、64 文字までの英数字に制限されます。名前にはアンダースコア文字 (_) を使用できますが、他の特殊文字やスペースは使用できません。名前の先頭を数字にすることはできません。
インポート ファイル名	作成する Analytics データ ファイルの名前。 *インポート ファイル名*、.FIL ファイル拡張子を持つ引用符で囲まれた文字列として指定します。例: "Invoices.FIL". デフォルトでは、データ ファイル(.FIL) は、Analytics プロジェクトが入っているフォルダーに保存されます。 既存の異なるフォルダーにデータ ファイルを保存するには、絶対または相対ファイル パスを指定します。 ◦ "C:\data\Invoices.FIL" ◦ "data\Invoices.FIL"
PASSWORD *番号*	使用するパスワード定義。 実際のパスワードを入力することを求めたり指定したりするのには、PASSWORD *番号*の構文は使用しません。パスワード定義とは、以前に PASSWORD コマンドか SET PASSWORD コマンドを使用して入力または設定されたパスワードのことです。 *番号*はパスワード定義の番号です。たとえば、以前に 2 つのパスワードをスクリプトで設定したり入力したりしている場合には、PASSWORD 2 により、2 番目のパスワードを使用することを指定するなどします。 パスワードの入力または設定の詳細については、次を参照してください。 • "PASSWORD コマンド" ページ 345 • "SET コマンド" ページ 404 • PASSWORD アナリティクス タグ パスワードの入力または設定の詳細については、次を参照してください。

<table>
<tr><th>名前</th><th>説明</th></tr>
<tr><td></td><td>

PASSWORD コマンド
SET コマンド
PASSWORD アナリティクス タグ

<blockquote>
メモ
PASSWORD は必要な場合と不要な場合があります。スクリプトを実行する環境によって異なります。
<table>
<tr><td>Analytics
(オンライン アクティベーション)</td><td>PASSWORD は必要ではありません。
現在のユーザーの HighBond アクセストークンが自動的に使用されます</td></tr>
<tr><td>Analytics
(オフライン アクティベーション)</td><td rowspan="4">PASSWORD が必要です。</td></tr>
<tr><td>ロボット</td></tr>
<tr><td>Analytics Exchange</td></tr>
<tr><td>分析アプリウィンドウ</td></tr>
</table>
</blockquote>
</td></tr>
<tr><td>FROM リザルト リソース パス</td><td>
インポートしているデータへのリザルト パス。
パスの形式は、インポートしているデータによって異なります。パスの形式の詳細については、"リザルト パス" ページ 253を参照してください。
<blockquote>
メモ
リザルト パスの形式は API によって提供され、変更されることがあります。パスの正確な現在の構文を取得する最も簡単で信頼できる方法は、対象データの手動インポートを実行し、コマンド ログからパスをコピーすることです。
</blockquote>
</td></tr>
<tr><td>FIELD <i>名前</i> AS <i>表示名</i> <...<i>n</i>>
省略可能</td><td>
インポートするソース データの個別のフィールド。名前を指定します。
FIELD を省略すると、すべてのフィールドがインポートされます。
<h3>名前</h3>
名前は、リザルトのテーブルの物理フィールド名と正確に一致する必要があります。大文字小文字も区別されます。物理フィールド名を表示するには、次のいずれかを実行します。

リザルトで、テーブル ビューの列見出しをクリックします。 [フィールド名]の後に物理フィールド名が表示されます。
Analytics では、リザルトのテーブルをインポートするときに、フィールドを選択できるダイアログ ボックスの表示名の後にかっこ付きで物理フィールド名が表示されます。
<blockquote>
メモ
リザルトの物理フィールド名は、テーブル ビューの列見出しに使用される表示名ではありません。
</blockquote>

"リザルト データをインポートおよびエクスポートするときのフィールド名の考慮事項" ページ 254を参照してください。
</td></tr>
</table>

名前	説明
	表示名 *表示名*は、新しい Analytics テーブルのビューにおけるフィールドの表示名 (代替列見出し) です。各 FIELD *名*の表示名を指定する必要があります. *表示名*の値は引用符で囲まれた文字列。 列見出しを改行したい場合は、語句の間にセミコロン(;) を入れます。 Analytics の他のいくつかの IMPORT コマンドとは異なり、FIELD 名を表示名として使用する方法として、空の*表示名*を指定することはできません。

例

リザルトのテーブル内の指定したフィールドをインポートする

リザルトのテーブルから Analytics テーブル **T and E exceptions** に指定されたフィールドをインポートしています。

```
IMPORT GRCRESULTS TO T_and_E_exceptions "C:\Secondary Analysis\T_and_E_exceptions.fil" PASSWORD 1 FROM "results/api/orgs/11594/control_tests/185699/exceptions" FIELD "metadata.status" AS "Status" FIELD "EmpNo" AS "Employee Number" FIELD "DATE" AS "Date" FIELD "CARDNUM" AS "Card Number" FIELD "CODES" AS "MC Codes" FIELD "AMOUNT" AS "Amount" FIELD "DESCRIPTION" AS "Description"
```

リザルトのテーブル内のすべてのフィールドをインポートする

次の例では、リザルトのテーブルから Analytics テーブル **T and E exceptions** にすべてのフィールドをインポートしています。

```
IMPORT GRCRESULTS TO T_and_E_exceptions "C:\Secondary Analysis\T_and_E_exceptions.fil" PASSWORD 1 FROM "results/api/orgs/11594/control_tests/185699/exceptions"
```

リザルトの解釈内のデータをインポートする

次の例では、リザルトの解釈を Analytics テーブル **T and E exceptions filtered** にインポートしています。

```
IMPORT GRCRESULTS TO T_and_E_exceptions_filtered "C:\Secondary Analysis\T_and_E_exceptions_filtered.fil" FROM "results/api/orgs/11594/control_tests/185699/interpretations/22699/exceptions"
```

備考

> **メモ**
>
> このコマンドの動作の詳細については、Analytics のヘルプを参照してください。

並べ替え順やフィルターを保持する

リザルトからデータをインポートする場合、並べ替えやフィルターなどのデータカスタマイズは、データのインポート方法に応じて、インポート後の Analytics テーブルでは保持されるか、破棄されます。

- **テーブルのインポート** - データカスタマイズが破棄されます。省略するように選択したフィールドを除き、テーブルのすべてのデータがインポートされます。
- **解釈のインポート** - データカスタマイズが保持されます。

リザルト パス

> **メモ**
>
> リザルト パスの形式は API によって提供され、変更されることがあります。パスの正確な現在の構文を取得する最も簡単で信頼できる方法は、対象データの手動インポートを実行し、コマンド ログからパスをコピーすることです。

FROM パラメーターのリザルト パスは次の一般的な形式を使用します。

`FROM "results <-`*地域コード*`>/api/orgs/<`*組織 ID*`>/control_tests/<`*統制テスト ID*`>/exceptions`

例: `FROM "results/api/orgs/11594/control_tests/4356/exceptions"`

組織 ID は Launchpad にログインするときにブラウザーのアドレスバーに表示されます。統制テスト ID および解釈 ID は、リザルトでテーブルを表示するときにアドレスバーに表示されます。

以下のテーブルは、リザルト パスのすべてのバリエーションを示します。

インポートするには	次の形式のリザルト パスを使用します。
統制テスト (テーブル) のデータ	`FROM "results/api/orgs/11594/control_tests/4356/exceptions"`
統制テスト (テーブル) の監査証跡	`FROM "results/api/orgs/11594/control_tests/4356/audit_trail"`
統制テスト (テーブル) のコメント	`FROM "results/api/orgs/11594/control_tests/4356/comments"`
解釈	`FROM "results/api/orgs/11594/control_tests/4356/interpretations/1192/exceptions"`
デフォルト地域 (us) 以外の HighBond 地域のデータ	◦ **アジア太平洋** - `FROM "results-ap/api/orgs/11594/control_tests/4356/exceptions"` ◦ **オーストラリア** - `FROM "results-au/api/orgs/11594/control_tests/4356/exceptions"` ◦ **カナダ** - `FROM "results-ca/api/orgs/11594/control_tests/4356/exceptions"` ◦ **欧州** - `FROM "results-eu/api/orgs/11594/control_tests/4356/exceptions"`

システムで生成された情報列

リザルトからデータをインポートするときには、次で示すシステムで生成された情報列を1つ以上インポートすることができます。

システムで生成された列は次のいずれかです。

- リザルトのテーブルの一部であり、個々のレコードに関連する処理情報を含んでいます。
- 追加情報 - コレクション名、テーブル名、またはレコードID番号

以下に示すように、システムで生成された列のフィールド名を正確に指定する必要があります。Analyticsのユーザーインターフェイスを使用してリザルトからインポートするときには、デフォルトの表示名が適用されます。インポート処理のスクリプトを作成する場合は、表示名を自由に変更できます。

フィールド名	デフォルトの表示名
metadata.priority	優先度
metadata.status	進捗状況
metadata.publish_date	公開済み
metadata.publisher	公開者名
metadata.assignee	割り当てられたユーザー
metadata.group	グループ
metadata.updated_at	更新
metadata.closed_at	クローズ済み
extras.collection	コレクション
extras.results_table	結果テーブル
extras.record_id	レコードID

リザルト データをインポートおよびエクスポートするときのフィールド名の考慮事項

リザルトとAnalytics間でデータを往復する場合は、リザルトテーブルのすべてのフィールド名がより厳しいAnalyticsフィールド名要件を満たすことを確認する必要があります。そうでない場合、Analyticsとリザルトデータが一致しないおそれがあります。

たとえば、リザルトフィールド名の特殊文字は、Analyticsにインポートされるときに自動的にアンダースコアに変換されます。これは、フィールド名がリザルトの元の名前と一致しないことを意味します。後からAnalyticsデータをリザルトの元のテーブルにエクスポートする場合は、フィールドが正しく一致しません。

往復するデータでこの問題を回避するには、CSV または Excel ファイルからリザルトにデータをアップロードする前に、以下の Analytics フィールド名要件を満たしていることを確認します。

- 特殊文字またはスペースがない
- 数字で始まらない
- 英数文字またはアンダースコア文字(_) のみを含む

パスワード定義の作成とパスワード値の指定

PASSWORD コマンド

PASSWORD コマンドを使用して、HighBond に接続するための番号付けされたパスワード定義を作成した場合、パスワードの値が指定されていないと、スクリプトを接続しようとするときにパスワード プロンプトが表示されます。

詳細については、"PASSWORD コマンド" ページ 345を参照してください。

SET PASSWORD コマンド

SET PASSWORD コマンドを使用して、HighBond に接続するための番号付けされたパスワード定義を作成した場合、パスワードの値が指定されていれば、パスワード プロンプトは表示されません。これは、無人で実行するように設計されたスクリプトに適しています。

詳細については、SET PASSWORD コマンドを参照してください。

HighBond アクセストークン

どの方法を用いてパスワード定義を作成したかにかかわらず、必要なパスワードの値は、HighBond アクセストークンです。

- **PASSWORD による方法** - ユーザーは、 [**ツール > HighBond アクセストークン**]を選択し、HighBond にサインインして、アクセストークンを取得できます。アクセストークンが返されるので、ユーザーはこれをコピーして、パスワード プロンプトに貼り付けることができます。
- **SET PASSWORD** による方法 - Analytics スクリプト内の SET PASSWORD コマンド構文にアクセストークンを挿入するには、**スクリプト エディター**で右クリックして [**挿入 > HighBond トークン**]を選択し、次に HighBond にサインインします。スクリプト内のカーソル位置にアクセストークンが挿入されます。

> **注意**
>
> 返されるアクセストークンは HighBond にサインインするために使用されるアカウントと一致します。他のユーザーが使用するスクリプトを作成している場合は、スクリプト作成者として、独自のアクセストークンを使用することは適切ではない場合があります。

インポート デバッグ機能

HighBond からのインポートには、簡易デバッグ機能があります。

インポートされたデータは、対象 Analytics プロジェクトを含むフォルダーの JSON 中間ファイルに一時的に格納されます。Analytics プロジェクトを含むフォルダーでは、データが Analytics にインポートされた後に削除する代わりに、JSON ファイルを保持するテキスト ファイルを作成できます。

- **JSON ファイルが存在する** -HighBond からのインポートが失敗し、JSON ファイルがコンピューターにある場合は、問題が HighBond 側ではなく、Analytics 側にあることがわかっています。

- **JSON ファイルが存在しない** - HighBond からのインポートが失敗し、JSON ファイルがコンピューターにない場合は、問題が HighBond 側にあることがわかっています。

この情報はトラブルシューティングで役立ちます。

JSON 中間ファイルの保持を設定する

ターゲット Analytics プロジェクトを含むフォルダーで、`_grc_import_debug.txt` という名前の空のテキストファイルを作成します。

HighBond のリザルトまたはプロジェクトからインポートするときには、JSON 中間ファイルは `results.json` という名前で保持されます。ファイルは後続の各 HighBond からのインポートするたびに上書きされます。

大きいテーブルのインポート

単一の IMPORT GRCRESULTS コマンドを使用して、多数のフィールドがあるテーブルは正常にインポートされない場合があります。リザルト外で多数のフィールドを含む単一のテーブルを操作する必要がある場合は、次のいずれかの方法を使用します。

- **テーブルの分割** -は、2 つ以上の IMPORT GRCRESULTS コマンドを使用して、フィールドのサブセットをインポートしてから、JOIN コマンドを使用して、Analytics の結果テーブルを結合します。
- **テーブルをファイルにエクスポート** -は、CSV 形式にエクスポートを使用してから、IMPORT DELIMITED コマンドを使用して、Analytics に結果ファイルをインポートします。

IMPORT LAYOUT コマンド

外部のテーブルレイアウト ファイル(.layout) を Analytics プロジェクトにインポートします。

> **メモ:**
>
> バージョン 11 より前の Analytics では、外部テーブルレイアウト ファイルのファイル拡張子には .fmt を使用していました。拡張子を手動で指定すれば、拡張子 .fmt のテーブルレイアウト ファイルを今後もインポートすることができます。

構文

IMPORT LAYOUT *外部レイアウトファイル* TO *テーブルレイアウト名*

パラメーター

名前	説明
外部レイアウトファイル	外部テーブルレイアウト ファイルの名前。ファイル名やパスにスペースが含まれている場合は、"Ap Trans.layout" のように引用符で囲む必要があります。 ファイル拡張子 .layout はデフォルトで使用されるので、指定する必要はありません。.fmt など、別のファイル拡張子を必要に応じて使用することもできます。 レイアウト ファイルが Analytics プロジェクトと同じフォルダーにない場合は、ファイルの場所を指定するために絶対パス(例 : "C:\Saved layouts\Ap_Trans.layout") または相対パス (例 : "Saved layouts\Ap_Trans.layout") を使用する必要があります。
TO *テーブルレイアウト名*	インポートされたテーブルレイアウトの、Analytics プロジェクトにおける名前。たとえば、"Ap Trans May" と指定します。*テーブルレイアウト名*にスペースが含まれる場合は、引用符で囲まれた文字列として指定する必要があります。*外部レイアウトファイル*の名前とは異なる*テーブルレイアウト名*を指定することができます。

例

外部テーブル レイアウト ファイルのインポート

次の例では、Analytics プロジェクトに **Ap_Trans.layout** という名前の外部テーブルレイアウト ファイルをインポートし、**Ap_Trans _May** という新しいテーブルレイアウトを作成しています。

```
IMPORT LAYOUT "C:\Saved layouts\Ap_Trans.layout" TO "Ap_Trans_May"
```

備考

IMPORT LAYOUT の使用に適する場面

外部テーブルレイアウト ファイルをインポートし、データ ファイルに関連付けると、新しいテーブルレイアウトをゼロから作成する手間を省くことができます。

- インポートしたテーブルレイアウトに特定の Analytics データ ファイル(.fil) との関連が指定されており、プロジェクトを含んでいるフォルダーに同じ名前のデータ ファイルが存在する場合には、フォルダー内のそのデータ ファイルとインポートしたテーブルレイアウトは自動的に関連付けられます。
- プロジェクト フォルダーに同じ名前のデータ ファイルが存在しない場合は、インポートしたテーブルレイアウトを新しいデータ ソースにリンクする必要があります。

テーブルレイアウトとソース データ ファイルが対応している必要性

インポートするテーブルレイアウトと、それに関連付けるデータ ファイルは、対応している必要があります。つまり、データ ファイル内のデータの構造は、テーブルレイアウトのメタデータによって規定されるフィールド定義に対応している必要があります。

データ構造は、データ ファイルに含まれるデータ要素(フィールド) 、フィールド数やフィールドの並び順、各フィールドのデータ型や長さを示します。テーブルレイアウトとデータ ファイルが一致していない場合は、雑然としたデータになったり、データが欠落する結果となります。

IMPORT MULTIDELIMITED コマンド

複数の区切り文字付きファイルを定義およびインポートすることで、複数のAnalyticsテーブルを作成します。

構文

IMPORT MULTIDELIMITED <TO *インポート フォルダー*> FROM {*ソース ファイル名*|*ソース フォルダー*} *ソースの文字エンコード* SEPARATOR {*文字*|TAB|SPACE} QUALIFIER {*文字*|NONE} <CONSECUTIVE> STARTLINE *行番号* <KEEPTITLE> <CRCLEAR> <LFCLEAR> <REPLACENULL> <ALLCHAR>

> **メモ**
>
> IMPORT MULTIDELIMITED パラメーターを上記と完全に同じ順序で、以下のテーブルで指定する必要があります。
>
> 複数の区切り文字付きファイルを完全な形でインポートするには、それらすべてのファイルのインポート前の構造が同一である必要があります。
>
> 詳細については、"同一のファイル構造が必要" ページ 264を参照してください。

パラメーター

名前	説明
TO *インポート フォルダー* 省略可能	データをインポートするフォルダー。 フォルダーを指定するには、絶対ファイル パス、またはAnalyticsプロジェクトを含むフォルダーに相対的なファイル パスを使用します。*インポート フォルダー*は引用符で囲んで指定します。 **例** `TO "C:\Point of sale audit\Data\Transaction working data"` `TO "Data\Transaction working data"` TOを省略すると、データはAnalyticsプロジェクトを含むフォルダーにインポートされます。
FROM *ソース ファイル名* | *ソース フォルダー*	ソース データ ファイル、またはソース データファイルを含むフォルダーの名前。 *ソース ファイル名* または*ソース フォルダー*は引用符で囲んで指定します。 本コマンドでは、以下の4種類の区切り文字付きファイルがサポートされています。 ◦ `*.csv` ◦ `*.dat` ◦ `*.del` ◦ `*.txt`

名前	説明
	ルートの Analytics プロジェクト フォルダーのソース データ ファイル 複数のファイルを指定するには、ファイル名の中の一意の文字の代わりに、ワイルドカード文字 (*) を使用します。ワイルドカード文字は、任意の文字、数字、または特殊文字のゼロ (0) 回以上の出現を表します。 **例** FROM "Transactions_FY*.csv" selects: `Transactions_FY18.csv` `Transactions_FY17.csv` ワイルドカードは、ファイル名およびファイル拡張子における複数の位置にも使用できます。 **例** FROM "Transactions_FY*.*" selects: `Transactions_FY18.txt` `Transactions_FY17.csv` ルートの Analytics プロジェクト フォルダー以外の場所にあるソース データ ファイル ソース データ ファイルが Analytics プロジェクトと同じディレクトリに位置していない場合、ファイルの位置を指定するために、絶対ファイル パス、またはプロジェクトを含むフォルダからの相対ファイル パスを使用する必要があります。 **例** FROM "C:\Point of sale audit\Data\Transaction master files\Transactions_FY*.csv" FROM "Data\Transaction master files\Transactions_FY*.csv" ソース データ ファイルを含むフォルダー ソース データ ファイル名を指定する代わりに、そのファイルを含むフォルダーの名前だけを指定することもできます。フォルダーに含まれるファイルのうち、サポートされている区切り文字付きファイル(`*.csv`、`*.dat`、`*.del`、`*.txt`) がすべてインポートされます。 ソース データ フォルダーを指定するには、絶対ファイル パス、またはAnalytics プロジェクトを含むフォルダーに相対的なファイル パスを使用します。 **例** FROM "C:\Point of sale audit\Data\Transaction master files" FROM "Data\Transaction master files"

<table>
<tr><th>名前</th><th>説明</th></tr>
<tr><td>ソースの文字エンコード</td><td>ソース データの文字セットおよび文字エンコード。

お使いの Analytics エディションとソース データのエンコードに応じて、次のうち該当するコードを指定してください。

<table>
<tr><th>コード</th><th>Analytics のエディション</th><th>ソース データのエンコード</th></tr>
<tr><td>0</td><td>非 Unicode 版</td><td>すべてのデータ</td></tr>
<tr><td>0</td><td>Unicode 版</td><td>ASCII データ</td></tr>
<tr><td>2</td><td>Unicode 版</td><td>Unicode データ、UTF-16 LE エンコード</td></tr>
<tr><td>3 数値コード</td><td>Unicode 版</td><td>Unicode データ、UTF-16 LE エンコード

ソース データのエンコに適合する数値コードを決定するには、データ定義ウィザードを使用してインポートを実行し、「エンコードされたテキスト」オプションを選択し、付属するドロップダウンリストを使って適合するエンコードを見つけます。

コードを指定するには、3、1 つのスペース、数値コードを順に指定します。</td></tr>
</table>

メモ
コードを指定しない場合は、非 Unicode 版の Analytics では 0 が、Unicode 版の Analytics では 2 が、それぞれ自動的に使用されます。</td></tr>
<tr><td>SEPARATOR 文字 | TAB | SPACE</td><td>ソース データのフィールド間で使用される区切り文字。文字は引用符で囲まれた文字列として指定する必要があります。

タブまたは空白を区切り文字として指定するには、それらを二重引用符で囲んで入力するか、またはそれらのキーワードを使用します。

◦ SEPARATOR " " または SEPARATOR SPACE
◦ SEPARATOR "," または SEPARATOR SPACE</td></tr>
<tr><td>QUALIFIER 文字 | NONE</td><td>ソース データ内のフィールド値を折り返すため と識別するために使用するテキスト修飾子文字。文字は引用符で囲まれた文字列として指定する必要があります。

文字をテキスト修飾子として指定するには、二重引用符を一重引用符で囲む必要があります。QUALIFIER '"'

テキスト修飾子がないことを指定するには、次のいずれかの方法を使用します。

◦ QUALIFIER '""'
◦ QUALIFIER NONE</td></tr>
<tr><td>CONSECUTIVE

省略可能</td><td>連続したテキスト修飾子を単一の修飾子として扱います。</td></tr>
<tr><td>STARTLINE 行番号</td><td>データが始まる行。

たとえば、データの最初の 4 行にヘッダー情報が含まれており、それらをインポートの対象外とす</td></tr>
</table>

名前	説明
	る場合は、行番号「*5*」を指定します。 **メモ** IMPORT MULTIDELIMITED を 1 回実行することでインポートするすべての区切り文字付きファイルにおいて、データの開始行を同じにすることをお勧めします。 異なる開始行にする場合は、"同一のファイル構造が必要" ページ 264を参照してください。
KEEPTITLE 省略可能	STARTLINE に指定する行番号をデータでなくフィールド名と見なします。KEEPTITLE を省略すると、汎用フィールド名が使用されます。 **メモ** IMPORT MULTIDELIMITED を 1 回実行することでインポートするすべての区切り文字付きファイルにおいて、各フィールド名が同じ行番号になければなりません。 各フィールド名の行番号が異なる場合は、"同一のファイル構造が必要" ページ 264を参照してください。
CRCLEAR 省略可能	テキスト修飾子間で発生するすべての CR 文字(キャリッジリターン) をスペース文字で置換します。CRCLEAR を使用するには、QUALIFIER と文字値を指定する必要があります。 CRCLEAR とLFCLEAR の両方を使用する場合は、CRCLEAR を最初に記述する必要があります。
LFCLEAR 省略可能	テキスト修飾子間で発生するすべての LF 文字(ラインフィード) をスペース文字で置換します。LFCLEAR を使用するには、QUALIFIER と文字値を指定する必要があります。 CRCLEAR とLFCLEAR の両方を使用する場合は、CRCLEAR を最初に記述する必要があります。
REPLACENULL 省略可能	スペースで区切られたファイルで発生するすべての NUL 文字を置換します。置き換えられたすべての NULL 文字の数がログに記録されます。
ALLCHAR 省略可能	インポートされたすべてのフィールドには、自動的に文字のデータ型が割り当てられます。 **ヒント** インポートされたすべてのフィールドに文字のデータ型を割り当てると、区切り文字付きテキスト ファイルのインポート処理が容易になります。Analytics にインポートされたデータのフィールドには、数値や日付時刻などのさまざまなデータ型を割り当て、書式の詳細を指定することができます。 Analytics によって識別子のフィールドに数値のデータ型が自動的に割り当てられたテーブルをインポートする際、実際には文字のデータ型を使用する必要がある場合には、ALLCHAR を使用することができます。

例

後続の例では、次のような 12 個の区切り文字付きファイルに保存された月次取引データがある場合を想定しています。

- `Transactions_Jan.csv ～ Transactions_Dec.csv`

> **メモ**
>
> インポートする区切り文字付きファイルごとに、独立した Analytics テーブルが作成されます。

すべての区切り文字付きファイルをインポートする

12 個すべての区切り文字付きファイルをインポートするとします。各ファイル名に含まれる月名の位置で、ワイルドカード記号 (*) を使用します。

Analytics は各フィールドに適切なデータ型を割り当てようとします。

```
IMPORT MULTIDELIMITED FROM "Transactions_*.csv" 0 SEPARATOR "," QUALIFIER ""
CONSECUTIVE STARTLINE 1 KEEPTITLE
```

すべての区切り文字付きファイルを文字型データとしてインポートする

この例は上の例と似ていますが、相違点は、インポートされるすべてのフィールドに対して Analytics により自動的に文字のデータ型が割り当てられる点です。

```
IMPORT MULTIDELIMITED FROM "Transactions_*.csv" 0 SEPARATOR "," QUALIFIER ""
CONSECUTIVE STARTLINE 1 KEEPTITLE ALLCHAR
```

指定されたフォルダーからすべての区切り文字付きファイルをインポートする

`C:\Point of sale audit\Data\Transaction master files` フォルダーにあるすべての区切り文字付きファイルをインポートします。

```
IMPORT MULTIDELIMITED FROM "C:\Point of sale audit\Data\Transaction master files" 0
SEPARATOR "," QUALIFIER "" CONSECUTIVE STARTLINE 1 KEEPTITLE
```

指定されたフォルダーにあるすべての区切り文字付きファイルをインポートし、Analytics テーブルを別のフォルダーに保存

この例は上記の例と同じですが、Analytics テーブルをルート プロジェクト フォルダーに保存するのではなく、`C:\Point of sale audit\Data\Transaction working data` フォルダーに保存します。

```
IMPORT MULTIDELIMITED TO "C:\Point of sale audit\Data\Transaction working data" FROM
"C:\Point of sale audit\Data\Transaction master files" 0 SEPARATOR "," QUALIFIER ""
CONSECUTIVE STARTLINE 1 KEEPTITLE
```

備考

同一のファイル構造が必要

IMPORT MULTIDELIMITED を使って複数の区切り文字付きファイルを完全な形でインポートするには、それらすべてのファイルの構造が同一である必要があります。

異なる構造の区切り文字付きファイルをインポートした場合でも、後から Analytics でデータをクレンジングおよび正規化することもできます。ただし、このように行うと、より多くの作業が必要となる可能性があります。通常、これより簡単なのは、区切り文字付きファイルをインポートする前にそれらを同じ構造にしておくことです。

複数の区切り文字付きファイルを完全な形でインポートするには、以下の項目がすべてのファイルにおいて同じである必要があります。

項目	ACLScript のキーワード	問題	解決策
ソースデータの文字セットおよび文字エンコード	*数値コード*	(Analytics の Unicode 版のみ) ソースの区切り文字付きファイルに応じて、異なる文字エンコードが使用されています。たとえば、ASCII エンコードを使用しているファイルもあれば、Unicode エンコードを使用しているファイルもあります。	エンコード タイプ別にソース ファイルをグループ化し、各グループを別々にインポートします。
区切り文字	SEPARATOR	ソースの区切り文字付きファイルに応じて、フィールド間に異なる区切り文字が使用されています。	次のいずれかを実行します。 ◦ ソース ファイルをインポートする前に、それらの中の区切り文字を正規化します。 ◦ 区切り文字別にソース ファイルをグループ化し、各グループを別々にインポートします。
テキスト修飾子文字	QUALIFIER	ソースの区切り文字付きファイルに応じて異なるテキスト修飾子文字が使用されることで、フィールド値が折り返されるとともに識別されています。	次のいずれかを実行します。 ◦ ソース ファイルをインポートする前に、その中の修飾子文字を正規化します。 ◦ 修飾子文字別にソース ファイルをグループ化し、各グループを別々にインポートします。
データの開始行	STARTLINE	ソースの区切り文字付きファイルごとにデータの開始行が異なります。	次のいずれかを実行します。 ◦ ソース ファイルをインポートする前に、

項目	ACLScript のキーワード	問題	解決策
			それらの中の開始行を正規化します。 ◦ 同じ開始行を持つソースファイルをグループ化し、各グループを別々にインポートします。 ◦ すべてのファイルのうちで最も遅い開始行を*行番号*に指定してください。ファイルを Analytics テーブルにインポートしたら、"EXTRACT コマンド" ページ 193を使用することで、任意のテーブルから、不要としたヘッダー情報が含まれたレコードだけを抽出できます。
フィールド名	KEEPTITLE	ソースの区切り文字付きファイルに応じて、各フィールド名が異なる行番号にあります。	次のいずれかを実行します。 ◦ ソースファイルをインポートする前に、それらの中のフィールド名がある行番号を正規化します。 ◦ 同じフィールド名が同じ行番号にあるソースファイルをグループ化し、各グループを別々にインポートします。
フィールド名	KEEPTITLE	ソースの区切り文字付きファイルによって、フィールド名があるファイルと、フィールド名がないファイルがあります。	次のいずれかを実行します。 ◦ すべてのファイルをインポートする前に、フィールド名が必要なソースファイルにフィールド名を追加します。 ◦ フィールド名のあるソースファイルをグループ化するとともに、フィールド名のないソースファイルもグループ化します。次いで、各グループを別々にインポートします。 ◦ 汎用フィールド名を使ってすべてのファイルをインポートするには、KEEPTITLE を省略します。ファイルを Analytics テーブルにインポートしたら、"EXTRACT コマンド" ページ 193を使用することで、任意のテーブルから必要なデータだけを抽出できます。

複数の IMPORT DELIMITED コマンド

IMPORT MULTIDELIMITED コマンドは、インポートされるファイルごとに実際に個別の IMPORT DELIMITED コマンドを 1 つずつ実行するものです。ログの IMPORT MULTIDELIMITED エントリをダブルクリックすると、個別の IMPORT DELIMITED コマンドが表示領域に表示されます。

複数の区切り文字付きファイルをインポート後に結合する

複数の区切り文字付きファイルを個々の Analytics テーブルにインポート後に、それらを 1 つの Analytics テーブルに結合できます。たとえば、12 個の月次テーブルのデータをすべてのデータを含む単一の年次テーブルに結合できます。

複数の Analytics テーブルを結合する方法については、"APPEND コマンド" ページ 70 を参照してください。

IMPORT MULTIEXCEL コマンド

複数の Microsoft Excel ワークシートまたは名前付き範囲を定義およびインポートして、複数の Analytics テーブルを作成します。

構文

IMPORT MULTIEXCEL <TO *インポート フォルダー*> FROM {*ソース ファイル名*|*ソース フォルダー*} TABLE *入力ワークシートまたは名前付き範囲* <PREFIX> <KEEPTITLE> <CHARMAX *最大フィールド長*>

> **メモ**
>
> IMPORT MULTIEXCEL パラメーターを上記と完全に同じ順序で、以下のテーブルで指定する必要があります。

パラメーター

名前	説明
TO *インポート フォルダー* 省略可能	データをインポートするフォルダー。 フォルダーを指定するには、絶対ファイル パス、または Analytics プロジェクトを含むフォルダーに相対的なファイル パスを使用します。*インポート フォルダー*は引用符で囲んで指定します。 **例** TO "C:\Point of sale audit\Data\Transaction working data" TO "Data\Transaction working data" TO を省略すると、データは Analytics プロジェクトを含むフォルダーにインポートされます。
FROM *ソース ファイル名* | *ソース フォルダー*	ソース データ ファイル、またはソース データファイルを含むフォルダーの名前。 *ソース ファイル名* または*ソース フォルダー*は引用符で囲んで指定します。 **ルートの Analytics プロジェクト フォルダーのソース データ ファイル** ◦ **単一の Excel ファイル** Excel ファイルの完全なファイル名と拡張子を指定します。 **例** FROM "Transactions_FY18.xlsx"

名前	説明
	◦ **複数の Excel ファイル** 複数のファイルを指定するには、ファイル名の中の一意の文字の代わりに、ワイルドカード文字 (*) を使用します。ワイルドカード文字は、任意の文字、数字、または特殊文字のゼロ (0) 回以上の出現を表します。 **例** FROM "Transactions_FY*.xlsx" selects: Transactions_FY18.xlsx Transactions_FY17.xlsx ワイルドカードは、ファイル名およびファイル拡張子における複数の位置にも使用できます。 **例** FROM "Transactions_FY*.*" selects: Transactions_FY18.xlsx Transactions_FY17.xls **ルートの Analytics プロジェクト フォルダーのソース データ ファイル** ソース データ ファイルまたはファイルが Analytics プロジェクトと同じディレクトリに位置しない場合、ファイルの位置を指定するために絶対ファイル パスまたはプロジェクトを含むフォルダへの相対ファイル パスを使用する必要があります。 **例** FROM "C:\Point of sale audit\Data\Transaction master files\Transactions_FY18.xlsx" FROM "Data\Transaction master files\Transactions_FY*.xlsx" **ソース データ ファイルを含むフォルダー** ファイル名を指定する代わりに、ソース データ ファイルを含むフォルダーの名前を指定できます。 ソース データ フォルダーを指定するには、絶対ファイル パス、またはAnalytics プロジェクトを含むフォルダーに相対的なファイル パスを使用します。 **例** FROM "C:\Point of sale audit\Data\Transaction master files" FROM "Data\Transaction master files"

名前	説明
	メモ フォルダーを指定すると、ワークシート名が TABLE 値と一致する場合に、フォルダーの任意の Excel ファイルのワークシートがインポートされます。
TABLE *入力ワークシートまたは名前付き範囲*	インポートするワークシートまたは名前付き範囲の名前。インポートされた各ワークシートまたは名前付き範囲の個別の Analytics テーブルが作成されます。 *input_worksheets_or_named_ranges* は引用符で囲んで指定します。 ワークシートまたは範囲の名前で、固有の文字の代わりに、ワイルドカード (*) を使用します。 たとえば、"Trans_*$" は、次のワークシートを選択します。 ◦ Trans_Jan ◦ Trans_Feb ◦ Trans_Mar ◦ など **メモ** ワイルドカード文字 (*) は、任意の文字、数字、または特殊文字のゼロ (0) 回以上の出現を表します。 複数の場所でワイルドカードを使用できます。たとえば、*Trans*$ は以下を選択します。 • Trans_Jan • Jan_Trans **ドル記号 ($) の意味** Excel ファイルでは、ワークシートは、ワークシート名 (Trans_Jan$) に付加されているドル記号 ($) で識別されます。ドル記号は Excel には表示されません。 名前付き範囲は、ドル記号がないことによって特定されます (Trans_Jan_commercial)。 IMPORT MULTIEXCEL を使用するときには、ドル記号の指定は不要です。ただし、次の状況では、それを含めるか、除外してください。 ◦ **"$" を含める** -ワークシートのみをインポートし、名前付き範囲をインポートしない場合は、ワークシート名の最後にドル記号 ($) を入れます。 ◦ **"$" を除外する** - 名前付き範囲またはワークシートと名前付き範囲を 1 つのインポート処理でインポートする場合は、ドル記号を入れません。
PREFIX 省略可能	Excel ファイルを Analytics テーブルの名前の前に追加します。 **ヒント** 別のファイルのワークシートの名前が同じ場合は、Excel ファイル名を前に付けると、テーブル名の競合を回避できます。
KEEPTITLE 省略可能	データの代わりに、フィールド名としてデータの最初の行を処理します。省略すると、汎用フィールド名が使用されます。

名前	説明
	メモ インポートするワークシートと名前付き範囲のすべての最初の行は、一貫性がある方法を使用してください。最初の行は、すべてのデータセットで、フィールド名、データのいずれかです。1 つのインポート処理で 2 つの方法を混在させないでください。 データセットの最初の行へのアプローチに一貫性がない場合、2 つの別のインポート処理を使用してください。
CHARMAX *最大フィールド長* 省略可能	ソース データ ファイル内の文字データから発生する Analytics テーブルの任意のフィールドの文字の最大長。

例

次の例は、Excel ファイルに保存された 3 年間分の月次取引データがあるとします。

- Transactions_FY18.xlsx
- Transactions_FY17.xlsx
- Transactions_FY16.xlsx

各 Excel ファイルには 12 ワークシート (年の各月に 1 つずつ) 。また、ワークシートには、取引の各種サブセットを識別する一部の名前付き範囲もあります。

> **メモ**
>
> インポートする各ワークシートまたは名前付き範囲の個別の Analytics テーブルが作成されます。

ワークシートのインポート

すべての FY18 ワークシートのインポート

FY18 Excel ファイルから、すべての 12 ヶ月のワークシートをインポートします。

- 各ワークシート名で月が発生する場所で、ワイルドカード記号 (*) を使用します。
- ワークシート名の最後にドル記号 ($) を入れ、ワークシートのみが選択され、名前付き範囲が選択されないようにします。

```
IMPORT MULTIEXCEL FROM "Transactions_FY18.xlsx" TABLE "Trans_*$"
```

すべての FY18 ワークシートをインポートし、フィールド名を保持し、最大文字フィールド長を指定します。

この例は上記の例と同じですが、Excel ファイルのフィールド名を保持し、文字フィールドの長さを制限します。

- KEEPTITLE を追加すると、Excel データの最初の行をフィールド名として使用できます。
- CHARMAX 50 を追加し、Excel ファイルの文字データとして発生するフィールドが、結果の Analytics テーブルで 50 文字に制限されます。

```
IMPORT MULTIEXCEL FROM "Transactions_FY18.xlsx" TABLE "Trans_*$" KEEPTITLE
CHARMAX 50
```

すべての 3 つのファイルからすべてのワークシートをインポート

3 つの Excel ファイルから、すべての 36 ヶ月のワークシートをインポートします。

- 各ワークシート名で月が発生する場所で、ワイルドカード記号 (*) を使用します。
- ワークシート名の最後にドル記号 ($) を入れ、ワークシートのみが選択され、名前付き範囲が選択されないようにします。
- 各 Excel ファイル名で年が発生する場所で、ワイルドカード記号 (*) を使用します。
- 命名の競合の可能性を削減する方法として、PREFIX を使用して、ソース Excel ファイルの名前を各 Analytics テーブル名の前に追加します。

```
IMPORT MULTIEXCEL FROM "Transactions_FY18.xlsx" TABLE "Trans_*$"
```

名前付き範囲のインポート

すべての FY18 "Commercial_transaction" 名前付き範囲のインポート

すべての "Commercial_transaction" 名前付き範囲を FY18 Excel ファイルからインポートし、ワークシートと他の名前付き範囲を無視します。

- 別の範囲の名前で一意の識別子が発生する場所でワイルドカード記号 (*) を使用します。
- ドル記号 ($) を除外し、名前付き範囲が選択できるようにします。

```
IMPORT MULTIEXCEL FROM "Transactions_FY18.xlsx" TABLE "Commercial_transaction_*"
```

ワークシートと名前付き範囲のインポート

すべての FY18 ワークシートと名前付き範囲のインポート

FY18 Excel ファイルから、すべての 12 ヶ月のワークシートと他の名前付き範囲をインポートします。

- TABLE でワイルドカード記号 (*) のみを使用して、ファイルのすべてのワークシートと名前付き範囲が選択されるようにします。
- ドル記号 ($) を除外し、名前付き範囲が選択できるようにします。

```
IMPORT MULTIEXCEL FROM "Transactions_FY18.xlsx" TABLE "*"
```

ディレクトリの管理

指定されたフォルダーですべての Excel ファイルからすべてのワークシートをインポート

`C:\Point of sale audit\Data\Transaction master files` フォルダーで、すべての Excel ファイルからすべてのワークシートをインポートします。

- TABLE でワイルドカード記号 (*) のみを使用して、各ファイルのすべてのワークシートが選択されるようにします。また、ドル記号 ($) を使用して、ワークシートのみが選択され、名前付き範囲が選択されないようにします。
- 命名の競合の可能性を削減する方法として、PREFIX を使用して、ソース Excel ファイルの名前を各 Analytics テーブル名の前に追加します。

```
IMPORT MULTIEXCEL FROM "C:\Point of sale audit\Data\Transaction master files" TABLE "*$" PREFIX
```

指定されたフォルダーですべての Excel ファイルからすべてのワークシートをインポートし、Analytics テーブルを別のフォルダーに保存

この例は上記の例と同じですが、Analytics テーブルをルート プロジェクト フォルダーに保存するのではなく、`C:\Point of sale audit\Data\Transaction working data` フォルダーに保存します。

```
IMPORT MULTIEXCEL TO "C:\Point of sale audit\Data\Transaction working data" FROM "C:\Point of sale audit\Data\Transaction master files" TABLE "*$" PREFIX
```

備考

複数の IMPORT EXCEL コマンド

IMPORT MULTIEXCEL コマンドは、実際に、複数の個別の IMPORT EXCEL コマンドを実行します。インポートされる各ワークシートに 1 つあります。ログの IMPORT MULTIEXCEL エントリをダブルクリックすると、個別の IMPORT EXCEL コマンドが表示領域に表示されます。

インポートされた最後のテーブルは、自動的に開きます。

IMPORT MULTIEXCEL は OPEN キーワードをサポートしません。ただし、コマンドの実行後、インポートされた最後のテーブルが自動的に開きます。

インポート後に複数のワークシートを結合

複数のワークシートを個々の Analytics テーブルにインポート後に、それらを 1 つの Analytics テーブルに結合できます。たとえば、12 個の月次テーブルのデータをすべてのデータを含む単一の年次テーブルに結合できます。

複数のAnalyticsテーブルを結合する方法については、"APPENDコマンド" ページ70を参照してください。

IMPORT ODBC コマンド

ODBC データソースからデータを定義およびインポートし、Analytics テーブルを作成します。

ODBC は Open Database Connectivity であり、データベースにアクセスするための標準的な方法です。

構文

IMPORT ODBC SOURCE *ソース名* TABLE *テーブル名* <QUALIFIER *データ修飾子*> <OWNER *ユーザー名*> <USERID *ユーザー ID*> <PASSWORD *数値*> <WHERE *WHERE句*> <TO *テーブル名*> <WIDTH *最大フィールド幅*> <MAXIMUM *最大フィールド長*> <FIELDS *フィールド* <,...*n*>>

パラメーター

名前	説明
SOURCE *ソース名*	接続する ODBC データソースのデータソース名 (DSN) 。DSN は既に存在し、正しく構成されている必要があります。 **メモ** コンピューターにインストールされている Windows ODBC ドライバーを使用するデータソースに制限されています。ACCESSDATA コマンドで使用できる Analytics ネイティブ データ コネクターは IMPORT ODBC で使用できない場合があります。
TABLE *テーブル名*	データをインポートする ODBC データソースのテーブル名。 通常、*テーブル名* はソース データのデータベース テーブルを参照しますが、Analytics がインポートするものは何でもテーブルとして参照することができます。たとえば、Microsoft Text Driver を使用するならば、*テーブル名* は、データをインポートするテキスト ファイルを参照します。
QUALIFIER *データ修飾子* 省略可能	フィールド値を折り返すためと識別するためにテキスト修飾子として使用する文字。文字は引用符で囲まれた文字列として指定する必要があります。 引用符を使用して、二重引用符文字を指定します: `""`。
OWNER *ユーザー名* 省略可能	接続するテーブルを所有するデータベース ユーザー アカウントの名前。
USERID *ユーザー ID* 省略可能	データソースにアクセスするために必要なユーザー名。
PASSWORD *番号* 省略可能	使用するパスワード定義。 実際のパスワードを入力することを求めたり指定したりするのには、PASSWORD *番号* の構文

名前	説明
	は使用しません。パスワード定義とは、以前にPASSWORD コマンドか SET PASSWORD コマンドを使用して入力または設定されたパスワードのことです。 *番号*はパスワード定義の番号です。たとえば、以前に2 つのパスワードをスクリプトで設定したり入力したりしている場合には、PASSWORD 2 により、2 番目のパスワードを使用することを指定するなどします。 パスワードの入力または設定の詳細については、次を参照してください。 • "PASSWORD コマンド" ページ 345 • "SET コマンド" ページ 404 • PASSWORD アナリティクス タグ パスワードの入力または設定の詳細については、次を参照してください。 • PASSWORD コマンド • SET コマンド • PASSWORD アナリティクス タグ
WHERE *WHERE句* 省略可能	指定した条件に基づいて返されたレコードを制限する SQL WHERE 句。有効な SQL ステートメントである必要があるほか、引用符で囲んだ文字列として入力する必要があります。 `WHERE "SALARY > 50000"`
TO *テーブル名* 省略可能	作成する Analytics データファイルの名前。 *テーブル名*、.FIL ファイル拡張子を持つ引用符で囲まれた文字列として指定します。例：TO "Invoices.FIL". デフォルトでは、データファイル(.FIL) は、Analytics プロジェクトが入っているフォルダーに保存されます。 既存の異なるフォルダーにデータファイルを保存するには、絶対または相対ファイルパスを指定します。 ◦ TO "C:\data\Invoices.FIL" ◦ TO "data\Invoices.FIL".
WIDTH *最大フィールド長* 省略可能	インポートしているソースの文字データから発生する Analytics テーブルの任意のフィールドの文字の最大長。 1 から 254 までの値を入力することができます。デフォルト値は 50 です。最大フィールド長を超えるデータは、Analytics にインポートすると切り捨てられます。
MAXIMUM *最大フィールド長* 省略可能	インポートするテキスト、ノート、またはメモフィールドの文字の最大長。 1 から 1100 までの値を入力することができます。デフォルト値は 100 です。最大フィールド長を超えるデータは、Analytics にインポートすると切り捨てられます。
FIELDS *field* <,...*n*> 省略可能	インポートするソースデータの個別のフィールド。名前を指定します。 複数のフィールドを指定する場合は、カンマによってそれぞれのフィールドを区切る必要があります。FIELDS を省略すると、すべてのフィールドがインポートされます。 フィールド名を引用符で囲むと、大文字と小文字が区別されるようになります。引用符を使用する場合は、フィールド名の大文字小文字は、FIELDS とODBC データソースとの間で正確に一致する必要があります。引用符を使用した場合、フィールド名の大文字小文字が一致しないフィールドはインポートされません。

名前	説明
	メモ FIELDS は、IMPORT ODBC パラメーターの間の最後に置く必要があります。FIELDS が最後にない場合は、コマンドが失敗します。

例

SQL Server からデータをインポートする

次の例では、SQL Server データベースから **Trans_Dec11** という名前の Analytics テーブルにデータをインポートしています。

```
IMPORT ODBC SOURCE "SQLServerAudit" TABLE "Transactions" OWNER "audit" TO "C:\ACL DATA\Trans_Dec11.FIL" WIDTH 100 MAXIMUM 200 FIELDS "CARDNUM","CREDLIM","CUSTNO","PASTDUEAMT"
```

備考

ODBC データソースに接続するための古い方法

IMPORT ODBC コマンドは、Analytics から ODBC 準拠のデータソースに接続するための古い方法です。ODBC データソースに接続するための新しい方法では、Data Access ウィンドウと ACCESSDATA コマンドを使用します。

IMPORT ODBC は引き続き Analytics で使用できます。ただし、この接続方法はスクリプトと Analytics コマンドラインでしか使用できなくなりました。この接続方法に**データ定義ウィザード**からアクセスすることはできなくなりました。

日付時刻値の時刻部分を非表示にする

IMPORT ODBC コマンドを使用して Analytics テーブルを定義する場合は、このコマンドの前に SET SUPPRESSTIME ON コマンドを置くことにより、日付時刻値の時刻部分を非表示にすることができます。

この機能により、日付時刻値の時刻部分が自動的に切り捨てられていた、バージョン 10.0 より前の Analytics で作成された Analytics スクリプトを組み込むことができます。それらのスクリプトは、SET SUPPRESSTIME ON を追加しない場合には、日付時刻型対応バージョンの Analytics では動作しません。

詳細については、"SET コマンド" ページ 404の「SET SUPPRESSTIME」セクションを参照してください。

IMPORT PDF コマンド

Adobe PDF ファイルを定義およびインポートして、Analytics テーブルを作成します。

構文

IMPORT PDF TO *table* <PASSWORD *番号*> *インポート ファイル名* FROM *ソース ファイル名* <SERVER *プロファイル名*> *スキップ長さ*<PARSER "VPDF"> <PAGES *ページ範囲*> {[レコード構文] [フィールド構文] <...*n*>} <...*n*>

レコード構文 ::=
RECORD *レコード名 レコード タイプ レコードの行 透明* [テスト構文] <...*n*>

テスト構文 ::=
TEST *含める/除外する 一致タイプ* AT *開始行,開始位置,範囲 論理 テキスト*

フィールド構文 ::=
FIELD *名前 型* AT *開始行,開始位置* SIZE *長さ,フィールドの行* DEC *値* WID *バイト* PIC *形式* AS *表示名*

パラメーター

一般パラメーター

名前	説明
TO *テーブル*	データをインポートする Analytics テーブルの名前。 **メモ** テーブルの名前は、64 文字までの英数字に制限されます。名前にはアンダースコア文字 (_) を使用できますが、他の特殊文字やスペースは使用できません。名前の先頭を数字にすることはできません。
PASSWORD *番号* 省略可能	パスワードで保護された PDF ファイルのパスワードに使用します。 使用するパスワード定義。 実際のパスワードを入力することを求めたり指定したりするのには、PASSWORD *番号*の構文は使用しません。パスワード定義とは、以前に PASSWORD コマンドか SET PASSWORD コマンドを使用して入力または設定されたパスワードのことです。 *番号*はパスワード定義の番号です。たとえば、以前に 2 つのパスワードをスクリプトで設定したり

名前	説明
	入力したりしている場合には、PASSWORD 2 により、2 番目のパスワードを使用することを指定するなどします。 パスワードの入力または設定の詳細については、次を参照してください。 • "PASSWORD コマンド" ページ 345 • "SET コマンド" ページ 404 • PASSWORD アナリティクス タグ パスワードの入力または設定の詳細については、次を参照してください。 • PASSWORD コマンド • SET コマンド • PASSWORD アナリティクス タグ
インポートファイル名	作成する Analytics データ ファイルの名前。 *インポートファイル名*、.FIL ファイル拡張子を持つ引用符で囲まれた文字列として指定します。例："Invoices.FIL". デフォルトでは、データ ファイル(.FIL) は、Analytics プロジェクトが入っているフォルダーに保存されます。 既存の異なるフォルダーにデータ ファイルを保存するには、絶対または相対ファイル パスを指定します。 ◦ "C:\data\Invoices.FIL" ◦ "data\Invoices.FIL"
FROM *ソースファイル名*	ソース データ ファイルの名前。*ソース ファイル名*は引用符で囲む必要があります。 ソース データ ファイルが Analytics プロジェクトと同じフォルダーに位置しない場合、ファイルの位置を指定するために絶対パスまたは相対パスを使用する必要があります。 ◦ "C:\data*ソース ファイル名*" ◦ "data*ソース ファイル名*"
SERVER *プロファイル名* 省略可能	インポートするデータを含むサーバーのプロファイル名。
スキップする長さ 省略可能	ファイルの先頭でスキップするバイト数。 たとえば、先頭から 32 バイトにヘッダー情報が含まれている場合、この情報を除外するには、スキップする長さの値として 32 を指定します。 **メモ** Unicode データでは、偶数バイトのみを指定します。奇数のバイトを指定すると、インポートされたデータの後続の処理で問題が発生する可能性があります。
PARSER "VPDF" 省略可能	ファイル定義の処理時に VeryPDF パーサーを使用して、PDF ファイルを解析します。 PARSER を省略した場合は、デフォルトの Xpdf パーサーが使用されます。 初めて PDF ファイルをインポートし、特に理由がない場合は、デフォルトの Xpdf パーサーを使用してください。Xpdf を使用して、既にデータの整列に問題が発生している場合は、VeryPDF パーサーを使用して解析結果が改善されるか確認してください。
PAGES *ページの範囲*	PDF ファイル内のすべてのページをインポートしない場合の、含めるページの範囲。*ページの範*

名前	説明
省略可能	*囲*は、引用符で囲んだ文字列として指定する必要があります。 以下を指定できます。 ◦ カンマで区切った個々のページ(1,3,5) ◦ ページの範囲(2-7) ◦ ページと範囲の組み合わせ(1, 3, 5-7, 11) PAGES を省略する場合は、PDF ファイルのすべてのページがインポートされます。

RECORD パラメーター

一般レコード定義情報。

> **メモ**
>
> 一部のレコード定義情報は、データ定義ウィザードのオプションにマッピングされる数値コードを使用して指定されます。
>
> **スクリプトでは、オプション名ではなく、数値コードを指定します。**

名前	説明
RECORD *レコード名*	データ定義ウィザードのレコード名。 IMPORT PDF コマンドでは、*レコード名*を指定する必要がありますが、*レコード名*値は結果の Analytics テーブルに表示されません。 データ定義ウィザードでは、Analytics は、レコードのタイプに基づいて、デフォルト名を提供します。 ◦ 詳細 ◦ ヘッダー*n* ◦ フッター*n* デフォルト名を指定するか、別の名前を指定できます。
レコードのタイプ	PDF ファイルを定義するときには、3 つのレコード タイプがあります。 ◦ 0 - 詳細 ◦ 1 - ヘッダー ◦ 2 - フッター > **メモ** > 単一の IMPORT PDF の実行では、複数のセットのヘッダーおよびフッター レコードを定義できますが、詳細レコードは 1 セットのみです。
レコードの行	PDF ファイルのレコードが占める行数。 PDF ファイルのデータと一致する単一行または複数行のレコードを定義します。
透明	ヘッダー レコードの透明設定。 > **メモ** > ヘッダー レコードにのみ適用されます。 ◦ 0 - 不透明

名前	説明
	◦ 1 - 透明 透明なヘッダー レコードは複数行の詳細レコードに分割されません。 ヘッダー レコードがソース PDF ファイルの複数行の詳細レコードを分割する場合 (改ページで発生する場合があります) 、 [1] (透明) を指定すると、結果の Analytics テーブルの詳細レコードが統合されます。

TEST パラメーター

PDF ファイルのレコードのセットを定義する条件。RECORD の各出現には、1 つ以上の TEST (8つまで) があります。

> **メモ**
>
> 一部の条件は、データ定義ウィザードのオプション(オプション名は以下で括弧内に示されています) にマッピングされる数値コードを使用して指定されます。
>
> **スクリプトでは、オプション名ではなく、数値コードを指定します。**

名前	説明
TEST *含める/除外する*	一致するデータを処理する方法: ◦ 0 - (**含める**) 条件を満たすデータがレコードのセットに含まれます ◦ 1 - (**除外する**) 条件を満たすデータがレコードのセットから除外されます
一致タイプ	実行する一致のタイプ: ◦ 0 - (**正確に一致する**) 一致するレコードには、指定された文字または文字列が、指定された位置で始まる、指定された開始行に含まれている必要があります。 ◦ 2 - (**英文字**) 一致するレコードには、指定された開始行の指定された開始位置にあるか、指定された範囲のすべての位置にある 1 つ以上の英文字を含んでいる必要があります。 ◦ 3 - (**数字**) 一致するレコードには、指定された開始行の指定された開始位置にあるか、指定された範囲のすべての位置にある 1 つ以上の数字を含んでいる必要があります。 ◦ 4 - (**空白**) 一致するレコードには、指定された開始行の指定された開始位置にあるか、指定された範囲のすべての位置にある 1 つ以上の空白を含んでいる必要があります。 ◦ 5 - (**空白以外**) 一致するレコードには、指定された開始行の指定された開始位置にあるか、指定された範囲のすべての位置にある 1 つ以上の空白以外の文字 (特殊文字を含む) を含んでいる必要があります。 ◦ 7 - (**行で検索**) 一致するレコードには、指定された文字または文字列が、指定された位置で始まる、指定された開始行に含まれている必要があります。 ◦ 8 - (**範囲で検索**) 一致するレコードには、 [テキスト] フィールドにある文字または文字列が、指定された範囲の、レコードの指定された行の指定された範囲のどこかに含まれている必要があります。 ◦ 10 - (**独自のマップ**) 一致するレコードには、指定された文字パターンと一致する文字が、指定された位置で始まる、指定された開始行に含まれている必要があります。
AT *開始行*, *開始位置*, *範囲*	◦ ***開始行*** -条件が適用されるレコードの行 たとえば、独自のマップを使用して郵便番号と一致させ、郵便番号が 3 行レコードの 3 行目に表示される場合は、 [*開始行*] で 3 を指定する必要があります。

名前	説明
	メモ 単一行のレコードの場合、*開始行*の値は常に1です。 ◦ ***開始位置*** -条件と比較するPDFファイルの開始バイト位置 ◦ ***範囲*** -条件との比較で使用するPDFファイルの開始バイト位置からのバイト数 範囲なしで開始バイト位置のみを使用している場合は、*範囲*に0を指定します。 **メモ** 非 Unicode 版 Analytics: 1バイト=1文字 Unicode 版 Analytics: 2バイト=1文字
論理	条件間の論理関係: ◦ 0 - (**And**) 現在および次の条件は、論理 AND に関連しています ◦ 1 - (**Or**) 現在および次の条件は、論理 OR に関連しています ◦ 4 - (**新しいグループ** > **And**) 現在の条件は論理条件のグループの最後です。現在のグループと次のグループは、論理 AND で関連付けられます。 ◦ 5 - (**新しいグループ** > **Or**) 現在の条件は論理条件のグループの最後です。現在のグループと次のグループは、論理 OR で関連付けられます。 ◦ 7 - (**終了**) 現在の条件は論理条件のグループの最後です。
テキスト	次に対して一致するリテラルまたはワイルドカード文字: ◦ **[正確に一致する]、[行で検索]、または [範囲で検索]** -]の場合、PDFファイルのレコードのセットを一意に識別する文字または文字列を指定します。 ◦ **[独自のマップ]** -]の場合、PDFファイルのレコードのセットを一意に識別する文字パターンを指定します。 **カスタム マップ**オプションは、"MAP( ) 関数" ページ 624と同じ構文を使用します。 他の一致タイプについては、*テキスト*は空の文字列 "" です。

FIELD パラメーター

フィールド定義情報。

名前	説明
FIELD *名前 型*	インポートするソースデータファイル内の個別フィールドの名前およびデータ型。フィールドをインポート対象から除外する場合は、そのフィールドを指定しないでください。 *型*については、"フィールド データ型の識別子" ページ 283を参照してください。
AT *開始行*, *開始位置*	◦ ***開始行*** -PDFファイルのレコードのフィールドの開始行 PDFファイルの複数行のレコードの場合、*開始行*はレコードの任意の行でフィールドを開始できます。*レコードの行*が1の場合、*開始行*は常に1です。 ◦ ***開始位置*** - PDFファイルのフィールドの開始バイト位置

名前	説明
	メモ 非 Unicode 版 Analytics: 1 バイト = 1 文字 Unicode 版 Analytics: 2 バイト = 1 文字 Unicode 版 Analytics では、一般的に、奇数で開始するバイト位置を指定してください。偶数の開始位置を指定すると、文字が正しく表示されない可能性があります。
SIZE *長さ, フィールドの行*	◦ ***長さ*** - Analytics テーブル レイアウトにおけるフィールドの長さ(バイト数) **メモ** 非 Unicode 版 Analytics: 1 バイト = 1 文字 Unicode 版 Analytics: 2 バイト = 1 文字 Unicode 版 Analytics では、偶数バイトのみを指定します。奇数バイトを指定すると、文字が正しく表示されない可能性があります。 ◦ ***フィールドの行*** -PDF ファイルの単一フィールド値が占める行数 ファイルのデータと一致する単一行または複数行のフィールドを定義できます。 **メモ** フィールドに指定された行数は、フィールドを含むレコードに指定された行数を超えることができません。
DEC *値*	数値フィールドの小数点以下の桁数
WID *バイト*	フィールドの表示幅(バイト数) 。 指定した値によって、Analytics のビューおよびレポートにおけるフィールドの表示幅が決まります。表示幅はデータを変更するものではありませんが、表示幅がフィールド長より短い場合にはデータが隠れる可能性があります。
PIC *書式*	**メモ** 数値フィールドまたは日付時刻フィールドにのみ適用されます。 ◦ **数値フィールド** - Analytics のビューとレポートに含まれる数値の表示形式。 ◦ **日付時刻フィールド** -ソース データの日付時刻値の物理形式(日付時刻文字、区切り文字の順など) **メモ** 日付時刻フィールドの場合、*形式*はソース データの物理形式と正確に一致する必要があります。たとえば、ソース データが 12/31/2014 である場合は、書式を "MM/DD/YYYY" として入力します。 *書式*は引用符で囲む必要があります。
AS *表示名*	新しい Analytics テーブルのビューにおけるフィールドの表示名(代替列見出し) 。 *表示名*の値は引用符で囲まれた文字列。列見出しを改行したい場合は、語句の間にセミコロン(;) を入れます。

名前	説明
	フィールドの定義時には、AS は必須です。表示名をフィールド名と同じにしたい場合は、空の表示名を入力します。つまり、次の構文を使用します。AS "" 2 つの二重引用符の間にスペースがないことを確認してください。

例

PDF ファイルの特定のページからデータをインポートする

次の例では、パスワードで保護された PDF ファイル `Vendors.pdf` の 1 ページからデータをインポートしています。

詳細レコードの1 セットと3 つのフィールドが、結果の Analytics テーブル **Vendor_List** に作成されます。

```
IMPORT PDF TO Vendor_List PASSWORD 1 "Vendor_List.FIL" FROM "Vendors.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 ""
```

備考

> **メモ**
>
> このコマンドの動作の詳細については、Analytics のヘルプを参照してください。

Analytics Unicode 版への PDF のインポートのトラブルシューティング

Analytics の Unicode 版を使って PDF ファイルをインポートする際に発生した問題は、長さの仕様に関連している可能性があります。

- 外国語の文字が突然表示されたり、結果の Analytics テーブルのレイアウトが歪んだりしている場合は、SIZE の長さを偶数に設定してください。

  SIZE の長さに奇数のバイトを指定すると、インポートされたデータの処理で問題が発生する可能性があります。
- 作成する Analytics テーブルにレコードが含まれない場合、スキップするファイルの先頭にヘッダーデータがあるときは、*skip_length* を 2 またはその他の偶数に設定してみてください。

フィールド データ型の識別子

以下の表は、FIELD でデータ*型*を指定するときに使用する必要がある文字の一覧を示します。各文字はデータ型に対応します。

たとえば、文字データ型を使用する姓フィールドを定義する場合は、"C": FIELD "Last_Name" C と指定します。

> **メモ**
>
> **データ定義ウィザード**を使用してEBCDIC、Unicode、またはASCIIフィールドを含むテーブルを定義する場合、それらのフィールドには自動的に文字 "C"(CHARACTER 型) が割り当てられます。
>
> IMPORT ステートメントを手作業で入力するか、既存のIMPORT ステートメントを編集する場合、EBCDIC またはUnicode フィールドに対して、より特有の文字 "E" または "U" に置き換えることができます。

文字	データ型
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

文字	データ型
X	NUMERIC
Y	UNISYS
Z	ZONED

IMPORT PRINT コマンド

印刷イメージ(レポート) ファイルを定義およびインポートして、Analytics テーブルを作成します。

構文

IMPORT PRINT TO *テーブル インポート ファイル名* FROM *ソース ファイル名* <SERVER *プロファイル名*> *文字セット値* <*コード ページ番号*> {[レコード構文] [フィールド構文] <...*n*>} <...*n*>

レコード構文 ::=
RECORD *レコード名 レコード タイプ レコードの行 透明* [テスト構文] <...*n*>

テスト構文 ::=
TEST *含める/除外する 一致タイプ* AT *開始行*,*開始位置*,*範囲 論理 テキスト*

フィールド構文 ::=
FIELD *名前 型* AT *開始行*,*開始位置* SIZE *長さ*,*フィールドの行* DEC *値* WID *バイト* PIC *形式* AS *表示名*

パラメーター

一般パラメーター

名前	説明
TO *テーブル*	データをインポートする Analytics テーブルの名前。 **メモ** テーブルの名前は、64 文字までの英数字に制限されます。名前にはアンダースコア文字(_) を使用できますが、他の特殊文字やスペースは使用できません。名前の先頭を数字にすることはできません。
インポート ファイル名	作成する Analytics データ ファイルの名前。 *インポート ファイル名*、.FIL ファイル拡張子を持つ引用符で囲まれた文字列として指定します。例: "Invoices.FIL". デフォルトでは、データ ファイル(.FIL) は、Analytics プロジェクトが入っているフォルダーに保存されます。 既存の異なるフォルダーにデータ ファイルを保存するには、絶対または相対ファイル パスを指定します。

名前	説明
	◦ "C:\data\Invoices.FIL" ◦ "data\Invoices.FIL"
FROM *ソースファイル名*	ソース データ ファイルの名前。*ソース ファイル名*は引用符で囲む必要があります。 ソース データ ファイルが Analytics プロジェクトと同じフォルダーに位置しない場合、ファイルの位置を指定するために絶対パスまたは相対パスを使用する必要があります。 ◦ "C:\data*ソース ファイル名*" ◦ "data*ソース ファイル名*"
SERVER *プロファイル名* 省略可能	インポートするデータを含むサーバーのプロファイル名。
文字セット値	印刷イメージ(レポート)ファイルのエンコードに使用する文字セット。以下の値がサポートされています。 ◦ 0 - ASCII ◦ 1 - EBCDIC ◦ 2 - Unicode ◦ 3 - エンコードされたテキスト
コードページ番号 省略可能	*文字セット値*に対して 3(エンコードされたテキスト)を指定した場合は、さらにコード ページ番号を入力する必要があります。

RECORD パラメーター

一般レコード定義情報。

> **メモ**
>
> 一部のレコード定義情報は、データ定義ウィザードのオプションにマッピングされる数値コードを使用して指定されます。
>
> **スクリプトでは、オプション名ではなく、数値コードを指定します。**

名前	説明
RECORD *レコード名*	データ定義ウィザードのレコード名。 IMPORT PRINT コマンドでは、*レコード名*を指定する必要がありますが、*レコード名*値は結果の Analytics テーブルに表示されません。 データ定義ウィザードでは、Analytics は、レコードのタイプに基づいて、デフォルト名を提供します。 ◦ 詳細 ◦ ヘッダー*n* ◦ フッター*n* デフォルト名を指定するか、別の名前を指定できます。
レコードのタイプ	印刷イメージ ファイルを定義するときには、3 つのレコード タイプがあります。

名前	説明
	◦ 0 - 詳細 ◦ 1 - ヘッダー ◦ 2 - フッター **メモ** 単一の IMPORT PRINT の実行では、複数のセットのヘッダーおよびフッターレコードを定義できますが、詳細レコードは 1 セットのみです。
レコードの行	印刷イメージ ファイルのレコードが占める行数。 ファイルのデータと一致する単一行または複数行のレコードを定義します。
透明	ヘッダー レコードの透明設定。 **メモ** ヘッダー レコードにのみ適用されます。 ◦ 0 - 不透明 ◦ 1 - 透明 透明なヘッダー レコードは複数行の詳細レコードに分割されません。 ヘッダー レコードがソースの印刷イメージ ファイルの複数行の詳細レコードを分割する場合 (改ページで発生する場合があります) 、 1 (透明) を指定すると、結果の Analytics テーブルの詳細レコードが統合されます。

TEST パラメーター

印刷イメージ ファイルのレコードのセットを定義する条件。RECORD の各出現には、1 つ以上の TEST (8つまで) があります。

> **メモ**
>
> 一部の条件は、データ定義ウィザードのオプション(オプション名は以下で括弧内に示されています) にマッピングされる数値コードを使用して指定されます。
>
> **スクリプトでは、オプション名ではなく、数値コードを指定します。**

名前	説明
TEST *含める/除外する*	一致するデータを処理する方法: ◦ 0 - (**含める**) 条件を満たすデータがレコードのセットに含まれます ◦ 1 - (**除外する**) 条件を満たすデータがレコードのセットから除外されます
一致タイプ	実行する一致のタイプ: ◦ 0 - (**正確に一致する**) 一致するレコードには、指定された文字または文字列が、指定された位置で始まる、指定された開始行に含まれている必要があります。 ◦ 2 - (**英文字**) 一致するレコードには、指定された開始行の指定された開始位置にあるか、指定された範囲のすべての位置にある 1 つ以上の英文字を含んでいる必要があります。 ◦ 3 - (**数字**) 一致するレコードには、指定された開始行の指定された開始位置にあるか、指定された範囲のすべての位置にある 1 つ以上の数字を含んでいる必要があります。 ◦ 4 - (**空白**) 一致するレコードには、指定された開始行の指定された開始位置にあるか、指

名前	説明
	定された範囲のすべての位置にある 1 つ以上の空白を含んでいる必要があります。 ◦ 5 - (**空白以外**) 一致するレコードには、指定された開始行の指定された開始位置にあるか、指定された範囲のすべての位置にある 1 つ以上の空白以外の文字(特殊文字を含む) を含んでいる必要があります。 ◦ 7 - (**行で検索**) 一致するレコードには、指定された文字または文字列が、指定された位置で始まる、指定された開始行に含まれている必要があります。 ◦ 8 - (**範囲で検索**) 一致するレコードには、 [テキスト]フィールドにある文字または文字列が、指定された範囲の、レコードの指定された行の指定された範囲のどこかに含まれている必要があります。 ◦ 10 - (**独自のマップ**) 一致するレコードには、指定された文字パターンと一致する文字が、指定された位置で始まる、指定された開始行に含まれている必要があります。
AT *開始行*, *開始位置*, *範囲*	◦ ***開始行*** -条件が適用されるレコードの行 たとえば、独自のマップを使用して郵便番号と一致させ、郵便番号が 3 行レコードの 3 行目に表示される場合は、 [*開始行*]で 3 を指定する必要があります。 **メモ** 単一行のレコードの場合、*開始行*の値は常に 1 です。 ◦ ***開始位置*** -条件と比較する印刷イメージ ファイルの開始バイト位置 ◦ ***範囲*** -条件との比較で使用する印刷イメージ ファイルの開始バイト位置からのバイト数 範囲なしで開始バイト位置のみを使用している場合は、*範囲*に 0 を指定します。 **メモ** 非 Unicode 版 Analytics: 1 バイト = 1 文字 Unicode 版 Analytics、拡張 ASCII (ANSI) データ: 1 バイト = 1 文字 Unicode 版 Analytics、Unicode データ: 2 バイト = 1 文字 Unicode データの場合、*範囲*は、偶数バイト数である必要があります。たとえば、50,59 (10 バイト) です。奇数バイト数を指定すると、条件に対する正しい一致が行われなくなる可能性があります。
論理	条件間の論理関係: ◦ 0 - (**And**) 現在および次の条件は、論理 AND に関連しています ◦ 1 - (**Or**) 現在および次の条件は、論理 OR に関連しています ◦ 4 - (**新しいグループ** > **And**) 現在の条件は論理条件のグループの最後です。現在のグループと次のグループは、論理 AND で関連付けられます。 ◦ 5 - (**新しいグループ** > **Or**) 現在の条件は論理条件のグループの最後です。現在のグループと次のグループは、論理 OR で関連付けられます。 ◦ 7 - (**終了**) 現在の条件は論理条件のグループの最後です。
テキスト	次に対して一致するリテラルまたはワイルドカード文字: ◦ **[正確に一致する]**、 **[行で検索]**、または **[範囲で検索]** -の場合、印刷イメージ ファイルのレコードのセットを一意に識別する文字または文字列を指定します。 ◦ **[独自のマップ]** -の場合、印刷イメージ ファイルのレコードのセットを一意に識別する文字パターンを指定します。 **カスタム マップ**オプションは、"MAP( ) 関数" ページ 624と同じ構文を使用します。

名前	説明
	他の一致タイプについては、テキストは空の文字列 "" です。

FIELD パラメーター

フィールド定義情報。

名前	説明
FIELD *名前 型*	インポートするソース データ ファイル内の個別フィールドの名前およびデータ型。フィールドをインポート対象から除外する場合は、そのフィールドを指定しないでください。 *型*については、"フィールド データ型の識別子" ページ 292を参照してください。
AT *開始行*, *開始位置*	◦ ***開始行*** -印刷イメージ ファイルのレコードのフィールドの開始行 印刷イメージ ファイルの複数行のレコードの場合、*開始行*はレコードの任意の行でフィールドを開始できます。*レコードの行*が 1 の場合、*開始行*は常に 1 です。 ◦ ***開始位置*** - 印刷イメージ ファイルのフィールドの開始バイト位置 **メモ** 非 Unicode 版 Analytics: 1 バイト = 1 文字 Unicode 版 Analytics、拡張 ASCII (ANSI) データ: 1 バイト = 1 文字 Unicode 版 Analytics、Unicode データ: 2 バイト = 1 文字
SIZE *長さ*, *フィールドの行*	◦ ***長さ*** - Analytics テーブル レイアウトにおけるフィールドの長さ(バイト数) **メモ** 非 Unicode 版 Analytics: 1 バイト = 1 文字 Unicode 版 Analytics、拡張 ASCII (ANSI) データ: 1 バイト = 1 文字 Unicode 版 Analytics、Unicode データ: 2 バイト = 1 文字 Unicode データでは、偶数バイトのみを指定します。奇数バイトを指定すると、文字が正しく表示されない可能性があります。 ◦ ***lフィールドの行*** -印刷イメージ ファイルの単一フィールド値が占める行数 ファイルのデータと一致する単一行または複数行のフィールドを定義できます。 **メモ** フィールドに指定された行数は、フィールドを含むレコードに指定された行数を超えることができません。
DEC *値*	数値フィールドの小数点以下の桁数

名前	説明
WID *バイト*	フィールドの表示幅(バイト数)。 指定した値によって、Analytics のビューおよびレポートにおけるフィールドの表示幅が決まります。表示幅はデータを変更するものではありませんが、表示幅がフィールド長より短い場合にはデータが隠れる可能性があります。
PIC *書式*	**メモ** 数値フィールドまたは日付時刻フィールドにのみ適用されます。 ◦ **数値フィールド** - Analytics のビューとレポートに含まれる数値の表示形式。 ◦ **日付時刻フィールド** -ソースデータの日付時刻値の物理形式(日付時刻文字、区切り文字の順など) **メモ** 日付時刻フィールドの場合、*形式*はソースデータの物理形式と正確に一致する必要があります。たとえば、ソースデータが 12/31/2014 である場合は、書式を "MM/DD/YYYY" として入力します。 *書式*は引用符で囲む必要があります。
AS *表示名*	新しい Analytics テーブルのビューにおけるフィールドの表示名(代替列見出し)。 *表示名*の値は引用符で囲まれた文字列。列見出しを改行したい場合は、語句の間にセミコロン(;)を入れます。 フィールドの定義時には、AS は必須です。表示名をフィールド名と同じにしたい場合は、空の*表示名*を入力します。つまり、次の構文を使用します。AS "" 2 つの二重引用符の間にスペースがないことを確認してください。

例

印刷イメージ(レポート)ファイルからデータをインポートする

印刷イメージ(レポート)ファイル `Report.txt` からデータをインポートします。

1 件のヘッダーレコード、詳細レコードの1 セット、および 5 つのフィールドが、結果の Analytics テーブル **Inventory_report** に作成されます。

```
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"
```

備考

> **メモ**
>
> このコマンドの動作の詳細については、Analytics のヘルプを参照してください。

フィールド データ型の識別子

以下の表は、FIELD でデータ*型*を指定するときに使用する必要がある文字の一覧を示します。各文字はデータ型に対応します。

たとえば、文字データ型を使用する姓フィールドを定義する場合は、"C": FIELD "Last_Name" C と指定します。

> **メモ**
>
> **データ定義ウィザード**を使用してEBCDIC、Unicode、またはASCIIフィールドを含むテーブルを定義する場合、それらのフィールドには自動的に文字 "C"(CHARACTER 型) が割り当てられます。
>
> IMPORT ステートメントを手作業で入力するか、既存の IMPORT ステートメントを編集する場合、EBCDIC または Unicode フィールドに対して、より特有の文字 "E" または "U" に置き換えることができます。

文字	データ型
A	ACL
B	BINARY
C	CHARACTER
D	DATETIME
E	EBCDIC
F	FLOAT
G	ACCPAC
I	IBMFLOAT
K	UNSIGNED
L	LOGICAL
N	PRINT
P	PACKED

文字	データ型
Q	BASIC
R	MICRO
S	CUSTOM
T	PCASCII
U	UNICODE
V	VAXFLOAT
X	NUMERIC
Y	UNISYS
Z	ZONED

IMPORT SAP コマンド

Direct Link を使用して、SAP システムからデータをインポートし、Analytics テーブルを作成します。

構文

IMPORT SAP PASSWORD *番号* TO *テーブル名* SAP SOURCE "SAP AGENT" *インポートの詳細*

パラメーター

名前	説明
PASSWORD *番号*	使用するパスワード定義。 実際のパスワードを入力することを求めたり指定したりするのには、PASSWORD *番号* の構文は使用しません。パスワード定義とは、以前に PASSWORD コマンドか SET PASSWORD コマンドを使用して入力または設定されたパスワードのことです。 *番号* はパスワード定義の番号です。たとえば、以前に 2 つのパスワードをスクリプトで設定したり入力したりしている場合には、PASSWORD 2 により、2 番目のパスワードを使用することを指定するなどします。 パスワードの入力または設定の詳細については、次を参照してください。 • "PASSWORD コマンド" ページ 345 • "SET コマンド" ページ 404 • PASSWORD アナリティクス タグ パスワードの入力または設定の詳細については、次を参照してください。 • PASSWORD コマンド • SET コマンド • PASSWORD アナリティクス タグ **メモ** このパスワードは、SAP システムにアクセスするのに使用します。
TO *テーブル名*	データをインポートする Analytics テーブルの名前。 **メモ** テーブルの名前は、64 文字までの英数字に制限されます。名前にはアンダースコア文字 (_) を使用できますが、他の特殊文字やスペースは使用できません。名前の先頭を数字にすることはできません。
SAP SOURCE "SAP AGENT"	SAP データをインポートするために必要です。"SAP AGENT" が選択可能な唯一の選択肢です。
インポートの詳細	クエリの詳細。<q></q> タグで囲む必要があります。この文字列では、"Direct Link クエリタグ" ページ 297に記載されているタグを使用してクエリを定義します。

名前	説明
	この値の物理的なサイズは最大 16 KB です。

例

複数のテーブルに対するクエリーを実行する

この例では、IMPORT SAP コマンドを使用して、複数のテーブルに対するクエリを実行します。

有効なクエリ文字列を作成するには、タグの順序と入れ子が正確であることが必要です。例内のタグの順序と入れ子は、正確です。この例は、必要とされる IMPORT SAP クエリタグの順序と入れ子を判断するのに使用してください。

> **メモ**
>
> この例は、読みやすくなるように複数の行に分けて書かれています。ただし、スクリプトでは、コマンドとクエリ文字列は改行せずに入力する必要があります。

> **ヒント**
>
> IMPORT SAP クエリ文字列の構文は通常、複雑です。IMPORT SAP コマンドとクエリ文字列をスクリプトに追加する最良の方法は、Analytics の [**ログ**]タブから既存の IMPORT SAP コマンドをコピーして、必要に応じてクエリタグを編集することです。

```
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>購買伝票ヘッダ</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>購買伝票明細</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>
```

備考

IMPORT SAP コマンドは、Direct Link がインストールおよび設定されている場合にのみサポートされます。

"Direct Link クエリタグ" 下に記載されているテーブルには、*import_details* パラメーターに指定できるタグの一覧が示されています。**必須**列では、以下の値を使用して、各タグの指定がどのような場合に必須であるかが示されています。

- **Y** - 必須
- **N** - オプション
- **M** - 複数テーブルのクエリの場合のみ必須
- **B** - 必須。ただし、値を渡すことはできません
- **W** - フィルターを利用する場合のオプション
- **S** - スケジュールモードが指定されるときに必須

Direct Link クエリタグ

名前	タグ	必須	説明
テーブル別名	<a>	M	クエリ内で一意にテーブルを識別する別名。これにより、同じテーブルを 2 回以上使用できるようになります。 6 文字を上限とします。
すべての行	<ar>	Y	すべての一致する行がクエリの結果セットの一部として返されることを示します。 有効な値は次のとおりです。 **1** - <r> タグで指定されたレコード数を上書きします(最大行) **0** - <r> タグで指定されたレコード数を返します(最大行) このタグは、常に <r></r> タグの後に指定します。
クライアント	<c>	N	SAP システム内のクライアント。
子テーブル別名	<ca>	M	子テーブルの別名。
子テーブル フィールド	<cf>	M	結合条件に基づく子テーブル内のフィールド。
クライアント ファイル名	<cf>	Y	クエリの結果が格納されるクライアント システム上のターゲット ファイルを識別します。
子テーブル	<ct>	M	結合条件の子テーブル。
宛先	<d>	N	SAP システムを特定するために使用する、SAP RFC ライブラリファイル (`saprfc.ini`) 内の宛先を指定します。

名前	タグ	必須	説明
データ長	<dl>	B	各行内の文字の数。これには、レコードの終わりを示すキャリッジ リターン文字とライン フィード文字(CR+LF、つまり 16 進数の文字 0D+0A) が含まれます。
日付	<dt>	S	スケジュール モードを使用するときに必須です。SAP ジョブを実行する時間を指定します。 YYYYMMDD の形式である必要があります。たとえば、2014 年 12 月 31 日は、20140131 と指定する必要があります。
予想される行数	<e>	B	クエリが返すと予想される行数。
フィールド名	<f>	Y	ネイティブ フィールド名。
フィルター フィールド	<f>	W	フィルターを適用するネイティブ フィールド名。
フィールド	<fs>	Y	クエリ結果の一部として返される、テーブル内のフィールドの一覧。
上限値	<h>	W	Between 演算子を使用するときに上限値を格納します。他の演算子を使用している場合は無視されます。
結合条件	<jc>	M	結合条件。
ジョブ数	<jcount>	B	バックグラウンド モードのクエリを識別するために、SAP によって内部的に使用されます。
ジョブ名	<jname>	B	バックグラウンド モードのクエリを識別するために、SAP によって内部的に使用されます。
結合関係	<js>	Y	クエリ内でテーブルをリンクするための結合条件の一覧。
結合切り替え	<jw>	N	結合切り替え列挙型に等価な数値。 有効な値は次のとおりです。 **0** - 内部結合 **1** - 左外部結合
下限値	<l>	W	Between 演算子を使用するときには下限値、他の演算子を使用するときには値を格納します。
言語	<lg>	Y	SAP データベースのフィールドのロケールを決定するために使用される言語識別子。
モード	<m>	Y	サブミッション モード列挙型に等価な数値。 有効な値は次のとおりです。 **0** -即時に抽出 **1** - バックグラウンド **2** - スケジュール済み

名前	タグ	必須	説明
テーブル名	<n>	Y	ネイティブ テーブル名。
演算子	<o>	W	演算子列挙型に等価な数値。 有効な値は次のとおりです。 **0** - - 等しい(=) **1** - - 等しくない(<>) **2** - - 未満 (<) **3** - - 以下(<=) **4** - - より大きい(>) **5** - - 以上(>=) **6** - - 間 **7** - - 含む
親テーブル別名	<pa>	M	親テーブルの別名。
親テーブル フィールド	<pf>	M	結合条件に基づく親テーブル内のフィールド。
親テーブル	<pt>	M	結合条件の親テーブル。
クエリ	<q>	Y	クエリを囲みます。
最大行数	<r>	Y	クエリが返す行の最大数。
選択済み	<s>	Y	<s> タグが <f> タグの下に現れる場合、それはフィールドがクエリの結果セットの一部として返されるかどうか示します。
システム	<s>	Y	<s> タグが <q> タグの下に現れる場合は、このクエリが使用されるシステムの種類を識別します(現在のところ、SAP のみがサポートされます) 。
サーバー ファイル名	<sf>	B	バックグラウンド モード クエリの結果を保持するサーバー上のファイルを識別します。
サーバー グループ名	<sg>	N	サーバー グループの名前。最大 20 文字。
サーバー名	<sn>	N	サーバーの名前。最大 20 文字。
テーブル	<t>	Y	テーブル。
テーブル記述	<td>	Y	SAP データ ディクショナリーのテーブル記述。常に <a> タグの下に現れます。
時間	<tm>	S	スケジュール モードを使用するときに必須です。SAP ジョブを実行する時間を指定します。 hhmmss の形式である必要があります。たとえば、2:30 pm は 143000 と指定する必要があります。

名前	タグ	必須	説明
テーブル	<ts>	Y	クエリがデータを抽出するテーブルの一覧。
テーブル タイプ	<tt>	Y	SAP テーブルのタイプ。 有効な値は次のとおりです。 **0** - クラスター **1** - 透明 **2** - プール **3** - ビュー
ユーザー名	<u>	N	ユーザーのログオン名。
フィルター	<w>	W	テーブルのデータに適用されるフィルター。
フィルター	<wc>	W	テーブル内に含まれるデータに適用されるフィルターの一覧。
フィルター切り替え	<ws>	N	フィルター切り替え列挙型に等価な数値。 有効な値は次のとおりです。 **0** - (Or) および (Or) **1** - (And) または (And)

IMPORT XBRL コマンド

XBRL ファイルを定義およびインポートして、Analytics テーブルを作成します。

構文

IMPORT XBRL TO *テーブル インポート ファイル名* FROM *ソース ファイル名* CONTEXT *コンテキスト名* <...*n*> [フィールド構文] <...*n*> <IGNORE *フィールド番号*> <...*n*>

フィールド構文 ::=
FIELD *名前 型* AT *開始位置* DEC *値* WID *バイト* PIC *書式* AS *表示名*

パラメーター

名前	説明
TO *テーブル*	データをインポートする Analytics テーブルの名前。 **メモ** テーブルの名前は、64 文字までの英数字に制限されます。名前にはアンダースコア文字(_) を使用できますが、他の特殊文字やスペースは使用できません。名前の先頭を数字にすることはできません。
インポート ファイル名	作成する Analytics データ ファイルの名前。 *インポート ファイル名*、.FIL ファイル拡張子を持つ引用符で囲まれた文字列として指定します。例: "Invoices.FIL". デフォルトでは、データ ファイル(.FIL) は、Analytics プロジェクトが入っているフォルダーに保存されます。 既存の異なるフォルダーにデータ ファイルを保存するには、絶対または相対ファイル パスを指定します。 ◦ "C:\data\Invoices.FIL" ◦ "data\Invoices.FIL"
FROM *ソースファイル名*	ソース データ ファイルの名前。*ソース ファイル名*は引用符で囲む必要があります。 ソース データ ファイルが Analytics プロジェクトと同じフォルダーに位置しない場合、ファイルの位置を指定するために絶対パスまたは相対パスを使用する必要があります。 ◦ "C:\data*ソース ファイル名*" ◦ "data*ソース ファイル名*"
CONTEXT *コンテキスト名*	テーブルを定義する XBRL コンテキスト。1 つ以上のコンテキストを指定する場合、コンテキストはすべて同じタイプ(Instant、Period、または Forever) である必要があります。

名前	説明
FIELD *名前 型*	インポートするソース データ ファイル内の個別フィールドの名前およびデータ型。フィールドをインポート対象から除外する場合は、そのフィールドを指定しないでください。 *型*については、"フィールド データ型の識別子" 見開きページを参照してください。
AT *開始位置*	Analytics データ ファイル内のフィールドの開始バイトを指定します。 **メモ** 非 Unicode 版 Analytics: 1 バイト = 1 文字 Unicode 版 Analytics: 2 バイト = 1 文字 Unicode 版 Analytics では、一般的に、奇数で開始するバイト位置を指定してください。偶数の開始位置を指定すると、文字が正しく表示されない可能性があります。
DEC *値*	数値フィールドの小数点以下の桁数
WID *バイト*	Analytics テーブル レイアウトにおけるフィールドの長さ(バイト数) **メモ** 非 Unicode 版 Analytics: 1 バイト = 1 文字 Unicode 版 Analytics: 2 バイト = 1 文字 Unicode 版 Analytics では、偶数バイトのみを指定します。奇数バイトを指定すると、文字が正しく表示されない可能性があります。
PIC *書式*	**メモ** 数値フィールドまたは日付時刻フィールドにのみ適用されます。 ◦ **数値フィールド** - Analytics のビューとレポートに含まれる数値の表示形式。 ◦ **日付時刻フィールド** -ソース データの日付時刻値の物理形式(日付時刻文字、区切り文字の順など) **メモ** 日付時刻フィールドの場合、*形式*はソース データの物理形式と正確に一致する必要があります。たとえば、ソース データが 12/31/2014 である場合は、書式を "MM/DD/YYYY" として入力します。 *書式*は引用符で囲む必要があります。
AS *表示名*	新しい Analytics テーブルのビューにおけるフィールドの表示名(代替列見出し) 。 *表示名*の値は引用符で囲まれた文字列。列見出しを改行したい場合は、語句の間にセミコロン(;) を入れます。 フィールドの定義時には、AS は必須です。表示名をフィールド名と同じにしたい場合は、空の*表示名*を入力します。つまり、次の構文を使用します。AS "" 2 つの二重引用符の間にスペースがないことを確認してください。
IGNORE *フィールド番号*	テーブル レイアウトからフィールドを除外します。

名前	説明
省略可能	*フィールド番号*は、ソース データにおけるフィールドの位置を指定します。たとえば、IGNORE 5 は、ソース データの 5 番目のフィールドを Analytics テーブル レイアウトから除外します。

例

XBRL ファイルを Analytics テーブルにインポートする

次の例は、XBRL ファイルの **Current_AsOf** コンテキストから **Financials** という Analytics テーブルにデータをインポートしています。

```
IMPORT XBRL TO Financials "Financials.fil" FROM "FinancialStatemenXBRL.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
```

備考

> **メモ**
>
> このコマンドの動作の詳細については、Analytics のヘルプを参照してください。

フィールド データ型の識別子

以下の表は、FIELD でデータ*型*を指定するときに使用する必要がある文字の一覧を示します。各文字はデータ型に対応します。

たとえば、文字データ型を使用する姓フィールドを定義する場合は、"C": FIELD "Last_Name" C と指定します。

> **メモ**
>
> **データ定義ウィザード**を使用して EBCDIC、Unicode、または ASCII フィールドを含むテーブルを定義する場合、それらのフィールドには自動的に文字 "C"(CHARACTER 型) が割り当てられます。
>
> IMPORT ステートメントを手作業で入力するか、既存の IMPORT ステートメントを編集する場合、EBCDIC または Unicode フィールドに対して、より特有の文字 "E" または "U" に置き換えることができます。

文字	データ型
A	ACL
B	BINARY

文字	データ型
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 XML コマンド

XML ファイルを定義およびインポートして、Analytics テーブルを作成します。

構文

IMPORT XML TO *テーブル インポート ファイル名* FROM *ソースファイル名* [フィールド 構文] <...*n*>

フィールド 構文 ::=
FIELD *名前 型* AT *開始位置* DEC *value* WID *バイト* PIC *書式* AS *表示名* RULE *xpath 式*

パラメーター

名前	説明
TO *テーブル*	データをインポートする Analytics テーブルの名前。 **メモ** テーブルの名前は、64 文字までの英数字に制限されます。名前にはアンダースコア文字(_) を使用できますが、他の特殊文字やスペースは使用できません。名前の先頭を数字にすることはできません。
インポート ファイル名	作成する Analytics データ ファイルの名前。 *インポート ファイル名*、.FIL ファイル拡張子を持つ引用符で囲まれた文字列として指定します。例: "Invoices.FIL". デフォルトでは、データ ファイル(.FIL) は、Analytics プロジェクトが入っているフォルダーに保存されます。 既存の異なるフォルダーにデータ ファイルを保存するには、絶対または相対ファイル パスを指定します。 ◦ "C:\data\Invoices.FIL" ◦ "data\Invoices.FIL"
FROM *ソースファイル名*	ソース データ ファイルの名前。*ソース ファイル名* は引用符で囲む必要があります。 ソース データ ファイルが Analytics プロジェクトと同じフォルダーに位置しない場合、ファイルの位置を指定するために絶対パスまたは相対パスを使用する必要があります。 ◦ "C:\data*ソース ファイル名*" ◦ "data*ソース ファイル名*"
FIELD *名前 型*	インポートするソース データ ファイル内の個別フィールドの名前およびデータ型。フィールドをインポート対象から除外する場合は、そのフィールドを指定しないでください。 *型* については、"フィールド データ型の識別子" ページ 307を参照してください。

名前	説明
AT *開始位置*	Analytics データ ファイル内のフィールドの開始バイトを指定します。 **メモ** 非 Unicode 版 Analytics: 1 バイト = 1 文字 Unicode 版 Analytics: 2 バイト = 1 文字 Unicode 版 Analytics では、一般的に、奇数で開始するバイト位置を指定してください。偶数の開始位置を指定すると、文字が正しく表示されない可能性があります。
DEC *値*	数値フィールドの小数点以下の桁数
WID *バイト*	Analytics テーブル レイアウトにおけるフィールドの長さ(バイト数) **メモ** 非 Unicode 版 Analytics: 1 バイト = 1 文字 Unicode 版 Analytics: 2 バイト = 1 文字 Unicode 版 Analytics では、偶数バイトのみを指定します。奇数バイトを指定すると、文字が正しく表示されない可能性があります。
PIC *書式*	**メモ** 数値フィールドまたは日付時刻フィールドにのみ適用されます。 ◦ **数値フィールド** - Analytics のビューとレポートに含まれる数値の表示形式。 ◦ **日付時刻フィールド** -ソース データの日付時刻値の物理形式(日付時刻文字、区切り文字の順など) **メモ** 日付時刻フィールドの場合、*形式*はソース データの物理形式と正確に一致する必要があります。たとえば、ソース データが 12/31/2014 である場合は、書式を "MM/DD/YYYY" として入力します。 *書式*は引用符で囲む必要があります。
AS *表示名*	新しい Analytics テーブルのビューにおけるフィールドの表示名(代替列見出し) 。 *表示名*の値は引用符で囲まれた文字列。列見出しを改行したい場合は、語句の間にセミコロン(;) を入れます。 フィールドの定義時には、AS は必須です。表示名をフィールド名と同じにしたい場合は、空の*表示名*を入力します。つまり、次の構文を使用します。AS "" 2 つの二重引用符の間にスペースがないことを確認してください。
RULE *xpath式*	xpath 式は XML ファイルからフィールド内容を選択するために使用されます。 xpath は XML ファイルのデータにアクセスする標準的な方法です。たとえば、acct/title/text() は XML ファイルの <title> タグ内のテキストを取得します。

例

XML ファイルのデータを Analytics テーブルにインポートする

次の例では、XML ファイルのデータを **Employees** という名前の Analytics テーブルにインポートしています。

```
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()"
```

備考

> **メモ**
>
> このコマンドの動作の詳細については、Analytics のヘルプを参照してください。

フィールド データ型の識別子

以下の表は、FIELD でデータ*型*を指定するときに使用する必要がある文字の一覧を示します。各文字はデータ型に対応します。

たとえば、文字データ型を使用する姓フィールドを定義する場合は、"C": FIELD "Last_Name" C と指定します。

> **メモ**
>
> **データ定義ウィザード**を使用して EBCDIC、Unicode、または ASCII フィールドを含むテーブルを定義する場合、それらのフィールドには自動的に文字 "C"(CHARACTER 型) が割り当てられます。
>
> IMPORT ステートメントを手作業で入力するか、既存の IMPORT ステートメントを編集する場合、EBCDIC または Unicode フィールドに対して、より特有の文字 "E" または "U" に置き換えることができます。

文字	データ型
A	ACL
B	BINARY
C	CHARACTER
D	DATETIME

文字	データ型
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

INDEX コマンド

物理的ではなく連続的な順番でレコードへのアクセスを可能にする、Analytics テーブルのインデックスを作成します。

構文

INDEX <ON> {*キーフィールド* <D> <...*n*>|ALL} TO *ファイル名* <IF *テスト*> <WHILE *テスト*> <FIRST *範囲*|NEXT *範囲*> <OPEN> <ISOLOCALE *ロケールコード*>

パラメーター

名前	説明
ON *キーフィールド* D <...*n*> | ALL	インデックスで使用するキーフィールドまたはフィールド、または式。 演算フィールドや一時的に作成した式など、データ型に関係なく、あらゆる種類のフィールドのインデックスを作成することができます。 ◦ ***キーフィールド*** -では、指定した 1 つまたは複数のフィールドが使用されます。 複数のフィールドでインデックス作成を行う場合は、テーブルは入れ子でインデックスが作成されます。入れ子でのフィールド間の順序は、フィールドを指定した順になります。 キーフィールドに降順にインデックスを作成する D を含めます。デフォルトのインデックス順は昇順です。 ◦ **ALL** - では、テーブル内のすべてのフィールドが使用されます。 テーブル内のすべてのフィールドを基準にしてインデックス作成を行うと、入れ子でインデックスが作成されます。入れ子でのフィールド間の順序は、フィールドを指定した順になります。 ALL では昇順でしかインデックスを作成することができません。
TO *ファイル名*	インデックスと関連付けられたインデックスファイルの名前。インデックスファイルは .INX 拡張子で作成されます。 **メモ** Analytics のユーザーインターフェイスでは、インデックスの名前は、64 文字までの英数字に制限されます。名前にはアンダースコア文字(_) を使用できますが、他の特殊文字やスペースは使用できません。名前の先頭を数字にすることはできません。
IF *テスト* 省略可能	各レコードを処理するために真である必要がある条件式。コマンドは、その条件を満たすレコードに対してのみ実行されます。 **メモ** IF パラメーターは、任意の範囲パラメーター(WHILE、FIRST、NEXT) が適用された後に、テーブルに残るレコードに対してのみ評価されます。

名前	説明
WHILE *テスト* 省略可能	各レコードを処理するために真である必要がある条件式。条件がfalseと評価するか、テーブルの最後に達したら、コマンドは実行を中止します。 **メモ** WHILEをFIRSTまたはNEXTとともに使用する場合は、1つの制限に達するとすぐに、レコードの処理が停止します。
FIRST *範囲* | NEXT *範囲* 省略可能	処理するレコード数： ◦ **FIRST** - 指定されたレコード数に達するまで、最初のレコードから処理を開始します ◦ **NEXT** - 指定されたレコード数に達するまで、現在選択されているレコードから処理を開始します *範囲*は処理するレコード数を指定します。 FIRSTとNEXTを省略すると、すべてのレコードがデフォルトで処理されます。
OPEN 省略可能	テーブルを開き、インデックスをテーブルに適用します。
ISOLOCALE *ロケールコード* 省略可能	**メモ** Analyticsの Unicode 版にのみ適用されます。 システムロケールは「*言語-国*」の形式で入力します。たとえば、カナダフランス語はコード「fr_ca」を入力します。 次のコードを使用します。 ◦ **言語** - ISO 639 標準言語コード ◦ **国** - ISO 3166 標準国コード 国コードを指定しない場合は、言語のデフォルト国が使用されます。 ISOLOCALEを使用しない場合は、デフォルト システム ロケールが使用されます。

例

メモ

このコマンドの動作の詳細については、Analytics のヘルプを参照してください。

インデックスを作成し、テーブルを開く

業者テーブルで、**業者市区郡**フィールドにインデックスを作成し、テーブルを開きます。

```
OPEN Vendor
INDEX ON Vendor_City to "CityIndex" OPEN
```

インデックスを作成し、テーブルに適用する

業者テーブルで、**業者市区郡**フィールドにインデックスを作成します。後から、インデックスをテーブルに適用し

ます。

```
OPEN Vendor
INDEX ON Vendor_City to "CityIndex"
.
.
.
SET INDEX TO "CityIndex"
```

JOIN コマンド

2 つの Analytics テーブルのフィールドを連結して 1 つの新しい Analytics テーブルにします。

> **メモ**
>
> 曖昧一致を使用して、テーブルを結合するには、"FUZZYJOIN コマンド" ページ 207を参照してください。

構文

JOIN PKEY *主テーブルのキーフィールド*|PKEY ALL} {FIELDS *主テーブルのフィールド*|FIELDS ALL} {SKEY *副テーブルのキーフィールド* |SKEY ALL} <WITH *副テーブルのフィールド*|WITH ALL> {*キーワード なし*|MANY|UNMATCHED|PRIMARY|SECONDARY|PRIMARY SECONDARY} <IF *テスト*> TO *テーブル名* <LOCAL> <OPEN> <WHILE *テスト*> <{FIRST *範囲*|NEXT *範囲*> <APPEND> <PRESORT> <SECSORT> <ISOLOCALE *ロケールコード*>

パラメーター

名前	説明
PKEY *主テーブルのキーフィールド* | PKEY ALL	主テーブルのキーフィールド (複数可) または式。 ◦ ***主テーブルのキーフィールド*** - 指定されたフィールドを使用します ◦ **ALL** - テーブルのすべてのフィールドを使用します。
FIELDS *主フィールド* | FIELDS ALL	結合先の出力テーブルに含める、主テーブル内のフィールドまたは式。 ◦ ***主フィールド*** - 指定した 1 つまたは複数のフィールドが含まれます。 ◦ **ALL** - テーブルのすべてのフィールドを含めます **メモ** 結合テーブルに含める場合は、主キーフィールドを明示的に指定する必要があります。ALL を指定すると含まれます。
SKEY *副テーブルのキーフィールド* | SKEY ALL	副テーブルのキーフィールド (複数可) または式。 ◦ ***副テーブルのキーフィールド*** - use the specified field or fields ◦ **ALL** - テーブルのすべてのフィールドを使用します。
WITH *副フィールド* | WITH ALL 省略可能	結合先の出力テーブルに含める、副テーブル内のフィールドまたは式。 ◦ ***副フィールド*** - 指定した 1 つまたは複数のフィールドが含まれます。 ◦ **ALL** - テーブルのすべてのフィールドを含めます

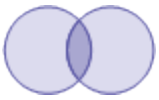
<table>
<tr><th>名前</th><th>説明</th></tr>
<tr><td></td><td>

> **メモ**
>
> 結合テーブルに含める場合は、副キー フィールドを明示的に指定する必要があります。ALL を指定すると含まれます。
>
> 結合タイプ "UNMATCHED" を使用する場合には、"WITH" は指定できません。

</td></tr>
<tr><td>キーワードなし | MANY | UNMATCHED | PRIMARY | SECONDARY | PRIMARY SECONDARY</td><td>

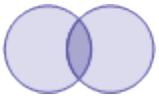
実行する結合の種類。

キーワードなし(結合タイプのどのキーワードも指定しない)



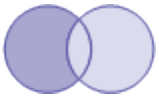
結合された出力テーブルの内容：	[結合]ダイアログ ボックスの対応するオプション
◦ すべての一致した主レコードおよび最初に一致した副レコード	**一致した主レコードおよび副レコード (一致する 1 件目の副レコード)**

MANY



結合された出力テーブルの内容：	[結合]ダイアログ ボックスの対応するオプション
◦ すべての一致した主レコードおよびすべての一致した副レコード ◦ 主および副テーブル間の各一致に 1 つのレコード	**一致した主レコードおよび副レコード (重複する副レコードのすべての一致)**

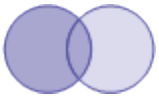
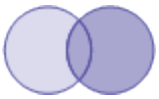
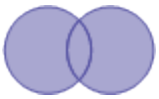
UNMATCHED



結合された出力テーブルの内容：	[結合]ダイアログ ボックスの対応するオプション
◦ 不一致の主レコード	**不一致の主レコード**

PRIMARY

</td></tr>
</table>

<table>
<tr><th>名前</th><th>説明</th></tr>
<tr><td></td><td>

<table>
<tr><th>結合された出力テーブルの内容：</th><th>[結合]ダイアログ ボックスの対応するオプション</th></tr>
<tr><td>◦ すべての主レコード(一致と不一致の両方)、最初に一致した服レコード</td><td>すべての主レコードと、キーに一致する副レコード</td></tr>
</table>
メモ
キーワード、BOTH は、PRIMARY を指定することと同じです。
<h3>SECONDARY</h3>

<table>
<tr><th>結合された出力テーブルの内容：</th><th>[結合]ダイアログ ボックスの対応するオプション</th></tr>
<tr><td>◦ すべての副レコード(一致と不一致)およびすべての一致する主レコード
重複する副一致の最初のインスタンスのみが主レコードに結合されます。</td><td>すべての副レコードと、キーに一致する主レコード</td></tr>
</table>
<h3>PRIMARY SECONDARY</h3>

<table>
<tr><th>結合された出力テーブルの内容：</th><th>[結合]ダイアログ ボックスの対応するオプション</th></tr>
<tr><td>◦ すべての主および副レコード(一致と不一致)
重複する副一致の最初のインスタンスのみが主レコードに結合されます。</td><td>すべての主レコードおよび副レコード</td></tr>
</table>
</td></tr>
<tr><td>IF テスト
省略可能</td><td>各レコードを処理するために真である必要がある条件式。コマンドは、その条件を満たすレコードに対してのみ実行されます。
メモ
IF パラメーターは、任意の範囲パラメーター(WHILE、FIRST、NEXT)が適用された後に、テーブルに残るレコードに対してのみ評価されます。</td></tr>
</table>

名前	説明
	メモ ほとんどの結合タイプで、IF 条件が主テーブルにのみ適用されます。 例外は多対多結合で、IF 条件が副テーブルも参照できます。
TO *テーブル名*	コマンドの結果を送信する場所： ◦ ***テーブル名*** -は、結果の保存先となる Analytics テーブルのことです。 *テーブル名*には、ファイル拡張子 .FIL を付けた文字列を引用符で囲んで指定する必要があります。例：TO "Output.FIL" デフォルトでは、テーブル データ ファイル(.fil) は、Analytics プロジェクトが入っているフォルダーに保存されます。 既存の異なるフォルダーにデータ ファイルを保存するには、絶対または相対ファイル パスを指定します。 • TO "C:\Output.FIL" • TO "Results\Output.FIL" **メモ** テーブル名は 64 文字の英数字(.FIL 拡張子を含まない) に制限されています。名前にはアンダースコア文字(_) を使用できますが、他の特殊文字やスペースは使用できません。名前の先頭を数字にすることはできません。
LOCAL 省略可能	Analytics プロジェクトと同じ場所に出力ファイルを保存します。 **メモ** Analytics テーブルである出力ファイルを含むサーバー テーブルに対してコマンドを実行するときにのみ適用されます。
OPEN 省略可能	コマンドを実行した後、コマンドによって作成されたテーブルを開きます。コマンドが出力テーブルを作成する場合にのみ有効です。
WHILE *テスト* 省略可能	各レコードを処理するために真である必要がある条件式。条件が false と評価するか、テーブルの最後に達したら、コマンドは実行を中止します。 **メモ** WHILE を FIRST または NEXT とともに使用する場合は、1 つの制限に達するとすぐに、レコードの処理が停止します。
FIRST *範囲* | NEXT *範囲* 省略可能	処理するレコード数： ◦ **FIRST** - 指定されたレコード数に達するまで、最初のレコードから処理を開始します ◦ **NEXT** - 指定されたレコード数に達するまで、現在選択されているレコードから処理を開始します *範囲*は処理するレコード数を指定します。 FIRST と NEXT を省略すると、すべてのレコードがデフォルトで処理されます。
APPEND 省略可能	コマンドの出力を既存ファイルに上書きしないで、そのファイルの末尾に追加します。

名前	説明
	メモ コマンドの出力と既存のファイルの構造が同一であるようにする必要があります。 • 同じフィールド • 同じフィールド順序 • 一致するフィールドが同じ長さ • 一致するフィールドが同じデータ型 出力は、既存ファイルとの間でファイル構造が違っている場合でも、Analytics によって既存ファイルに追加されます。出力と既存のファイルの構造が一致しない場合は、データが混在、不足、不正確になります。
PRESORT 省略可能	コマンドを実行する前に主キーフィールドで主テーブルを並べ替えます。 **メモ** GROUP コマンドの内部では PRESORT を使用することができません。 **並べ替えの代わりのインデックス** 主テーブルは、並べ替える代わりにインデックスを付けることができます。大きなテーブルでは並べ替えよりインデックス作成を行った方が、テーブルの結合に必要な時間を短くすることができます。 また、インデックス付きの共通のキーフィールドを使用して 2 つのテーブルを結合する場合は、PRESORT と SECSORT を省略します。
SECSORT 省略可能	コマンドを実行する前に副テーブルのキーフィールドで副テーブルを並べ替えます。 **メモ** SECSORT は GROUP コマンド内では使用できません。 **並べ替えの代わりのインデックス** 副テーブルは、並べ替える代わりにインデックスを付けることができます。大きなテーブルでは並べ替えよりインデックス作成を行った方が、テーブルの結合に必要な時間を短くすることができます。 また、インデックス付きの共通のキーフィールドを使用して 2 つのテーブルを結合する場合は、PRESORT と SECSORT を省略します。
ISOLOCALE *ロケールコード* 省略可能	**メモ** Analytics の Unicode 版にのみ適用されます。 システムロケールは「*言語-国*」の形式で入力します。たとえば、カナダフランス語はコード「fr_ca」を入力します。 次のコードを使用します。 ◦ **言語** - ISO 639 標準言語コード ◦ **国** - ISO 3166 標準国コード 国コードを指定しない場合は、言語のデフォルト国が使用されます。 ISOLOCALE を使用しない場合は、デフォルト システム ロケールが使用されます。

例

業者である可能性がある従業員を検索するための方法として、2 つのテーブルを結合する

次の例は、共通キーフィールドとして住所(Address および Vendor_Street フィールド)を使用し、Empmast および Vendor テーブルを結合します。

JOIN コマンドは、新しいテーブルと、一致する主および副レコードを作成します。これにより、同じ住所の従業員と業者のリストが作成されます。

```
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
```

このバージョンの JOIN コマンドは、主テーブルと副テーブルのすべてのフィールドを結合された出力テーブルに含めます。

```
OPEN Empmast PRIMARY
OPEN Vendor SECONDARY
JOIN PKEY Address FIELDS ALL SKEY Vendor_Street WITH ALL TO "Employee_Vendor_Match"
OPEN PRESORT SECSORT
```

一致する顧客がない売掛金レコードを検出する方法として、2 つのテーブルを結合します。

次の例では、Ar テーブルと Customer テーブルを、顧客番号(CustNo)を共通のキーフィールドとして用いて結合します。

JOIN コマンドは、結合タイプ UNMATCHED を使用して、主テーブルの不一致レコードを含む新しいテーブルを作成しています。その結果、どの Customer レコードとも関連付けられていない Ar レコードの一覧が生成されます。

```
OPEN Ar PRIMARY
OPEN Customer SECONDARY
JOIN PKEY CustNo FIELDS CustNo Due Amount SKEY CustNo UNMATCHED TO
"CustomerNotFound.fil" OPEN PRESORT SECSORT
```

備考

> **メモ**
>
> このコマンドの動作の詳細については、Analytics のヘルプを参照してください。

LIST コマンド

Analytics テーブルの 1 つ以上のフィールドのデータを、列で書式設定されたディスプレイに出力します。

構文

LIST {FIELDS *フィールド名* <AS *表示名*> <...*n*>|FIELDS ALL} <LINE *番号 フィールドリスト*> <TO {SCREEN|*ファイル名*|PRINT}> <UNFORMATTED> <IF *テスト*> <WHILE *テスト*> <FIRST *範囲*|NEXT *範囲*> <HEADER *ヘッダーテキスト*> <FOOTER *フッターテキスト*> <SKIP *行数*> <EOF> <APPEND>

パラメーター

名前	説明
FIELDS *フィールド名* <...*n*> | FIELDS ALL	出力に含めるフィールド： ◦ **FIELDS *フィールド名*** - 指定されたフィールドを使用します ◦ **FIELDS ALL** - テーブルのすべてのフィールドを使用します。
AS *表示名* 省略可能	FIELDS *フィールド名*を使ってデータを一覧表示する場合にのみ使用します。 出力におけるフィールドの表示名（代替列見出し）。表示名をフィールド名、またはソーステーブル内の既存の表示名と同じにしたい場合は、AS を使用しないでください。 *表示名*の値は引用符で囲まれた文字列。列見出しを改行したい場合は、語句の間にセミコロン(;) を入れます。
LINE *番号 フィールドリスト* 省略可能	複数の行が各レコードの出力に使用されます。 ◦ ***番号*** -の値は 2 から 60 までの行番号（両端を含む）のいずれかでなければなりません。 ◦ ***フィールド リスト*** - その行に含めるフィールド
TO SCREEN | *ファイル名* | PRINT 省略可能	コマンドの結果を送信する場所： ◦ **SCREEN** - は Analytics の表示領域に結果を表示します ◦ ***ファイル名*** -は結果の保存先となるファイルです。 *ファイル名*には、適切なファイル拡張子を付けた文字列を引用符で囲んで指定します。 例：`TO "Output.TXT"` デフォルトでは、テーブル ファイルは、Analytics プロジェクトが入っているフォルダーに保存されます。 既存の異なるフォルダーにファイルを保存するには、絶対または相対ファイル パスを指定します。 • `TO "C:\Output.TXT"` • `TO "Results\Output.TXT"` ◦ **印刷** -- 通常使うプリンターに結果を送信します

名前	説明
UNFORMATTED 省略可能	出力は書式設定されていないテキストとして表示されます。出力はEXPORT ASCIIコマンドで作成される出力と同じです。書式なしのデータは、ほかのソフトウェアプログラムで処理する目的でファイルに出力できます。
IF *テスト* 省略可能	各レコードを処理するために真である必要がある条件式。コマンドは、その条件を満たすレコードに対してのみ実行されます。 **メモ** IF パラメーターは、任意の範囲パラメーター(WHILE、FIRST、NEXT) が適用された後に、テーブルに残るレコードに対してのみ評価されます。
WHILE *テスト* 省略可能	各レコードを処理するために真である必要がある条件式。条件がfalseと評価するか、テーブルの最後に達したら、コマンドは実行を中止します。 **メモ** WHILE を FIRST または NEXT とともに使用する場合は、1 つの制限に達するとすぐに、レコードの処理が停止します。
FIRST *範囲* | NEXT *範囲* 省略可能	処理するレコード数: ◦ **FIRST** - 指定されたレコード数に達するまで、最初のレコードから処理を開始します ◦ **NEXT** - 指定されたレコード数に達するまで、現在選択されているレコードから処理を開始します *範囲*は処理するレコード数を指定します。 FIRSTとNEXTを省略すると、すべてのレコードがデフォルトで処理されます。
HEADER *ヘッダーテキスト* 省略可能	レポートの各ページの最上部に挿入されるテキスト。 *ヘッダーテキスト*は引用符で囲んだ文字列として指定する必要があります。この値は、Analytics のシステム変数である HEADER の値よりも優先されます。
FOOTER *フッターテキスト* 省略可能	レポートの各ページの最下部に挿入されるテキスト。 *フッターテキスト*は引用符で囲んだ文字列として指定する必要があります。この値は、Analytics のシステム変数である FOOTER の値よりも優先されます。
SKIP *行数* 省略可能	リストの各レコード間に、指定した数の空白行が挿入されます。たとえば「LIST ALL SKIP 1」と指定すると、1 行おきの行間、つまり各レコード間に空白行が 1 行あるリストが作成されます。
EOF 省略可能	ファイルの終わりに達した後、コマンドをもう一度実行します。 これにより、GROUP コマンド内でテーブルの最後のレコードが処理されることが保証されます。すべてのフィールドが以前のレコードを参照する演算フィールドである場合にのみ使用してください。
APPEND 省略可能	コマンドの出力を既存ファイルに上書きしないで、そのファイルの末尾に追加します。

例

例外をリストアップしてテキスト ファイルに保存する

LIST を使用して、在庫テーブル内で発見された例外をリストアップするレポートを作成するとします。レポートはテキスト ファイルとして保存されます。以上を行うコマンドの例を次に示します。

```
LIST Product_number Description Quantity Unit_cost Value IF Quantity < 0 OR Unit_cost < 0
HEADER "Negative Values" TO "Exceptions.txt"
```

備考

LIST を使用する場面

LIST を使用する場面は、データの印刷、画面へのデータの表示、およびテキスト ファイルへのデータのエクスポートを行うときです。

書式設定と合計

UNFORMATTED を指定する場合を除いて、次の情報が自動的に含まれます。

- ページ番号
- 日付
- 時刻
- ユーザー ID
- 列見出し

また、数値列が自動的に合計されます。

LOCATE コマンド

指定された値または条件に一致する最初のレコードを検索するか、または指定されたレコード番号に移動します。

構文

```
LOCATE {IF テスト <WHILE テスト> <FIRST 範囲|NEXT 範囲>|RECORD 番号}
```

パラメーター

名前	説明
IF *テスト*	検索する値または条件。文字リテラル値は引用符で囲み、日付時刻値は逆引用符で囲む必要があります。
WHILE *テスト* 省略可能	各レコードを処理するために真である必要がある条件式。条件が false と評価するか、テーブルの最後に達したら、コマンドは実行を中止します。 **メモ** WHILE を FIRST または NEXT とともに使用する場合は、1 つの制限に達するとすぐに、レコードの処理が停止します。
FIRST *範囲* | NEXT *範囲* 省略可能	処理するレコード数： ◦ **FIRST** - 指定されたレコード数に達するまで、最初のレコードから処理を開始します ◦ **NEXT** - 指定されたレコード数に達するまで、現在選択されているレコードから処理を開始します *範囲*は処理するレコード数を指定します。 FIRST と NEXT を省略すると、すべてのレコードがデフォルトで処理されます。
RECORD *番号*	位置を示すレコード番号。

例

指定した値と一致する最初のレコードを検索する

次の例では、LOCATE コマンドを使用して、特定の値が最初に現れるテーブルを見つける方法を示します。

```
LOCATE IF Vendor_Name = "United Equipment"
```

```
LOCATE IF Vendor_Name = "Uni"
```

```
LOCATE IF Invoice_Amount > 1000
```

```
LOCATE IF Invoice_Date = `20141231`
```

指定した条件または式と一致する最初のレコードを検索する

次の例では、LOCATE コマンドを使用して、特定の条件または式が最初に現れるテーブルを見つける方法を示します。

```
LOCATE IF Vendor_Name = "United Equipment" AND Invoice_Amount > 1000 AND Invoice_Date > `20140930`
```

```
LOCATE IF Vendor_City = v_city
```

レコード番号によってレコードを検索する

次の例では、LOCATE コマンドを使用して、テーブル内の特定のレコードへ移動する方法を示します。

```
LOCATE RECORD 50
```

備考

> **メモ**
> このコマンドの動作の詳細については、Analytics のヘルプを参照してください。

機能の仕組み

LOCATE コマンドは、指定した値または条件と一致する、テーブル内の最初のレコードに直接移動する場合に使用します。

指定した値または条件が見つかった場合、テーブル内で最初に一致するレコードが選択されます。指定した値または条件が見つからなかった場合、テーブルは最初のレコードに位置付けられます。

また、LOCATE を使用すると、特定のレコード番号に直接移動できます。

LOCATE と FIND および SEEK の比較

FIND コマンドや SEEK コマンドとは違い、LOCATE コマンドでは、検索対象はインデックス付きのテーブルや単一の文字フィールドに制限されません。LOCATE を使用すると、あらゆる型のリテラル、任意のデータ型を

使用した式、データ型が混在した式を検索することができます。

インデックス付きでないテーブルの検索に使用した場合、LOCATE コマンドはテーブル内の各レコードを順次処理する必要があるため、FIND コマンドや SEEK よりも処理速度が著しく低下する可能性があります。必要な処理時間は、テーブルのサイズ、一致するレコードの場所、および WHILE、FIRST、または NEXT を使用して検索範囲を狭めているかどうかによって異なります。

部分一致がサポートされる場合とは

文字検索では、部分一致がサポートされます。フィールドまたは検索対象フィールドに含まれる長い値の一部を検索値に指定できるのです。ただし、検索値は、一致を成すフィールドの先頭に現れる必要があります。

部分一致を有効または無効にする

SET コマンドを使用するか、 [**オプション**]ダイアログボックスを使用して、部分一致を有効または無効にできます。

部分一致を有効にする	部分一致を無効にする
指定： SET EXACT OFF または **選択解除：** [**オプション**]ダイアログボックス(**ツール > オプション > テーブル**) の [**正確な文字比較**] **結果：** フィールドまたは検索対象フィールドに含まれる長い値の一部を検索値に指定できるのです。検索値は、一致を成すフィールドの先頭に現れる必要があります。	**指定：** SET EXACT ON または **選択：** [**オプション**]ダイアログボックス(**ツール > オプション > テーブル**) の [**正確な文字比較**] **結果：** 検索値は、一致を成すフィールドの値と正確に一致しなければなりません。

SET EXACT の詳細については、"SET コマンド" ページ 404を参照してください。

[**正確な文字比較を行う**]オプションの詳細については、テーブルタブ(オプション ダイアログボックス) を参照してください。

LOOP コマンド

指定された条件が真と評価されている間、一連の ACLScript コマンドをレコードで繰り返し実行します。

> **メモ**
> LOOP コマンドは GROUP コマンドで囲む必要があります。

構文

```
LOOP WHILE テスト
 コマンド
 <...n>
END
```

パラメーター

名前	説明
WHILE *テスト*	LOOP コマンド内のコマンドを実行するために、True と評価される必要があるテスト。そのテストが True と評価された場合、それが False と評価されるまで対象コマンドが繰り返し実行されます。
コマンド <...n>	実行対象となる 1 つまたは複数のコマンド。 LOOP コマンド内には複数のコマンドを入力することができます。各コマンドは別々の行で入力する必要があります。
END	LOOP コマンドの終わり。

例

カンマ区切りのフィールドを分割する

請求データを含むテーブルがあり、請求金額の特定の情報を部署ごとに抽出する必要があるとします。1 つの請求書は複数の部署に関連付けることができ、部署コードがカンマ区切りでテーブルに保存されます。

請求金額を部署ごとに抽出するには

1. GROUP コマンドを使用して、テーブルの各レコードを処理します。
2. 各レコードに関連付けられた部署数 (n) を計算します。
3. LOOP コマンドを使用して、レコードに関連付けられた各部署のデータを抽出する操作を n 回繰り返します。

COMMENT GROUP を使用して、各部署コード フィールドのカンマを、レコードに関連付けられた部署数を特定する方法としてカウントするフィールドの各コードの各レコードで "LOOP" し、各コードを独自のレコードに抽出する END GROUP v_department_count = OCCURS(Department_Code,',') v_counter = 0 LOOP WHILE v_counter <= v_department_count v_dept = SPLIT(Department_Code, ',', (v_counter + 1)) EXTRACTFIELDS Invoice_Number, Invoice_Amount, v_dept AS "Department" TO result1 v_counter = v_counter + 1 END END

備考

> **ヒント**
>
> LOOP および GROUP コマンドの詳細なチュートリアルについては、"グループ化とループ処理" ページ 32を参照してください。

LOOP の用途

ループは、処理するデータのセグメントがレコード内に繰り返し含まれている場合に多く用いられます。

機能の仕組み

各 LOOP コマンドでは、テストする WHILE 条件を指定し、END ステートメントで閉じる必要があります。指定したテストが True と評価される間は、LOOP と END の間にあるコマンドが現在のレコードに対して繰り返し実行されます。

テストが False になると、その時点でコマンドは実行されなくなります。

無限ループを回避する

無限ループを回避するために、必ず最終的に False が返されるテストを指定してください。また、SET LOOP コマンドを使用すると、無限ループを防ぐことができます。

MERGE コマンド

同一の構造を持つ2つの並べ替え済みAnalyticsテーブルのレコードを結合して、元のテーブルと同じ並べ替え順になっている新しいAnalyticsテーブルに出力します。

構文

MERGE {ON *キーフィールド*|PKEY *主テーブルのキーフィールド* SKEY *副テーブルのキーフィールド*} <IF *test*> TO *テーブル名* <LOCAL> <OPEN> <WHILE *テスト*> <FIRST *範囲*|NEXT *範囲*> <APPEND> <PRESORT> <ISOLOCALE *ロケールコード*>

パラメーター

名前	説明
ON *キーフィールド* |PKEY *主テーブルのキーフィールド* SKEY *副テーブルのキーフィールド*	**メモ** MERGEでは、文字型のフィールドまたは文字型の演算フィールドのみ、キーフィールドとして使用できます。 ◦ **ON *キーフィールド*** - 主テーブルおよび副テーブルの対応するキーフィールドが同じ名前の場合にマージするキーフィールド 対応するフィールドに異なる名前がある場合、または実際の物理フィールドではなく式である場合は、PKEYおよびSKEYを使用する必要があります。 ◦ **PKEY *主キーフィールド*** - 主テーブルのキーフィールド(複数可)または式 ◦ **SKEY *主キーフィールド*** - 副テーブルのキーフィールド(複数可)または式 **並べ替えの要件** 主テーブルおよび副テーブルのキーフィールドは、どちらも昇順で並べ替えられている必要があります。一方または両方のキーフィールドが並べ替えられていない、あるいは降順で並べ替えられている場合には、MERGEコマンドは失敗します。 PRESORTを使用すると、主テーブルのキーフィールドを並べ替えることができます。副テーブルのキーフィールドが並べ替えられていない場合は、マージを実行する前に、まず別個の並べ替え操作でフィールドの並べ替えを行ってください。 **並べ替えの代わりのインデックス** 主テーブルと副テーブルは、並べ替える代わりにインデックスを付けることができます。大きなテーブルでは並べ替えよりインデックス作成を行った方が、テーブルのマージに必要な時間を短くすることができます。
IF *テスト* 省略可能	各レコードを処理するために真である必要がある条件式。コマンドは、その条件を満たすレコードに対してのみ実行されます。

名前	説明
	メモ IF パラメーターは、任意の範囲パラメーター(WHILE、FIRST、NEXT) が適用された後に、テーブルに残るレコードに対してのみ評価されます。
TO *テーブル名*	コマンドの結果を送信する場所： ◦ ***テーブル名*** -は、結果の保存先となる Analytics テーブルのことです。 *テーブル名*には、ファイル拡張子 .FIL を付けた文字列を引用符で囲んで指定する必要があります。例：TO "Output.FIL" デフォルトでは、テーブル データ ファイル(.fil) は、Analytics プロジェクトが入っているフォルダーに保存されます。 既存の異なるフォルダーにデータ ファイルを保存するには、絶対または相対ファイル パスを指定します。 • TO "C:\Output.FIL" • TO "Results\Output.FIL" **メモ** テーブル名は 64 文字の英数字(.FIL 拡張子を含まない) に制限されています。名前にはアンダースコア文字(_) を使用できますが、他の特殊文字やスペースは使用できません。名前の先頭を数字にすることはできません。
LOCAL 省略可能	Analytics プロジェクトと同じ場所に出力ファイルを保存します。 **メモ** Analytics テーブルである出力ファイルを含むサーバー テーブルに対してコマンドを実行するときにのみ適用されます。
OPEN 省略可能	コマンドを実行した後、コマンドによって作成されたテーブルを開きます。コマンドが出力テーブルを作成する場合にのみ有効です。
WHILE *テスト* 省略可能	各レコードを処理するために真である必要がある条件式。条件が false と評価するか、テーブルの最後に達したら、コマンドは実行を中止します。 **メモ** WHILE を FIRST または NEXT とともに使用する場合は、1 つの制限に達するとすぐに、レコードの処理が停止します。
FIRST *範囲* | NEXT *範囲* 省略可能	処理するレコード数： ◦ **FIRST** - 指定されたレコード数に達するまで、最初のレコードから処理を開始します ◦ **NEXT** - 指定されたレコード数に達するまで、現在選択されているレコードから処理を開始します *範囲*は処理するレコード数を指定します。 FIRST と NEXT を省略すると、すべてのレコードがデフォルトで処理されます。
APPEND 省略可能	コマンドの出力を既存ファイルに上書きしないで、そのファイルの末尾に追加します。

名前	説明
	メモ コマンドの出力と既存のファイルの構造が同一であるようにする必要があります。 • 同じフィールド • 同じフィールド順序 • 一致するフィールドが同じ長さ • 一致するフィールドが同じデータ型 出力は、既存ファイルとの間でファイル構造が違っている場合でも、Analyticsによって既存ファイルに追加されます。出力と既存のファイルの構造が一致しない場合は、データが混在、不足、不正確になります。
PRESORT 省略可能	コマンドを実行する前に主キーフィールドで主テーブルを並べ替えます。 **メモ** GROUPコマンドの内部ではPRESORTを使用することができません。 以下の場合は、PRESORTを省略します。 ◦ 主テーブルのキーフィールドが既に並べ替えられている場合 ◦ インデックス付きの共通のキーフィールドを使用して2つのテーブルをマージする場合
ISOLOCALE *ロケールコード* 省略可能	**メモ** Analyticsの Unicode 版にのみ適用されます。 システムロケールは「*言語-国*」の形式で入力します。たとえば、カナダフランス語はコード「fr_ca」を入力します。 次のコードを使用します。 ◦ **言語** - ISO 639 標準言語コード ◦ **国** - ISO 3166 標準国コード 国コードを指定しない場合は、言語のデフォルト国が使用されます。 ISOLOCALEを使用しない場合は、デフォルト システム ロケールが使用されます。

例

キーフィールド名が同一のテーブルをマージする

次の例では、キーフィールド名が同一である2つのテーブルをマージします。

```
OPEN Employees_Location_1 PRIMARY
OPEN Employees_Location_2 SECONDARY
MERGE ON Last_Name TO "AllEmployees" PRESORT
```

キーフィールド名が異なるテーブルをマージする

次の例では、キーフィールド名が異なる2つのテーブルをマージします。

```
OPEN Employees_Location_1 PRIMARY
OPEN Employees_Location_2 SECONDARY
MERGE PKEY Last_Name SKEY Surname TO "AllEmployees" PRESORT
```

備考

> **メモ**
> このコマンドの動作の詳細については、Analytics のヘルプを参照してください。

マージの代替手段

マージを正常に実行するには、注意が必要です。追加してから並べ替える、あるいは抽出または追加してから並べ替えるのでも、同じ結果を得ることができます。

詳細については、"APPEND コマンド" ページ 70と"EXTRACT コマンド" ページ 193を参照してください。

2 つのソーステーブルが既に並べ替えられている場合は、マージはより効率的で、より迅速に実行できます。

NOTES コマンド

Analytics テーブル内の個々のレコードに関連付けられている注釈(ノート)を作成、編集、または削除します。

構文

```
NOTES <IF テスト> <TEXT ノート テキスト> <APPEND> <CLEAR>
```

パラメーター

名前	説明
IF テスト 省略可能	各レコードを処理するために真である必要がある条件式。コマンドは、その条件を満たすレコードに対してのみ実行されます。 **メモ** IF パラメーターは、任意の範囲パラメーター(WHILE、FIRST、NEXT) が適用された後に、テーブルに残るレコードに対してのみ評価されます。 ◦ IF テストを指定しなければ、ノートテキストはテーブルの各レコードに追加されます。 ◦ IF テストとCLEAR を指定すると、条件を満たすレコードのノートが削除されます。
TEXT *ノートテキスト* 省略可能	テキストはノートとして追加されます。*ノートテキスト*は、引用符で囲まれた文字列、または文字式である必要があります。
APPEND 省略可能	ノート テキストは既存ノートの末尾に追加されます。省略した場合は、既存のノートがすべて上書きされます。
クリア 省略可能	ノートは削除されます。テーブル内のすべてのレコード ノートを削除しても、自動生成された **RecordNote** フィールドはテーブル レイアウトから削除されません。

例

複数のレコードに同じノートを追加する

次の例では、指定したレコードの既存のノートがすべて上書きされます。

```
NOTES IF MATCH(RECNO(),1,3,5,7) TEXT "ノート本文"
```

複数のレコードに同じノートを追加または付加する

次の例では、指定したレコードの既存のノートに、新しいノート テキストが付加されます。

```
NOTES IF MATCH(RECNO(),1,3,5,7) TEXT "ノート本文" APPEND
```

複数のレコードからノートを削除する

次の例では、テーブルのすべてのレコード メモが削除されます。

```
NOTES CLEAR
```

次の例では、指定したレコードのノートが削除されます。

```
NOTES IF MATCH(RECNO(),1,3,5,7) CLEAR
```

次の例では、レコード 1 ～ 100 のノートが削除されます。

```
NOTES IF RECNO() <= 100 CLEAR
```

備考

RecordNote フィールドを削除する

テーブル レイアウトから **RecordNote** フィールドを削除することでテーブル内のすべてのノートを削除するには、DELETE NOTES コマンドをオプションの指定なしで使用します。

NOTIFY コマンド

電子メール通知メッセージを送信します。

構文

NOTIFY USER *ユーザー名* <PASSWORD *パスワード*> MAILBOX *パス名* ADDRESS *受信者* <CC *Cc 受信者*> <BCC *Bcc 受信者*> <SUBJECT *件名*> MESSAGE *メッセージ* <ATTACHMENT *パス名*>

パラメーター

名前	説明
USER *ユーザー名*	送信者の電子メール アドレス。
PASSWORD *パスワード* 省略可能	メール サーバー用のパスワード。
MAILBOX *パス名*	電子メール メッセージを送信するために使用する SMTP サーバー名。例： MAILBOX "mailserver.acl.com"
ADDRESS *受信者*	1 人または複数人の受信者の電子メール アドレス。複数人の受信者の電子メール アドレスの場合は、カンマで区切ります。 最大 1020 文字を入力します。
CC *Cc_受信者* 省略可能	1 人または複数人のカーボン コピー受信者の電子メール アドレス。複数人の受信者の電子メール アドレスの場合は、カンマで区切ります。 最大 1000 文字を入力します。
BCC *Bcc_受信者* 省略可能	1 人または複数人のブラインド カーボン コピー受信者の電子メール アドレス。複数人の受信者の電子メール アドレスの場合は、カンマで区切ります。
SUBJECT *件名* 省略可能	電子メール メッセージの件名。
MESSAGE *メッセージ*	電子メール メッセージの本文テキスト。メッセージはプレーン テキストのため、HTML はサポートされていません。 メッセージに改行を挿入したい場合は、次の 2 文字を使用します。^^
ATTACHMENT *パス名*	1 つ以上の添付ファイルのパスとファイル名。文字列を引用符で囲んで指定する必要がありま

名前	説明
省略可能	す。 複数の添付ファイルを指定するには、パス名にカンマ区切りのファイルのリストを入力します。 ATTACHMENT "result1,result2"

例

エラー報告の電子メールを送信する

スクリプトを実行し、失敗した場合には電子メール通知を送信したいとします。NOTIFY を使って電子メールメッセージを定義し、以下の2つの添付ファイルを添付します。

- ログファイル
- エラーが記録された .fil ファイル

```
NOTIFY USER "support@company.com" MAILBOX "mail.company.com" ADDRESS "script_admin@acl.com" SUBJECT "エラーレポート" MESSAGE "スクリプトの処理に失敗しました。詳細を添付しました。" ATTACHMENT "Errors.fil,ACL_Demo.log"
```

備考

受信者と添付ファイル

NOTIFY コマンドを使用すると、1人以上の受信者に電子メール通知メッセージを送信することができます。メッセージには、データファイルや Analytics プロジェクトを添付して含めることができます。

NOTIFY コマンドは、スクリプトが予期せず失敗したときに、適切な担当者へ通知するために使用できます。

プロトコルとポート

コマンドは、Microsoft Exchange やその他の多くのメールサーバーで使用される SMTP(簡易メール転送プロトコル) をサポートする任意のメールサーバーで使用することができます。またこのコマンドは、ローカルにメールを送信する(Microsoft およびその他提供の) 古い電子メールアプリケーションで使用することもできます。

NOTIFY はポート 25 を使用するため、このポートが開いている必要があります。開いていない場合、コマンドは失敗します。

エラー処理

Analytics をメールサーバーに接続できない場合は、接続の試行が10秒おきに5回繰り返されます。接続の試行がすべて失敗した場合、NOTIFY コマンドはキャンセルされ、メッセージがログに記録されますが、スクリプト

は処理を続行します。

このデフォルトの動作は、SETコマンドを使用して変更できます。接続の試行回数と試行の間隔に別の値を指定したり、追加の接続試行をオフにしたりすることができます。また、NOTIFYコマンドがキャンセルされたらスクリプトの処理が停止されるように指定することもできます。詳細については、"SETコマンド" ページ 404を参照してください。

無効な電子メール受信者は、NOTIFYコマンドの失敗と見なされないので、関連する設定に関係なく、これによりスクリプトが停止されることはありません。

OPEN コマンド

Analytics テーブルおよび関連するデータ ファイルを開きます。

構文

OPEN {*テーブル名*|*データ ファイル*<FORMAT *レイアウト名*>}<BUFFERLENGTH *長さ*> <CRLF> <DBASE> <INDEX *インデックス ファイル*> <PRIMARY|SECONDARY> <SKIP *バイト*> <RELATION *キー フィールド*>

パラメーター

名前	説明
テーブル名	開く Analytics テーブルの名前。
データファイル	FORMAT *レイアウト名* で指定されたテーブルに関連付けるデータ ファイル。 拡張子が指定されない場合、Analytics はファイル拡張子を .fil と仮定します。拡張子のないファイルを開くには、ファイル名の最後にピリオド (.) を入れてください。
FORMAT *レイアウト名* 省略可能	開くデータ ファイルに適用する Analytics テーブル レイアウト。
BUFFERLENGTH *長さ* 省略可能	テーブルに割り当てる入力バッファー領域の長さ(バイト数) 。デフォルト値は 33,000 バイトです。 バッファー領域を増やすと処理速度は向上しますが、Analytics コマンドを格納するために使用可能な RAM が使われます。 バッファー長を超す IBM 可変長ブロックを読み取ると、Analytics はエラー メッセージを表示して処理を停止します。デフォルト値は、[**オプション**]ダイアログ ボックスの [**テーブル**]タブにある [**バッファー サイズ**]フィールドで設定します。 BUFFERLENGTH *n* のデフォルト値は、ほとんどの状況を処理できるような設定になっているため、パラメーターの変更が必要になることはほとんどありません。
CRLF 省略可能	可変長 ASCII ファイルが読み取られることを指定します。変化するレコード長は Analytics によって自動的に調整されます。 デフォルトで、ファイルは固定長ファイルであると仮定されます。
DBASE 省略可能	データ ソースが dBASE ファイルであることを指定します。Analytics によって dBASE ファイルの種類が認識され、そのファイル記述を基に自動的にテーブルが作成されます。.dbf ファイル拡張子の付いた dBASE ファイルでは省略することができます。
INDEX *インデックスファイル*	テーブルを開くとき、そのテーブルに適用されるインデックス ファイル。

名前	説明
省略可能	インデックス ファイル名に拡張子を指定しないと、.inx と見なされます。INDEX は、主テーブルまたは副テーブルのいずれかに指定することができます。
PRIMARY | SECONDARY 省略可能	テーブルを主テーブルまたは副テーブルのいずれとして開くかを指定します。省略した場合、テーブルは主テーブルとして開かれます。
SKIP *バイト* 省略可能	テーブルの物理的な先頭部分でバイパスするバイトの数。 テーブルのヘッダー レコード、あるいはテーブル先頭部分でテーブルのレイアウトに従っていない部分を無視する場合にSKIP を使用できます。省略した場合、テーブルは最初のバイトから読み取られます。 **メモ** 非 Unicode 版 Analytics: 1 バイト = 1 文字 Unicode 版 Analytics、拡張 ASCII (ANSI) データ: 1 バイト = 1 文字 Unicode 版 Analytics、Unicode データ: 2 バイト = 1 文字 Unicode データでは、偶数バイトのみを指定します。奇数バイトを指定すると、文字が正しく表示されない可能性があります。
RELATION *キー フィールド* 省略可能	テーブルを一時的に関連付けられたテーブルとして開くことを指定します。Analytics は、テーブルを閉じるときにこの関係を保存しません。 RELATION を使用する際は、INDEX パラメーターも指定する必要があります。*キーフィールド* は、ACL が 2 テーブル間の関係を作成するためのキー フィールドまたは式です。

例

テーブル レイアウトを指定してテーブルを開く

次の例では、**March_2012** テーブル レイアウトを使って **April_2012** テーブルを開きます。

```
OPEN April_2012 FORMAT March_2012
```

dBASE ファイルを開く

次の例では、既存のテーブルが存在しない `Inventory.dbf` という名前の dBASE ファイルを開きます。

```
OPEN Inventory
```

テーブルを開き、既存のインデックスを適用する

主または副いずれかのテーブルを開いて、そのテーブルに既存のインデックスを適用するには、次の構文を使

用します。

```
OPEN Accounts_receivable INDEX Customer_number_AR
```

```
OPEN Customer SECONDARY INDEX Customer_number
```

テーブルを開き、別のテーブルとの間の一時的な関連付けを作成する

Customers テーブル(主テーブル) を開き、このテーブルと**Accounts_receivable** テーブル(副テーブル) の間に一時的な関連付けを作成する必要があるとします。

Customer_index というインデックスと、主テーブル内の **Last_name** というキーフィールドを使用する場合、コマンドは次のようになります。

```
OPEN Accounts_receivable INDEX Customer_index RELATION Last_name
```

OUTLIERS コマンド

数値型フィールドにおける統計上の異常値を検出します。異常値の検出対象は、数値型フィールド全体であるか、あるいは 1 つまたは複数の文字型、数値型、または日付時刻型キーフィールドの値に基づく複数のグループです。

構文

OUTLIERS {AVERAGE|MEDIAN} {PKEY *キーフィールド* <...*n*>|NOKEY} ON *数値型フィールド* <OTHER *フィールド* <...*n*>> NUMSTDEV *標準偏差の数* <IF *テスト*> <TO {SCREEN|*テーブル名*}> <PRESORT> <WHILE *テスト*> <FIRST *範囲*|NEXT *範囲*> <OPEN>

> **メモ**
>
> サーバーのテーブルに対してローカルで OUTLIERS コマンドを実行することはできません。
>
> OUTLIERS コマンドはその全体を指定する必要があります。簡略化することはできません。

パラメーター

名前	説明
AVERAGE | MEDIAN	*数値型フィールド* (異常値フィールド) の値の中心点を計算する方法。 ◦ **AVERAGE** - を指定すると、値の平均 (平均値) が計算されます。 ◦ **MEDIAN** - を指定すると、値の中央値が計算されます。 中心点は、以下のいずれかについて計算されます。 ◦ 数値型フィールド全体 ◦ 各キーフィールド グループの数値 これらの中心点は、後で数値型フィールドまたは各グループの標準偏差を計算する際に使用されます。 **メモ** MEDIAN を指定する場合は、*数値型フィールド*を基準にして並べ替えを行っておく必要があります。*数値型フィールド*を基準にした並べ替えがまだ行われていない場合は、PRESORT を使用します。 **ヒント** 異常値がないかどうかを調べるデータに大きな偏りがある場合は、MEDIAN を指定した方が、データの大勢をより正しく表す結果を生成することができます。
PKEY *キーフィールド* <...*n*> | NOKEY	PKEY を指定した場合は、グループレベルの異常値が検出されます。NOKEY を指定した場合は、フィールド レベルの異常値が検出されます。 ◦ **PKEY *キーフィールド*** - テーブルのデータをグループ化するのに使用される 1 つまたは複数のフィールド

名前	説明
	キー フィールドとしては、文字型、数値型、または日付時刻型のフィールドを設定することができます。複数のフィールドはスペースで区切る必要があります。また、異なるデータ型を指定できます。 複数のフィールドを指定すると、入れ子のグループが作成されます。入れ子でのフィールド間の順序は、フィールドを指定した順になります。 キー フィールド グループごとに、*数値型フィールド*におけるそのグループの数値に関する標準偏差が計算されます。このグループの標準偏差は、グループの異常値を検出するための基準値として使用されます。 **メモ** 1 つまたは複数のキー フィールドを基準にして並べ替えを行っておく必要があります。1 つまたは複数のフィールドを基準にした並べ替えがまだ行われていない場合は、PRESORT を使用します。 ◦ **NOKEY** - は、テーブルのデータをグループ化しません。 *数値型フィールド*全体に対する標準偏差が計算されます。このフィールドの標準偏差は、フィールドの異常値を検出するための基準値として使用されます。
ON *数値フィールド*	異常値がないかどうかを調べる数値型フィールド。一度に 1 つのフィールドしか調べることができません。 異常値とは、フィールドまたは主キー グループの標準偏差またはそのような標準偏差の指定倍数によって設定される上限と下限の範囲に含まれない値のことです。
OTHER *フィールド* <...*n*> 省略可能	出力に含める 1 つ以上の追加フィールド。 **メモ** キー フィールドと異常値フィールドは、自動的に出力テーブルに追加されるため、OTHER に指定する必要はありません。
NUMSTDEV *標準偏差の数*	*数値型フィールド*において、平均または中央値から異常値の上限および下限までに含まれる標準偏差の数。任意の正の整数または 10 進数(0.5、1、1.5、2、...) を指定できます。 異常値の境界を作成するための式: 平均/中央値 ± (*標準偏差数* * 標準偏差) **メモ** 標準偏差はデータセットの分布の測定です。つまり、値の拡散方法を測定します。異常値計算は母集団標準偏差を使用します。 **異常値の境界の例** NUMSTDEV 2 *数値型フィールド*全体または各キー フィールド グループに対し、以下が設定されます。 • 平均または中央値より標準偏差の 2 倍だけ大きい、異常値の上限 平均/中央値 + (2 * SD) • 平均または中央値より標準偏差の 2 倍だけ小さい、異常値の下限 平均/中央値 - (2 * SD)

<table>
<tr><th>名前</th><th>説明</th></tr>
<tr><td></td><td>上限より大きいか下限より小さい任意の値が、異常値として出力結果に追加されます。

メモ
同じデータセットに対し、標準偏差の数の値を大きくすると、返される異常値の数が減る可能性があります。</td></tr>
<tr><td>IF テスト
省略可能</td><td>各レコードを処理するために真である必要がある条件式。コマンドは、その条件を満たすレコードに対してのみ実行されます。

メモ
IF パラメーターは、任意の範囲パラメーター(WHILE、FIRST、NEXT) が適用された後に、テーブルに残るレコードに対してのみ評価されます。</td></tr>
<tr><td>TO SCREEN | テーブル名
省略可能</td><td>コマンドの結果を送信する場所：
◦ SCREEN - は Analytics の表示領域に結果を表示します
◦ テーブル名 -は、結果の保存先となる Analytics テーブルのことです。

テーブル名には、ファイル拡張子 .FIL を付けた文字列を引用符で囲んで指定する必要があります。例：TO "Output.FIL"

デフォルトでは、テーブル データ ファイル(.fil) は、Analytics プロジェクトが入っているフォルダーに保存されます。

既存の異なるフォルダーにデータ ファイルを保存するには、絶対または相対ファイル パスを指定します。
• TO "C:\Output.FIL"
• TO "Results\Output.FIL"

メモ
テーブル名は 64 文字の英数字(.FIL 拡張子を含まない) に制限されています。名前にはアンダースコア文字(_) を使用できますが、他の特殊文字やスペースは使用できません。名前の先頭を数字にすることはできません。</td></tr>
<tr><td>PRESORT
省略可能</td><td>当該のコマンドを実行する前に並べ替え操作を実行します。
<table>
<tr><th>PRESORT と以下を指定した場合：</th><th>並べ替えの基準:</th></tr>
<tr><td>PKEY, AVERAGE</td><td>◦ キーフィールドまたはフィールド
◦ キーフィールドまたはフィールド、次に数値フィールド(数値フィールドが計算される場合)

メモ
計算された数値フィールドの並べ替えは内部的な Analytics の技術要件です。</td></tr>
<tr><td>PKEY, MEDIAN</td><td>キーフィールドまたはフィールド、次に数値フィールド</td></tr>
<tr><td>NOKEY, AVERAGE</td><td>並べ替えなし</td></tr>
</table>
</td></tr>
</table>

名前	説明
	PRESORT と以下を指定した場合： / 並べ替えの基準： NOKEY, MEDIAN / *数値フィールド* **ヒント** 入力テーブル内の 1 つまたは複数の該当フィールドを基準にした並べ替えが既に行われている場合は、PRESORT を指定しないことで処理時間を短縮できます。 **メモ** GROUP コマンドの内部では PRESORT を使用することができません。
WHILE *テスト* 省略可能	各レコードを処理するために真である必要がある条件式。条件が false と評価するか、テーブルの最後に達したら、コマンドは実行を中止します。 **メモ** WHILE を FIRST または NEXT とともに使用する場合は、1 つの制限に達するとすぐに、レコードの処理が停止します。
FIRST *範囲* | NEXT *範囲* 省略可能	処理するレコード数： ◦ **FIRST** - 指定されたレコード数に達するまで、最初のレコードから処理を開始します ◦ **NEXT** - 指定されたレコード数に達するまで、現在選択されているレコードから処理を開始します *範囲*は処理するレコード数を指定します。 FIRST と NEXT を省略すると、すべてのレコードがデフォルトで処理されます。
OPEN 省略可能	コマンドを実行した後、コマンドによって作成されたテーブルを開きます。コマンドが出力テーブルを作成する場合にのみ有効です。

例

異常な取引金額の検出

`Sample Project.acl` 内の Ar テーブル全体から、異常な取引金額を検出したいとします。

異常値の上下限を「**Amount**(金額) フィールドから標準偏差の±3 倍」に設定します。次のテストにより、772 個のレコードから成るテーブルから、16 個の異常値が返されます。

```
OPEN Ar
OUTLIERS AVERAGE NOKEY ON Amount NUMSTDEV 3 PRESORT TO "Outliers_AR.fil" OPEN
```

「標準偏差の±3.5 倍」にしてテストを繰り返します。異常値の上下限が **Amount** フィールドの値群の中心点から遠くなったため、6 個の異常値がこのテストによって返されます。

```
OPEN Ar
OUTLIERS AVERAGE NOKEY ON Amount NUMSTDEV 3.5 PRESORT TO "Outliers_AR.fil"
OPEN
```

顧客別の異常な取引金額の検出

`Sample Project.acl` 内の Ar テーブルから、顧客別に異常な取引金額を検出したいとします。

異常値の上下限を「各顧客の取引グループから標準偏差の±3 倍」に設定します。

```
OPEN Ar
OUTLIERS AVERAGE PKEY No ON Amount NUMSTDEV 3 PRESORT TO "Outliers_Customer_
AR.fil" OPEN
```

このテストにより、7 個の異常値が返されます。各顧客の取引グループについて、標準偏差と平均が報告されます。

	顧客番号(No.)	取引金額	STDEV	AVERAGE	グループ番号
1	065003	4,954.64	1015.58	833.83	1
2	**262001**	3,567.34	772.44	438.81	2
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

顧客 262001 に関する異常値の特定方法

Ar テーブルには顧客 262001 の取引が 101 個あり、そのうち 2 つが異常値として報告されたとします。これらは、その顧客の異常値境界を超えるためです。

異常値	下限	上限	異常値
(2,044.82)	(1,878.51)	2,756.13	3,567.34

顧客 262001 に関する異常値境界の計算方法

異常値の上下限は、顧客 262001 の全取引の平均に、取引の標準偏差の指定倍数をプラスマイナスしたものです。

顧客 262001 の全取引の平均	438.81
標準偏差の指定倍数	3
取引の標準偏差	772.44
	438.81 ± (3 * 772.44) = 438.81 ± 2,317.32 = **(1,878.51) (下限)** = **2,756.13 (上限)**

MEDIAN を使用した、顧客別の異常な取引金額の検出

AVERAGE の代わりにMEDIAN を使用した場合でも、上記の例で行ったのと同じ異常値テストを行うことができます。

```
OPEN Ar
OUTLIERS MEDIAN PKEY No ON Amount NUMSTDEV 3 PRESORT TO "Outliers_Customer_AR_
Median.fil" OPEN
```

上のテストでは、直前のテストで返された7 個の異常値の代わりに、10 個の異常値が返されます。MEDIAN とAVERAGE では、データの性質に応じて多少異なる結果が返されます。

	顧客番号(No.)	取引金額	STDEV	MEDIAN	グループ番号
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

各顧客に関する異常値境界の計算方法

異常値の上下限は、各顧客の取引の中央値に、取引の標準偏差の指定倍数をプラスマイナスしたものです。

顧客 262001 の例：450.67 ± (3 * 772.44)

備考

> **メモ**
>
> このコマンドの動作の詳細については、Analytics のヘルプを参照してください。

異常値境界フィールドを結果テーブルに追加する

Analytics は、自動的に、**STDEV** および **AVERAGE** または **MEDIAN** 演算フィールドを異常値結果テーブルに追加します。結果テーブルで異常値を特定するために使用される異常値境界を示す 2 つの演算フィールドも追加すると役立つ場合があります。

1. 異常値結果テーブルを開きます。
2. この式を Analytics コマンドラインに貼り付けて、必要に応じて編集し、Enter を押します。

   ```
   DEFINE FIELD 下限 COMPUTED AVERAGE - (標準偏差数 * STDEV)
   ```

   - *標準偏差数*については、使用した実際の標準偏差乗数を代替します。
   - 平均ではなく中央点として中央値を使用した場合は、AVERAGE に MEDIAN を使用します。
3. この式を Analytics コマンドラインに貼り付けて、必要に応じて編集し、Enter を押します。

   ```
   DEFINE FIELD 上限 COMPUTED AVERAGE + (標準偏差数 * STDEV)
   ```

   - *標準偏差数*については、使用した実際の標準偏差乗数を代替します。
   - 平均ではなく中央点として中央値を使用した場合は、AVERAGE に MEDIAN を使用します。
4. ビュー内で右クリックしてから [**列の追加**]を選択します。
5. [**使用可能なフィールド**]リストから、**下限**および**上限**をダブルクリックし、[**選択したフィールド**]リストに追加します。
6. [**OK**]をクリックします。
7. 省略可能。列ヘッダーをドラッグして、追加したフィールドを再配置します。

PASSWORD コマンド

スクリプトの実行中にパスワードの入力をユーザーに求めるために、パスワード値を含まないパスワード定義を作成します。

構文

PASSWORD *番号* <*プロンプト*>

パラメーター

名前	説明
数字	パスワード定義を一意に識別する 1 から 10 までの値。
プロンプト 省略可能	パスワードの入力を求めるために使用されるダイアログ ボックスに表示される文字型の有効な式。この場合、リテラル文字列は引用符で囲みます。 *プロンプト*を省略した場合は、メッセージのないデフォルト ダイアログ ボックスが表示されます。

例

パスワード情報の入力を求める

PASSWORD コマンドを使用して、必要な 3 つのパスワードを入力するようスクリプトでユーザーに求めるとします。ユーザーが必要なパスワードを入力したら、スクリプトは中断することなく、残りの処理を完了することができます。

```
PASSWORD 1 "売掛金データベースのパスワードを入力してください"
PASSWORD 2 "買掛金データベースのパスワードを入力してください"
PASSWORD 3 "顧客データベースのパスワードを入力してください"
```

Analytics テーブルを更新するときパスワードを指定する

パスワードで保護されたデータ ファイルを更新するために、REFRESH コマンドと PASSWORD コマンドを組み合わせた例を次に示します。

```
PASSWORD 1 "パスワード:"
REFRESH Abc PASSWORD 1
```

サーバー テーブルを定義するためにパスワードを指定する

AX Connector 経由でサーバー テーブルを定義するとき、データベース プロファイルと関連するサーバー プロファイルのパスワードを必要とする場合には、DEFINE TABLE DB コマンドと共にPASSWORD コマンドを使用できます。

```
DEFINE TABLE DB SOURCE Inventory_DBProfile PASSWORD 9 PASSWORD 3
```

備考

PASSWORD の用途

PASSWORD コマンドを使用すると、スクリプトがパスワード保護されたデータにアクセスするか、パスワード保護されたデータをインポートまたは構成する前に、ユーザーのパスワード入力を要求します。

スクリプトでは最大 10 個の異なるパスワード定義を作成できます。

PASSWORD は次の場合に有用です。

- スクリプトに実際のパスワードを入力する(SET PASSWORD コマンドで要求される) ことを避けたい場合
- 各ユーザーが異なるパスワードを入力する必要がある場合

パスワードの保存方法

ユーザーが入力したパスワードは一時的に安全にメモリに格納されます。

ユーザーがプロンプト ダイアログボックスにパスワードを入力するときには、アスタリスク(*) を使用して、文字がマスクされます。 パスワードは、スクリプトにもログにも表示されません。

サーバーに基づくアナリティクスのパスワードを保存する

PASSWORD コマンドは、ロボット、AX Server、またはレガシー サーバー スクリプトのアナリティクス実行ではサポートされません。

ユーザーがロボットまたは AX Server でアナリティクスをスケジュールするときには、パスワードを確認するためにPASSWORD タグを使用できます。

SET PASSWORD コマンドを使用すると、レガシー サーバー スクリプトでパスワードを指定できます。

PAUSE コマンド

スクリプトを一時停止し、ユーザーのダイアログ ボックスに情報を表示します。

構文

PAUSE *メッセージ* <IF *テスト*>

パラメーター

名前	説明
メッセージ	ダイアログ ボックスに表示するメッセージ。199 文字を上限とします。 *メッセージ*は二重引用符で囲む必要があります。メッセージ内に二重引用符がある場合は、二重引用符を一重引用符で囲んでください。
IF *テスト* 省略可能	各レコードを処理するために真である必要がある条件式。コマンドは、その条件を満たすレコードに対してのみ実行されます。 **メモ** IF パラメーターは、任意の範囲パラメーター(WHILE、FIRST、NEXT) が適用された後に、テーブルに残るレコードに対してのみ評価されます。

例

エラー メッセージを表示する

特定の要件を満たすユーザー入力が必要であるとします。入力が要件を満たさないことを検出した場合には、PAUSE コマンドを使ってエラー メッセージをダイアログ ボックスに表示します。次のようにします。

```
PAUSE "この製品クラスの値は 2 桁にしてください。"
```

備考

PAUSE の用途

PAUSE の用途は、スクリプトの実行中に読み取り専用のメッセージを画面に表示することです。エラー メッセージや分析操作の結果などの情報を表示できます。

機能の仕組み

メッセージ ダイアログ ボックスが表示されている間、スクリプトの実行は停止され、ユーザーが [OK] をクリックしてメッセージ ダイアログ ボックスを閉じた後にのみ再開されます。このため、無人で実行する必要があるスクリプトやアナリティクスで PAUSE を使用することはできません。

制限

PAUSE には次の制限があります。

- GROUP コマンド内に含めることができない
- ロボットまたは AX Server で実行されるアナリティクスで使用できない

PREDICT コマンド

予測モデルをラベルがないデータセットに適用し、個別のレコードに関連付けられたクラスまたは数値を予測します。

構文

```
PREDICT MODEL モデル名 TO テーブル名 <IF テスト> <WHILE テスト> <FIRST 範囲|NEXT 範囲>
```

パラメーター

名前	説明
MODEL *モデル名*	クラスまたは値を予測するために使用するモデルファイルの名前。以前に TRAIN コマンドで生成されたモデルファイルを使用します。 ファイル拡張子 `*.model` を指定する必要があります。例: `MODEL "Loan_default_prediction.model"` **メモ** モデルファイルは、ラベルのないデータセットと同じフィールドまたはほぼ同じフィールドのデータセットで学習されている必要があります。
TO *テーブル名*	予測プロセスで生成された Analytics テーブル この出力テーブルには、学習プロセス中に指定したキーフィールドと、予測プロセスで生成された 1 つまたは 2 つのフィールドが含まれます。 ◦ **予測** - ラベルがないデータセットの各レコードに関連付けられた予測されたクラスまたは数値。 ◦ **確率** -(分類のみ) 予測されたクラスが正しい確率 *テーブル名*、.FIL ファイル拡張子を持つ引用符で囲まれた文字列として指定します。例: `TO "Loan_applicants_default_predicted.FIL"` デフォルトでは、テーブルデータファイル(.FIL) は、Analytics プロジェクトが入っているフォルダーに保存されます。 既存の異なるフォルダーにデータファイルを保存するには、絶対または相対ファイルパスを指定します。 ◦ `TO "C:\Loan_applicants_default_predicted.FIL"` ◦ `TO "ML Predict output\Loan_applicants_predicted_default.FIL"` **メモ** テーブル名は 64 文字の英数字(.FIL 拡張子を含まない) に制限されています。名前にはアンダースコア文字(_) を使用できますが、他の特殊文字やスペースは使用できません。名前の先頭を数字にすることはできません。

名前	説明
IF *テスト* 省略可能	各レコードを処理するために真である必要がある条件式。コマンドは、その条件を満たすレコードに対してのみ実行されます。 **メモ** IF パラメーターは、任意の範囲パラメーター(WHILE、FIRST、NEXT) が適用された後に、テーブルに残るレコードに対してのみ評価されます。
WHILE *テスト* 省略可能	各レコードを処理するために真である必要がある条件式。条件が false と評価するか、テーブルの最後に達したら、コマンドは実行を中止します。 **メモ** WHILE を FIRST または NEXT とともに使用する場合は、1 つの制限に達するとすぐに、レコードの処理が停止します。
FIRST *範囲* | NEXT *範囲* 省略可能	処理するレコード数: ◦ **FIRST** - 指定されたレコード数に達するまで、最初のレコードから処理を開始します ◦ **NEXT** - 指定されたレコード数に達するまで、現在選択されているレコードから処理を開始します *範囲*は処理するレコード数を指定します。 FIRST とNEXT を省略すると、すべてのレコードがデフォルトで処理されます。

例

分類モデルを使用して、予測を行う

分類モデルを PREDICT コマンドを入力し、融資が実行された場合に、現在の融資申請者が債務不履行になる予測を行います。

以前に、融資のデフォルト情報を含む履歴融資データのセットに対して TRAIN コマンドを使用して、分類モデルを生成しました。

```
OPEN "Loan_applicants_current"
PREDICT MODEL "Loan_default_prediction.model" TO "Loan_applicants_default_predicted.FIL"
```

回帰モデルを使用して予測を行う

回帰モデルを PREDICT コマンドに入力することで、将来の住宅販売価格を予測します。

あなたは以前に、販売価格を含む最近の住宅販売データのセットに対して TRAIN コマンドを使用して、回帰モデルを作成しました。

```
OPEN "House_price_evaluation"
PREDICT MODEL "House_price_prediction.model" TO "House_prices_predicted.FIL"
```

備考

> **メモ**
> このコマンドの動作の詳細については、Analytics のヘルプを参照してください。

PRINT コマンド

テキスト ファイル、Analytics ログ ファイル、その他 外 部 ファイルとしてエクスポートされた Analytics プロジェクト 項目 (スクリプト (.aclscript) 、テーブル(.layout) 、ワークスペース(.wsp) など) を印 刷します。コマンドで生 成 されたグラフも印 刷 できます。

構 文

```
PRINT {ファイル名|GRAPH}
```

パラメーター

名前	説明
ファイル名|GRAPH	印 刷する項 目 : ◦ ***ファイル名*** -印 刷 するファイルの相 対または絶 対 パスとファイル名 例 : "C:\ACL Data\Sample Data Files\ACL_Demo.log" または "Sample Data Files\ACL_Demo.log"。 パスまたはファイル名 にスペースが含 まれている場 合 は、*ファイル名*を引 用 符 で囲む必 要 があります。 ◦ **GRAPH** - コマンド の結 果 として以 前 に出 力 されたグラフを印 刷します

例

ログ ファイルを印 刷する

ACL_Demo.acl プロジェクト 用 のログ ファイルを印 刷 するには、次 のコマンドを指 定します。

```
PRINT "C:\ACL Data\Sample Data Files\ACL_Demo.log"
```

グラフを印 刷する

BENFORD コマンド から生 成 されたグラフを印 刷 するには、次 のコマンドを指 定します。

```
OPEN Metaphor_APTrans_2002
BENFORD ON Invoice_Amount LEADING 1 TO GRAPH
PRINT GRAPH
```

備考

プリンターを選択する

使用されるプリンターは、Microsoft Windows で設定された通常使うプリンターです。プリンターを変更するには Windows の通常使うプリンターを変更する必要があります。

関連コマンド

プロジェクト内の Analytics テーブルの内容を印刷するには、DO REPORT コマンドを使用します。

PROFILE コマンド

Analytics テーブルで 1 つまたは複数の数値フィールドまたは数値式について要約統計値を生成します。

構文

PROFILE {<FIELDS> *数値フィールド* <...*n*>|<FIELDS> ALL} <IF *テスト*> <WHILE *テスト*> <FIRST *範囲*|NEXT *範囲*>

パラメーター

名前	説明
FIELDS *数値フィールド* <...*n*> | FIELDS ALL	個別のフィールドを指定してプロファイルするか、ALL を指定して Analytics テーブルのすべての数値フィールドをプロファイルします。
IF *テスト* 省略可能	各レコードを処理するために真である必要がある条件式。コマンドは、その条件を満たすレコードに対してのみ実行されます。 **メモ** IF パラメーターは、任意の範囲パラメーター(WHILE、FIRST、NEXT) が適用された後に、テーブルに残るレコードに対してのみ評価されます。
WHILE *テスト* 省略可能	各レコードを処理するために真である必要がある条件式。条件が false と評価するか、テーブルの最後に達したら、コマンドは実行を中止します。 **メモ** WHILE を FIRST または NEXT とともに使用する場合は、1 つの制限に達するとすぐに、レコードの処理が停止します。
FIRST *範囲* | NEXT *範囲* 省略可能	処理するレコード数: ◦ **FIRST** - 指定されたレコード数に達するまで、最初のレコードから処理を開始します ◦ **NEXT** - 指定されたレコード数に達するまで、現在選択されているレコードから処理を開始します *範囲*は処理するレコード数を指定します。 FIRST と NEXT を省略すると、すべてのレコードがデフォルトで処理されます。

例

単一フィールドのプロファイルを作成する

Salary フィールドのプロファイルを作成するコマンドを次に示します。

```
OPEN Employee_Payroll
PROFILE FIELDS Salary
```

このコマンドにより、次の出力が生成されます。

フィールド名	合計値	絶対値	最小	最大
給与(Salary)	1,152,525	1,152,525	15,340	52,750

備考

出力に表示される統計

コマンドに指定された数値フィールドまたは数値式ごとに次の統計が表示されます。

- 合計値
- 絶対値
- 最小値
- 最大値

QUIT コマンド

現在のセッションを終了し、Analytics を終了します。

構文

```
QUIT
```

例

ファイルが存在するかどうかを確認して、存在しない場合に Analytics を終了させる例を考えます。

他のユーザーに実行させるスクリプトを作成しましたが、必要なファイルが存在しない場合に Analytics を終了させる必要があるとします。

次の例では、必要な `Inventory.csv` ファイルが存在するかどうかを確認して、存在しない場合に Analytics を終了させています。

```
IF FILESIZE("Inventory.csv") = -1 QUIT
```

スクリプトが完了したら自動的に Analytics を終了させる

次のスクリプトは、Inventory テーブルを要約し、出力結果を生成して、自動的に Analytics を終了させています。

```
OPEN Inventory
SUMMARIZE ON Location ProdCls SUBTOTAL Value TO "Inventory_value_by_location_class.FIL"
PRESORT CPERCENT
QUIT
```

備考

変更は保存される

QUIT を実行すると、Analytics が終了させられる前に、開いているすべての Analytics テーブルが保存されて閉じられます。

アクティブなビューやスクリプトの変更を保存していない場合、Analytics は終了前にその変更を保存するように指示するメッセージを表示します。

RANDOM コマンド

一連の乱数を生成します。

構文

RANDOM NUMBER *n* <SEED *シード値*> MINIMUM *最小値* MAXIMUM *最大値* <COLUMNS *n*> <UNIQUE> <SORTED> <TO {SCREEN|*ファイル名*> <APPEND>

パラメーター

名前	説明
NUMBER *n*	生成される乱数一式のサイズ。 最大 32767 の数字を生成できます。
SEED *シード値* 省略可能	乱数ジェネレーターを初期化するために使用される値。 シード値を指定する場合、任意の数字を指定することができます。シード値がそれぞれ一意であると、異なる乱数一式になります。同じシード値を指定すると、同じ乱数一式が生成されます。分析を複製する場合は、同じ乱数一式を再生成しなければならないことがあります。 ◦ **シード値** -には、特定の乱数一式を複製したい場合のシード値を明示的に指定し、保存します。 ◦ **シード値なし** -には、シード値を Analytics にランダムに選択させたい場合に、'0' またはブランクのシード値を入力します。
MINIMUM *最小値*	乱数一式に使用可能な最小の数値。有効な任意の数値または数式が使用できます。
MAXIMUM *最大値*	乱数一式に使用可能な最大の数値。有効な任意の数値または数式が使用できます。
COLUMNS *n* 省略可能	乱数一式の表示に使用する列数。 COLUMNS を省略した場合のデフォルトは 6 列です。
UNIQUE 省略可能	一意の数値のみが乱数一式に含まれます。 UNIQUE を省略した場合は、乱数一式内で重複する値が生成される可能性があります。 **メモ** 乱数一式の指定サイズが最小値と最大値の差異の 75% を超える場合は、UNIQUE を選択しないでください。選択すると、乱数の選択肢が破棄される数が多くなりすぎます。
SORTED 省略可能	乱数一式が昇順で表示されます。 SORTED を省略した場合は、数字が無作為な順序で選択されて表示されます。

名前	説明
TO SCREEN | *ファイル名* 省略可能	コマンドの結果を送信する場所： ◦ **SCREEN** - はAnalyticsの表示領域に結果を表示します ◦ ***ファイル名*** -は結果の保存先となるファイルです。 *ファイル名*には、適切なファイル拡張子を付けた文字列を引用符で囲んで指定します。例：TO "Output.TXT" デフォルトでは、テーブルファイルは、Analyticsプロジェクトが入っているフォルダーに保存されます。 既存の異なるフォルダーにファイルを保存するには、絶対または相対ファイルパスを指定します。 • TO "C:\Output.TXT" • TO "Results\Output.TXT" TOを省略した場合は、乱数一式が画面に出力されます。
APPEND 省略可能	コマンドの出力を既存ファイルに上書きしないで、そのファイルの末尾に追加します。 **メモ** コマンドの出力と既存のファイルの構造が同一であるようにする必要があります。 • 同じフィールド • 同じフィールド順序 • 一致するフィールドが同じ長さ • 一致するフィールドが同じデータ型 出力は、既存ファイルとの間でファイル構造が違っている場合でも、Analyticsによって既存ファイルに追加されます。出力と既存のファイルの構造が一致しない場合は、データが混在、不足、不正確になります。

例

100個の乱数が含まれるテキストファイルを生成する

10,000 ～ 20,000の番号が付いた一連のファイルから、ランダムに100個のハードコピーファイルを取り出したいとします。

10,000 ～ 20,000の範囲に含まれる100個の乱数が含まれたテキストファイルを生成するには、RANDOMコマンドを使用することができます。その後で、それらの乱数に一致する番号の付いたハードコピーファイルを取り出します。これらの番号は、一意であり、10個の列に昇順で並べられます。

```
RANDOM NUMBER 100 SEED 45387 MINIMUM 10000 MAXIMUM 20000 COLUMNS 10 UNIQUE
SORTED TO "Random_Numbers.txt"
```

備考

> **メモ**
> このコマンドの動作の詳細については、Analytics のヘルプを参照してください。

乱数アルゴリズム

RANDOM コマンドでは、Analytics のデフォルトの乱数アルゴリズムが使用されます。SAMPLE コマンドと異なり、RANDOM コマンドではメルセンヌツイスター乱数アルゴリズムは使用できません。

RCOMMAND コマンド

Analytics テーブルを外部 R スクリプトに**データ フレーム**として渡し、外部 R スクリプトからの出力を使用して、Analytics プロジェクトで新しいテーブルを作成します。

構文

RCOMMAND FIELDS *フィールド* <*...n*> RSCRIPT *スクリプトへのパス* TO *テーブル名* <IF *テスト*> <WHILE *テスト*> <FIRST *範囲*|NEXT *範囲*> <KEEPTITLE> <SEPARATOR *文字*> <QUALIFIER *文字*> <OPEN>

パラメーター

名前	説明
FIELDS *フィールド名* <*...n*>	R スクリプトに送信されるデータ フレームに追加する、ソースの Analytics テーブル内のフィールド、あるいは式。 使用する Analytics のエディションによっては、次の特殊文字を含むデータを R スクリプトに送信するときにエラーが発生する可能性があります。 ◦ **非 Unicode** - "\" ◦ **Unicode** - "ÿ" または "Ŝ" ◦ **両方** -ボックスは、ブロック、黒い四角、および垂直の破損した棒などの文字を描画します **メモ** 日本語の文字と中国語の文字が両方とも含まれているテーブルなど、複数の言語が混在するデータも、サポートされていません。
RSCRIPT *スクリプトのパス*	R スクリプトの、ファイル システムにおける絶対パスまたは相対パス。*スクリプトへのパス*は引用符で囲んでください。
TO *テーブル名*	コマンドの結果を送信する場所： ◦ ***テーブル名*** -は、結果の保存先となる Analytics テーブルのことです。 *テーブル名*には、ファイル拡張子 .FIL を付けた文字列を引用符で囲んで指定する必要があります。例：TO "Output.FIL" デフォルトでは、テーブル データ ファイル(.fil) は、Analytics プロジェクトが入っているフォルダーに保存されます。 既存の異なるフォルダーにデータ ファイルを保存するには、絶対または相対ファイル パスを指定します。 • TO "C:\Output.FIL" • TO "Results\Output.FIL"

名前	説明
	メモ テーブル名は 64 文字の英数字(.FIL 拡張子を含まない) に制限されています。名前にはアンダースコア文字(_) を使用できますが、他の特殊文字やスペースは使用できません。名前の先頭を数字にすることはできません。 R スクリプトによって返すデータ フレームまたはマトリクスから作成する出力テーブルの名前。
IF *テスト* 省略可能	現在のレコードを処理するのに満たす必要がある条件。R スクリプトに渡されるデータ フレームには、この条件を満たすレコードのみが含まれます。
WHILE *テスト* 省略可能	各レコードを処理するために真である必要がある条件式。条件が false と評価するか、テーブルの最後に達したら、コマンドは実行を中止します。 **メモ** WHILE を FIRST または NEXT とともに使用する場合は、1 つの制限に達するとすぐに、レコードの処理が停止します。
FIRST *範囲* | NEXT *範囲* 省略可能	処理するレコード数: ◦ **FIRST** - 指定されたレコード数に達するまで、最初のレコードから処理を開始します ◦ **NEXT** - 指定されたレコード数に達するまで、現在選択されているレコードから処理を開始します *範囲*は処理するレコード数を指定します。 FIRST と NEXT を省略すると、すべてのレコードがデフォルトで処理されます。 **注意** 現在のバージョンでは、RCOMMAND を実行する際、NEXT の動作に問題があることが知られています。このオプションは使用しないでください。選択しているレコードに関係なく、レコード参照で最初のレコードが参照されるようにリセットされてしまう可能性があるためです。
KEEPTITLE 省略可能	データの代わりに、フィールド名としてデータの最初の行を処理します。省略すると、汎用フィールド名が使用されます。 このオプションは、R スクリプトの列名を使ってデータを取得したい場合に必要です。
SEPARATOR *文字* 省略可能	フィールド間の区切りとして使用する文字。文字は引用符で囲まれた文字列として指定する必要があります。 デフォルトの <文字> はカンマです。 **メモ** 入力フィールドに表示される文字を使用しないでください。SEPARATOR 文字が入力データに表示される場合は、結果に影響する可能性があります。
QUALIFIER *文字* 省略可能	フィールド値を折り返すため と識別するためにテキスト修飾子として使用する文字。文字は引用符で囲まれた文字列として指定する必要があります。 デフォルトの <文字> は二重引用符です。

名前	説明
	メモ 入力フィールドに表示される文字を使用しないでください。QUALIFIER 文字が入力データに表示される場合は、結果に影響する可能性があります。
OPEN 省略可能	コマンドを実行した後、コマンドによって作成されたテーブルを開きます。コマンドが出力テーブルを作成する場合にのみ有効です。

例

R スクリプト(Hello world) を起動し、稼働する

Hello world スクリプトを作成し、Analytics と R スクリプトの接続をテストする場合を考えます。

Analytics コマンド

```
RCOMMAND FIELDS "Hello", ", world!" TO "r_result" RSCRIPT "C:\scripts\r_scripts\analysis.r"
```

R スクリプト(analysis.r)

```
srcTable<-acl.readData()

# ACL に返送するテーブルを作成
output<-data.frame(
 c(srcTable[1,1]),
 c(srcTable[1,2])
)

# 列名を追加して ACL にテーブルを返送
colnames(output) <- c("Greeting","Subject")
acl.output<-output
```

行と列の座標を使用してフィールド データにアクセスする

いくつかの請求書関連フィールドを Analytics 外での分析のため R スクリプトに送信するとします。

Analytics コマンド

```
RCOMMAND FIELDS Department_Code Invoice_Amount Invoice_Date Invoice_Number Vendor_
Number TO "r_result" RSCRIPT "C:\scripts\r_scripts\analysis.r"
```

R スクリプト(analysis.r)

```
# R スクリプトでデータ フレームの 2 行目から請求書番号を取り出す
srcTable<-acl.readData()
srcTable[2,4]
```

列名を使用してフィールド データにアクセスする

いくつかの請求書関連フィールドを Analytics 外での分析のため R スクリプトに送信するとします。列が R スクリプト内で名前で呼び出せるように、KEEPTITLE オプションを使用します。

Analytics コマンド

```
RCOMMAND FIELDS Department_Code Invoice_Amount Invoice_Number TO "r_result" RSCRIPT
"C:\scripts\r_scripts\analysis.r" KEEPTITLE
```

R スクリプト(analysis.r)

```
# R スクリプトでデータ フレームの 2 行目から請求書番号を取り出す
srcTable<-acl.readData()
srcTable["2","Invoice_Number"]
```

1000.00 の金額を超える請求書レコードを R スクリプトに送信する

いくつかの請求書関連フィールドを Analytics 外での分析のため R スクリプトに送信するとします。R スクリプトに送信するレコード数を制限するには、IF を使用します。1000.00 を超える請求金額が含まれるレコードのみが送信されます。

Analytics コマンド

```
RCOMMAND FIELDS Department_Code Invoice_Amount Invoice_Number TO "r_result" IF Invoice_
Amount > 1000.00 RSCRIPT "C:\scripts\r_scripts\analysis.r" KEEPTITLE
```

R スクリプト(analysis.r)

```
# R スクリプトでデータ フレームの 2 行目から請求書番号を取り出す
srcTable<-acl.readData()
srcTable["2","Invoice_Number"]
```

請求書レコードを送信し、乗算した請求書金額を返す

いくつかの請求書関連フィールドを Analytics 外での分析のため R スクリプトに送信するとします。この R スクリプ

トでは、指定した列のすべてのセルに対して単一のアクションが実行されます。

Analytics コマンド

```
RCOMMAND FIELDS Department_Code Invoice_Amount Invoice_Number TO "r_result" RSCRIPT
"C:\scripts\r_scripts\analysis.r" KEEPTITLE
```

R スクリプト(analysis.r)

```
# 2 倍の値の ACL テーブルのスライスを返す
srcTable<-acl.readData()
acl.output<-srcTable["Invoice_Amount"] * 2
```

備考

> **メモ**
> このコマンドの動作の詳細については、Analytics のヘルプを参照してください。

R スクリプトでの Analytics データの参照

Analytics テーブルは R **データ フレーム**としてスクリプトに渡されます。データ フレームはタブ形式のデータ オブジェクトであり、データの異なるモードまたは型の列を含むことがあります。

R スクリプトで Analytics によって作成されたデータ型を操作するには、acl.readData() 関数を呼び出し、返されたデータ フレームを変数に格納します。

```
# Analytics テーブルをデータ フレーム myTable に格納します。これはスクリプトを使用して参照できます
myTable<-acl.readData()
```

データ フレームのセルからデータを取得するには、次のアプローチのうちの 1 つを使用できます。

- 行と列の座標の使用:

  ```
  # データ フレームの 1 行 2 列の値を取得します。
  myTable[1,2]
  ```

  > **メモ**
  > 座標はコマンドで指定されたフィールドの順序に基づいているのであり、テーブル レイアウトや、現在開いているビューに基づいているのではありません。

- 行と列の名前の使用:

```
# データフレームの 1 行 "myColumnTitle" 列の値を取得します。
myTable["1","myColumnTitle"]
```

列の名前を使用するには、コマンドの **KEEPTITLE** オプションを指定する必要があります。

行は、"1"、"2"、"3" と命名されており、そのように増加します。名前と座標の組み合わせを使用することもできます。

Analytics へのデータの返送

データフレームまたはマトリクスを Analytics に戻し、新しいテーブルを作成するには、次の構文を使用します。

```
# myNewTable データフレームを Analytics に戻し、新しいテーブルを作成します
acl.output<-myNewTable
```

> **メモ**
>
> R スクリプトが終了するときに、データフレームまたはマトリクスを Analytics に戻す必要があります。データフレームまたはマトリクスの列には原子値のみが含まれ、リスト、マトリクス、配列、非原子オブジェクトがないことを確認します。値を Analytics データ型に変換できない場合は、コマンドが失敗します。

データ型のマッピング

Analytics データ型は、Analytics プロジェクトと R スクリプトの間の翻訳プロセスを使用して、R データ型に変換されます。

Analytics データ型	R データ型
論理	論理
数値	数値
文字	文字
日付時刻	日付、POSIXct、POSIXlt

パフォーマンスとファイルサイズ制限

R スクリプトの実行と返されるデータの処理にかかる時間は、入力データが 1 GB を超える場合には長くなります。R では、2 GB 以上の入力ファイルはサポートされていません。

また、R に送信されるレコード数もパフォーマンスに影響します。ファイルサイズは同じだがレコード数が異なる 2 つのテーブルでは、レコード数が少ないテーブルの方が処理が高速になります。

複数バイト文字データの処理

中国語のような、複数バイトの文字セットでデータをRに送信する場合、Rスクリプトにシステムロケールを適切に設定する必要があります。複数バイトのデータのテーブルをRに正常に送信するには、Rスクリプトの第1行に、次の関数が含まれている必要があります。

```
#ロケールを中国語に設定する例
Sys.setlocale("LC_ALL","Chinese")
```

Sys.setlocale( ) の詳細については、Rドキュメントを参照してください。

R のログファイル

R言語のメッセージは、Analyticsによりプロジェクトフォルダーの `aclrlang.log` ファイルに記録されます。このファイルを使って、Rのエラーをデバッグします。

> **ヒント**
>
> ログファイルは Analytics Exchange のアナリティクスジョブの結果フォルダーにあります。

AX Server での R スクリプトの実行

AX Serverで実行する分析アプリを作成していて、外部のRスクリプトを使用したい場合は、次の手順を実行します。

1. 分析アプリとともに、このファイルを関連ファイルとしてアップロードします。
2. このファイルを指定する際、FILE ANALYTIC タグを使用します。
3. ファイルの参照には、相対パス `./filename.r` を使用します。

> **メモ**
>
> 関連ファイルを使用することで、Analytics ExchangeとともにRを実行する際に、TomEE アプリケーション サーバーのアカウントに、このファイルにアクセスするための十分な権限が与えられます。

REFRESH コマンド

関連するデータソースを基に、Analytics テーブル内のデータを更新します。

構文

```
REFRESH <テーブル名> <PASSWORD 番号>
```

パラメーター

名前	説明
テーブル名 省略可能	更新する Analytics テーブルの名前。*テーブル名*を指定しない場合、開いているテーブルが更新されます。
PASSWORD *番号* 省略可能	使用するパスワード定義。 実際のパスワードを入力することを求めたり指定したりするのには、PASSWORD *番号*の構文は使用しません。パスワード定義とは、以前に PASSWORD コマンドか SET PASSWORD コマンドを使用して入力または設定されたパスワードのことです。 *番号*はパスワード定義の番号です。たとえば、以前に 2 つのパスワードをスクリプトで設定したり入力したりしている場合には、PASSWORD 2 により、2 番目のパスワードを使用することを指定するなどします。 パスワードの入力または設定の詳細については、次を参照してください。 • "PASSWORD コマンド" ページ 345 • "SET コマンド" ページ 404 • PASSWORD アナリティクス タグ パスワードの入力または設定の詳細については、次を参照してください。 • PASSWORD コマンド • SET コマンド • PASSWORD アナリティクス タグ **メモ** このパスワードは、元のソース データ システムにアクセスするのに使用します。 PDF を除き、ファイルベースのデータソースでパスワードでは、REFRESH を使用できません。

例

パスワードを要求せずにテーブルを更新する

データソースにパスワードが必要でない場合は、REFRESHコマンドと、更新するAnalyticsテーブルの名前を入力します。

```
REFRESH Invoices
```

対話型のスクリプト内でユーザーに入力するよう求めたパスワードを使ってテーブルを更新する

対話型のスクリプトを作成している場合は、ユーザーにパスワードの入力を求めることができます。

```
PASSWORD 1 "パスワードを入力してください:"
REFRESH Invoices PASSWORD 1
```

ACCESSDATAコマンドを使用して、パスワード保護されたデータソースから最初にインポートされたテーブルを更新する場合、パスワード プロンプトは自動であり、個別に指定する必要はありません。

```
REFRESH Invoices
```

非対話型のスクリプト内に設定したパスワードを使ってテーブルを更新する

パスワード値の入力をユーザーに求めないようにするには、スクリプト内にパスワードを設定します。

```
SET PASSWORD 1 TO "password"
REFRESH Invoices PASSWORD 1
```

この方法の欠点 h

AX Serverアナリティクス内でユーザーに入力するよう求めたパスワードを使ってテーブルを更新する

AX Serverアナリティクスを作成している場合は、アナリティクスの予定実行時にパスワードの入力をユーザーに求めることができます。あるいは、アドホックで次のコマンドを実行します。

```
COMMENT
//ANALYTIC Refresh Table
//PASSWORD 1 "パスワードを入力してください:"
END
REFRESH Invoices PASSWORD 1
```

備考

> **メモ**
> このコマンドの動作の詳細については、Analytics のヘルプを参照してください。

機能の仕組み

REFRESH コマンドは、テーブルを定義およびインポートするときに最初に使用される IMPORT コマンドまたは ACCESSDATA コマンドを再実行して、テーブルの内容を更新します。

REFRESH でテーブル内容のみが更新される

REFRESH コマンドでは、Analytics テーブルにおける既存のフィールドの内容のみが更新されます。Analytics テーブルレイアウトは更新されません。

ソースデータの構造が変更された場合 (フィールドが追加または削除された場合など) は、REFRESH を使用できません。データを再インポートする必要があります。

更新がサポートされるデータソース

REFRESH コマンドを使用すると、以下のコマンドのいずれかを使用して作成された Analytics テーブルの内容を更新できます。

- IMPORT ACCESS
- IMPORT DELIMITED
- IMPORT EXCEL
- IMPORT ODBC(レガシー ODBC コマンド)
- IMPORT PDF
- IMPORT PRINT
- IMPORT SAP
- IMPORT XBRL
- IMPORT XML
- ACCESSDATA(ODBC データソース)

REFRESH と ACCESSDATA

次のガイドラインは、ACCESSDATA コマンドを使用して ODBC データソースからインポートされたテーブルを更新するときに適用されます。

- **開いているテーブル** - 更新するときにテーブルが開いている場合は、一時的に、テーブルのサイズの 2 倍に相当するディスク領域が必要です。ディスク領域が限られている場合は、更新する前にテーブルを閉じてください。
- **Analytics 12** - Analytics バージョン 12 で ACCESSDATA コマンドを使用してインポートされたテーブルは、新しいバージョンの Analytics を使用している場合でも更新できません。

  これらのテーブルを更新するには、Analytics 12.5 以降を使用して再インポートしてください。

REFRESH とパスワード

データベースまたはクラウド データ サービスに存 在 するパスワードで保 護されたデータソースで、REFRESH コマンドを使 用できます。

Excel ファイルなどのパスワードで保 護されたファイルベースのデータソースでは、REFRESH コマンドを使 用できません。ただし、パスワードで保 護された PDF は例 外 です。

REFRESH と分 析 アプリ ウィンドウ

分 析 アプリ ウィンドウで実 行 する予 定 のスクリプトでは、REFRESH コマンドを使 用しないでください。

テーブルのインポート方 法 によっては、テーブルのデータの更 新 がサポートされていないか、分 析 アプリ ウィンドウで試 行 された場 合 に予 期しない結 果 を生 成します。

分 析 アプリ ウィンドウで実 行 されるスクリプトの一 部 としてデータを更 新 する場 合 は、IMPORT コマンドまたは ACCESSDATA コマンドを使 用し、テーブルを上 書きします。

RENAME コマンド

Analytics プロジェクト項目またはファイルの名前を変更します。

構文

RENAME *項目の種類 名前* <AS|TO> *新しい名前* <OK>

パラメーター

名前	説明
項目の種類 名前	名前の変更元となるプロジェクト項目またはファイルの種類および名前。 **メモ** ほとんどの場合、アクティブであるか、開いているか、使用中の場合には、項目またはファイル名を変更できません。 次に示す有効な種類のうち、いずれかを指定します。 ◦ **FIELD** - 物理的なデータフィールド、演算フィールド、または変数 • フィールドを含むテーブルが開いている必要があります。ただし、アクティブなビューにはフィールドを含めることができません。 • ただし、演算フィールドが参照しているフィールドの名前を変更することはできません。 ◦ **FORMAT** - Analytics テーブル ◦ **INDEX** - インデックス ◦ **REPORT** - レポートまたはビュー ◦ **WORKSPACE** - ワークスペース ◦ **SCRIPT(またはBATCH** - スクリプト ◦ **DATA** - Analytics データファイル (*.FIL) ◦ **FILE** - ファイルシステムのデータファイル ◦ **LOG** - Analytics ログファイル (.log) ◦ **TEXT** - テキストファイル
AS | TO *新しい名前*	プロジェクト項目またはファイルの新しい名前 **メモ** 長さ制限はほとんどの Analytics プロジェクト項目名に適用されます。詳細については、Analytics における文字およびサイズの制限を参照してください。
OK 省略可能	アクションを確認せずに、項目を削除または上書きします。

例

フィールド名の変更

ProdNo フィールドの名前を **ProdNum** に変更する場合を考えます。追加の確認を行わずにこのアクションを実行する場合は、「OK」を使用します。以上を行うコマンドを次に示します。

```
OPEN Inventory
RENAME FIELD ProdNo AS ProdNum OK
```

REPORT コマンド

開いている Analytics テーブルに基づき、レポートを書式設定し、生成します。

構文

REPORT <ON *内訳フィールド* <PAGE> <NODUPS> <WIDTH *文字数*> <AS *表示名*>> <...*n*> FIELD *その他のフィールド* <WIDTH *文字数*> <AS *表示名*> <...*n*> <SUPPRESS> <NOZEROS> <LINE *n その他のフィールド*> <PRESORT <*並べ替えフィールド*>> <...*n*> <SUMMARIZED> <SKIP *n*> <EOF> <TO {SCREEN|PRINT|*ファイル名* <HTML>}> <IF *テスト*> <WHILE *テスト*> <FIRST *範囲*|NEXT *範囲*> <HEADER *ヘッダーテキスト*> <FOOTER *フッターテキスト*> <APPEND>

パラメーター

名前	説明
ON *内訳フィールド* PAGE NODUPS WIDTH *文字* AS *表示名* <...*n*> 省略可能	レポートをセクションに区切るために使用される文字フィールド。 *内訳フィールド*の値が変わるたびに、新しいレポート セクションと小計が作成されます。レポートをセクションに分割することで、レポートがスキャンしやすくなります。 ◦ ***ブレークフィールド*** -ブレークフィールドとして使用するフィールド ビューに基づいたレポートを実行(DO REPORT) するには、ビューにおける左端の 1 つまたは複数の文字型フィールドを内訳フィールドとする必要があります。 ◦ **PAGE** -は、内訳フィールド値が変化するたびに改ページを挿入します。 ◦ **NODUPS** -ブレークフィールドから重複表示値を抑制します たとえば、請求書レコードごとに顧客名が記載されている場合、各顧客名の最初のインスタンスだけを記載すれば、レポートが読みやすくなるかもしれません。 ◦ **WIDTH *文字*** -フィールドの出力長さ(文字数) ◦ **AS *表示名*** -レポートのフィールドの表示名(代替列タイトル) *表示名*を文字列として指定します。列見出しを改行したい場合は、語句の間にセミコロン(;) を入れます。表示名をフィールド名、またはソース テーブル内の既存の表示名と同じにしたい場合は、AS を使用しないでください。 **メモ** *内訳フィールド*、PAGE、NODUPS、または PRESORT を使用するには、ON を指定する必要があります。
FIELD *他のフィールド* WIDTH *文字* AS *表示名* <...*n*>	レポートに含めるフィールド。 ◦ **WIDTH *文字*** -フィールドの出力長さ(文字数) ◦ **AS *表示名*** -レポートのフィールドの表示名(代替列タイトル) *表示名*を文字列として指定します。列見出しを改行したい場合は、語句の間にセミコロン(;) を入れます。表示名をフィールド名、またはソース テーブル内の既存の表示名と同じにしたい場合は、AS を使用しないでください。

名前	説明
	SUBTOTAL および ACCUMULATE キーワードは、廃止され、FIELD に取って代わられました。すべての数値フィールドの小計は、自動で計算されます。 **メモ** 内訳フィールドは、自動的にレポートに追加されるため、*その他のフィールド*に指定する必要はありません。
SUPPRESS 省略可能	レポートから空白の明細行を除外します。
NOZEROS 省略可能	フィールドのゼロ値に空白値を使用します。 たとえば、レポートのフィールド大量のゼロ値がある場合、ゼロ以外の値のみを表示すると読みやすくなります。
LINE *n その他のフィールド* 省略可能	行番号 *n* に表示される列とフィールドの出力行数を指定します。 値が指定されない場合、列は単一行をデフォルトとします。*n* には、2 ～ 60 に含まれる値(両端を含む)を指定する必要があります。 レポートの列見出しは、最初の行にあるフィールドによって決まります。*その他のフィールド*は、レポートに使用するフィールドや式を指定します。
PRESORT *並べ替えフィールド* <...*n*> 省略可能	◦ 並べ替え *内訳フィールド* 、1つ以上の内訳フィールドが指定されている場合。 ◦ 並べ替え *並べ替えフィールド*, 1つ以上の並べ替えフィールドが指定されている場合。 PRESORT は、*並べ替えフィールド*としてリストされていない限り、*その他のフィールド*としてリストされたフィールドを並べ替えません。
SUMMARIZED 省略可能	詳細行のない、小計と合計のみのレポートを作成します。 小計は、一意の内訳フィールドの値に対して生成されます。SUMMARIZED を指定しない場合には、KEY で指定した各内訳フィールドの小計だけでなく詳細行も含んだレポートが作成されます。
SKIP *n* 省略可能	レポート内の詳細行の間に空白行を挿入します。 *n* には、挿入する行数を示す整数を指定する必要があります。たとえば、`SKIP 1` は 1 行おきの行間のレポートを生成します。
EOF 省略可能	ファイルの終わりに達した後、コマンドをもう一度実行します。 これにより、GROUP コマンド内でテーブルの最後のレコードが処理されることが保証されます。すべてのフィールドが以前のレコードを参照する演算フィールドである場合にのみ使用してください。
TO SCREEN | PRINT| *ファイル名* <HTML> 省略可能	コマンドの結果を送信する場所: ◦ **SCREEN** - は Analytics の表示領域に結果を表示します ◦ ***ファイル名*** -は結果の保存先となるファイルです。 *ファイル名*には、適切なファイル拡張子を付けた文字列を引用符で囲んで指定します。 例: `TO "Output.TXT"` デフォルトでは、テーブルファイルは、Analytics プロジェクトが入っているフォルダーに保存されます。

名前	説明
	既存の異なるフォルダーにファイルを保存するには、絶対または相対ファイル パスを指定します。 • TO "C:\Output.TXT" • TO "Results\Output.TXT" ◦ **印刷** -- 通常使うプリンターに結果を送信します デフォルトでは、レポートはファイルに出力され、ASCII テキスト ファイルとして保存されます。レポートを HTML ファイル(.htm) として出力したい場合は、HTML を指定します。 TO を省略した場合は、レポートが画面に出力されます。
IF *テスト* 省略可能	各レコードを処理するために真である必要がある条件式。コマンドは、その条件を満たすレコードに対してのみ実行されます。 **メモ** IF パラメーターは、任意の範囲パラメーター(WHILE、FIRST、NEXT) が適用された後に、テーブルに残るレコードに対してのみ評価されます。
WHILE *テスト* 省略可能	各レコードを処理するために真である必要がある条件式。条件が false と評価するか、テーブルの最後に達したら、コマンドは実行を中止します。 **メモ** WHILE を FIRST または NEXT とともに使用する場合は、1 つの制限に達するとすぐに、レコードの処理が停止します。
FIRST *範囲* | NEXT *範囲* 省略可能	処理するレコード数: ◦ **FIRST** - 指定されたレコード数に達するまで、最初のレコードから処理を開始します ◦ **NEXT** - 指定されたレコード数に達するまで、現在選択されているレコードから処理を開始します *範囲*は処理するレコード数を指定します。 FIRST とNEXT を省略すると、すべてのレコードがデフォルトで処理されます。
HEADER *ヘッダーテキスト* 省略可能	レポートの各ページの最上部に挿入されるテキスト。 *ヘッダーテキスト*は引用符で囲んだ文字列として指定する必要があります。この値は、Analytics のシステム変数である HEADER の値よりも優先されます。
FOOTER *フッターテキスト* 省略可能	レポートの各ページの最下部に挿入されるテキスト。 *フッターテキスト*は引用符で囲んだ文字列として指定する必要があります。この値は、Analytics のシステム変数である FOOTER の値よりも優先されます。
APPEND 省略可能	コマンドの出力を既存ファイルに上書きしないで、そのファイルの末尾に追加します。

名前	説明
	メモ コマンドの出力と既存のファイルの構造が同一であるようにする必要があります。 • 同じフィールド • 同じフィールド順序 • 一致するフィールドが同じ長さ • 一致するフィールドが同じデータ型 出力は、既存ファイルとの間でファイル構造が違っている場合でも、Analytics によって既存ファイルに追加されます。出力と既存のファイルの構造が一致しない場合は、データが混在、不足、不正確になります。

例

HTML レポートを生成する

次の例では、Ar テーブルからレポートを生成し、書式付きの HTML ファイルに出力します。

```
OPEN Ar
REPORT ON No FIELD Due Type Amount TO "C:\Reports\AR.htm" HTML
```

RETRIEVE コマンド

バックグラウンド処理に送信された *Direct Link* クエリの結果を取得します。

構文

RETRIEVE *テーブル名* PASSWORD *番号*

パラメーター

名前	説明
テーブル名	Direct Link クエリによって Analytics 内に最初に作成されるテーブルの名前。 RETRIEVE を使用する前に、テーブルが存在する必要があります。
PASSWORD *番号*	使用するパスワード定義。 実際のパスワードを入力することを求めたり指定したりするのには、PASSWORD *番号*の構文は使用しません。パスワード定義とは、以前に PASSWORD コマンドか SET PASSWORD コマンドを使用して入力または設定されたパスワードのことです。 *番号*はパスワード定義の番号です。たとえば、以前に 2 つのパスワードをスクリプトで設定したり入力したりしている場合には、PASSWORD 2 により、2 番目のパスワードを使用することを指定するなどします。 パスワードの入力または設定の詳細については、次を参照してください。 • "PASSWORD コマンド" ページ 345 • "SET コマンド" ページ 404 • PASSWORD アナリティクス タグ パスワードの入力または設定の詳細については、次を参照してください。 • PASSWORD コマンド • SET コマンド • PASSWORD アナリティクス タグ **メモ** このパスワードは、SAP システムにアクセスするのに使用します。

例

バックグラウンド モードでのクエリの結果を取得する

次の例では、パスワードを設定した後、**DD02T_Data** という名前の Analytics テーブルに対するバックグラウンド モードでのクエリの結果を取得します。

```
SET PASSWORD 1 TO "pwd"
RETRIEVE DD02T_Data PASSWORD 1
```

備考

はじめに

このコマンドは、Direct Link がインストールおよび設定されている場合にのみサポートされます。

SAMPLE コマンド

レコード サンプリングまたは金額単位サンプリング方法を使用して、レコードのサンプルを抽出します。

レコードのサンプリング 金額単位のサンプリング

構文

> **メモ**
>
> フィルターリング(IF ステートメント) や範囲パラメーターを適用するとサンプルの妥当性を低下させてしまうため、それらのオプションはこの構文に含まれていません。

固定間隔選択方法

SAMPLE <ON> RECORD INTERVAL *間隔値* <FIXED *最初の値*> {RECORD|FIELDS *フィールド名* <...*n*>} TO *テーブル名* <OPEN> <APPEND> <LOCAL>

セル選択方法

SAMPLE <ON> RECORD CELL INTERVAL *間隔値* <RANDOM *シード値*> {RECORD|FIELDS *フィールド名* <...*n*>} TO *テーブル名* <OPEN> <APPEND> <MERSENNE_TWISTER> <LOCAL>

ランダム選択方法

SAMPLE <ON> RECORD NUMBER *サンプルサイズ*<RANDOM *シード値*> <ORDER> {RECORD|FIELDS *フィールド名* <...*n*>} TO *テーブル名* <OPEN> <APPEND> <MERSENNE_TWISTER> <LOCAL>

パラメーター

> **メモ**
>
> 値を指定する際、3 桁の区切り記号は含めないでください。

名前	説明
ON RECORD	レコード サンプリングを使用します。
INTERVAL *間隔値* FIXED	**INTERVAL *間隔値* FIXED *最初の値***

名前	説明
最初の値 | CELL INTERVAL *間隔値* | NUMBER *サンプルサイズ*	**固定間隔**選択方法を使用します。 最初のレコードを選択すると、そのレコードから固定した間隔、つまり固定した距離だけ離れたレコードが続いて選択されます。たとえば、最初に選択したレコードの後は、20 件に 1 件のレコードが選択されます。 ◦ **INTERVAL *の間隔値*** -には、サンプル サイズの計算によって出力された間隔値を指定します。 ◦ **FIXED *の最初の値*** -には、選択する最初のレコード番号を指定します。 *最初の値*にゼロ('0') を指定するか、FIXED を省略した場合は、Analytics により最初のレコードがランダムに選択されます。 **CELL INTERVAL *間隔値*** **セル**選択方法を使用します。 データ セットが複数の均等なセルまたはグループに分割され、各セルから 1 つのレコードがランダムに選択されます。 各セルのサイズは*間隔値*によって指定します。サンプル数の計算によって生成された間隔の値を入力します。 **NUMBER *サンプルサイズ*** **ランダム**選択方法を使用します。 すべてのレコードがデータセット全体からランダムに選択されます。 サンプル サイズの計算によって生成されたサンプル サイズを入力します。
RANDOM *シード値* 省略可能	**メモ** セルおよびランダム選択方法の場合のみ。 Analytics の乱数ジェネレーターを初期化するために使用するシード値。 ゼロ('0') の値を指定するか、または RANDOM を省略した場合は、シード値がランダムに選択されます。
ORDER 省略可能	**メモ** ランダム選択方法の場合のみ。 FIELDS を指定する場合には、ORDER のみを使用できます。 出力結果に ORDER フィールドを追加します。 このフィールドには、各レコードの選択順序がランダムであることが表示されます。
RECORD | FIELDS *フィールド名 <...n>*	◦ **RECORD** - レコード全体が出力テーブルに含まれます。 ◦ **FIELDS** - レコード全体でなく、個々のフィールドが出力テーブルに含まれます。 含めるフィールドまたは式を指定します。複数のフィールドを指定する場合、空白によってそれぞれのフィールドを区切る必要があります。
TO *テーブル名*	コマンドの結果を送信する場所: ◦ ***テーブル名*** -は、結果の保存先となる Analytics テーブルのことです。 *テーブル名*には、ファイル拡張子 .FIL を付けた文字列を引用符で囲んで指定する必要があ

名前	説明
	ります。例：TO "Output.FIL" デフォルトでは、テーブル データ ファイル(.fil) は、Analytics プロジェクトが入っているフォルダーに保存されます。 既存の異なるフォルダーにデータ ファイルを保存するには、絶対または相対ファイル パスを指定します。 • TO "C:\Output.FIL" • TO "Results\Output.FIL" **メモ** テーブル名は 64 文字の英数字(.FIL 拡張子を含まない) に制限されています。名前にはアンダースコア文字(_) を使用できますが、他の特殊文字やスペースは使用できません。名前の先頭を数字にすることはできません。
OPEN 省略可能	コマンドを実行した後、コマンドによって作成されたテーブルを開きます。コマンドが出力テーブルを作成する場合にのみ有効です。
APPEND 省略可能	コマンドの出力を既存ファイルに上書きしないで、そのファイルの末尾に追加します。 **メモ** コマンドの出力と既存のファイルの構造が同一であるようにする必要があります。 • 同じフィールド • 同じフィールド順序 • 一致するフィールドが同じ長さ • 一致するフィールドが同じデータ型 出力は、既存ファイルとの間でファイル構造が違っている場合でも、Analytics によって既存ファイルに追加されます。出力と既存のファイルの構造が一致しない場合は、データが混在、不足、不正確になります。
MERSENNE_TWISTER 省略可能	**メモ** セルおよびランダム選択方法の場合のみ。 Analytics の乱数ジェネレーターでは、メルセンヌ ツイスター アルゴリズムが使用されます。 このパラメーターを省略した場合は、Analytics のデフォルトのアルゴリズムが使用されます。 **メモ** バージョン 12 より前の Analytics で作成された Analytics スクリプトまたはサンプリングの結果との後方互換性が必要な場合は、単純にデフォルトの Analytics アルゴリズムを使用してください。
LOCAL 省略可能	Analytics プロジェクトと同じ場所に出力ファイルを保存します。 **メモ** Analytics テーブルである出力ファイルを含むサーバー テーブルに対してコマンドを実行するときにのみ適用されます。

例

レコード サンプルの抽出

レコード サンプルを使用して、請求書を含む勘定の規定された統制からの逸脱率を推定します。

統計的に有効なサンプル サイズを計算した後、サンプルを抽出する準備ができています。ランダム選択方法を使用します。

以下の例：

- 開いている Analytics テーブルからサンプルを抽出する
- シード値 123456 を指定してランダム選択方法を使用する
- サンプルサイズとして 95 レコードを指定する
- 指定したフィールドのみを出力テーブルに含める
- Analytics の乱数ジェネレーターでメルセンヌツイスター アルゴリズムを使用することを指定する

```
SAMPLE ON RECORD RANDOM 123456 NUMBER 95 FIELDS RefNum CustNum Amount Date
Type TO "Ar_record_sample" OPEN MERSENNE_TWISTER
```

備考

> **メモ**
>
> このコマンドの動作の詳細については、Analytics のヘルプを参照してください。

構文

> **メモ**
>
> フィルターリング(IF ステートメント) や範囲パラメーターを適用するとサンプルの妥当性を低下させてしまうため、それらのオプションはこの構文に含まれていません。

固定間隔選択方法

```
SAMPLE <ON> MUS( 金額単位サンプリング) 数値フィールド INTERVAL 間隔値 <FIXED 最初の値>
<CUTOFF 最上層のカットオフ値> <SUBSAMPLE> <NOREPLACEMENT> {RECORD|FIELDS
フィールド名 <...n>} TO テーブル名 <OPEN> <APPEND> <LOCAL>
```

セル選択方法

SAMPLE <ON> *MUS数値フィールド* CELL INTERVAL *間隔値* <CUTOFF *最上層のカットオフ値*> <RANDOM *シード値*> <SUBSAMPLE> <NOREPLACEMENT> {RECORD|FIELDS *フィールド名* <*...n*>} TO *テーブル名* <OPEN> <APPEND> <MERSENNE_TWISTER> <LOCAL>

ランダム選択方法

SAMPLE <ON> *MUS数値フィールド* NUMBER *サンプルサイズ* POPULATION *絶対値* <RANDOM *シード値*> <SUBSAMPLE> <NOREPLACEMENT> <ORDER> {RECORD|FIELDS *フィールド名* <*...n*>} TO *テーブル名* <OPEN> <APPEND> <MERSENNE_TWISTER> <LOCAL>

パラメーター

> **メモ**
> 値を指定する際、3 桁の区切り記号は含めないでください。

名前	説明
ON *MUS数値フィールド*	金額単位サンプリング(monetary unit sampling：MUS) を使用します。 *MUS数値フィールド*は、サンプリングのベースとして使用する数値フィールドまたは式です。
INTERVAL *間隔値* FIXED *最初の値* | CELL INTERVAL *間隔値* | NUMBER *サンプルサイズ* POPULATION *絶対値*	**INTERVAL *間隔値* FIXED *最初の値*** **固定間隔**選択方法を使用します。 最初の金額単位を選択すると、その金額単位から固定した間隔、つまり固定した距離だけ離れた金額単位が続いて選択されます。たとえば、最初に選択した金額単位の後は、5000 個おきの金額単位が選択されます。 ◦ **INTERVAL *の間隔値*** -には、サンプル サイズの計算によって出力された間隔値を指定します。 ◦ **FIXED *の最初の値*** -には、選択する最初の金額単位を指定します。 *最初の値*にゼロ('0') を指定するか、FIXED を省略した場合は、Analytics により最初の金額単位がランダムに選択されます。 **CELL INTERVAL *間隔値*** **セル**選択方法を使用します。 データ セットが複数の均等なセルまたはグループに分割され、各セルから 1 つの金額単位がランダムに選択されます。 各セルのサイズは*間隔値*によって指定します。サンプル数の計算によって生成された間隔の値を入力します。

名前	説明
	NUMBER *サンプルサイズ* POPULATION *絶対値* **ランダム**選択方法を使用します。 すべての金額単位がデータセット全体からランダムに選択されます。 ◦ **NUMBER** - の*サンプルサイズ*には、計算によって出力されたサンプル サイズを入力します。 ◦ **POPULATION *の絶対値*** -には、*MUS数値フィールド*の絶対値合計額を指定します。この合計額が、サンプルの選択元の母集団になります。
CUTOFF *最上層のカットオフ値* 省略可能	**メモ** 固定間隔選択方法およびセル選択方法の場合のみ。 最上層のカットオフ値。 カットオフ値以上の*MUS(Monetary Unit Sampling：金額単位サンプリング) 数値フィールド*の金額が自動的に選択され、サンプルに取り込まれます。 CUTOFF を省略した場合は、*間隔値*に等しいデフォルトのカットオフ値が使用されます。
RANDOM *シード値* 省略可能	**メモ** セルおよびランダム選択方法の場合のみ。 Analytics の乱数ジェネレーターを初期化するために使用するシード値。 ゼロ('0') の値を指定するか、または RANDOM を省略した場合は、シード値がランダムに選択されます。
SUBSAMPLE 省略可能	**メモ** FIELDS を指定する場合には、SUBSAMPLE のみを使用できます。 出力結果に SUBSAMPLE フィールドを追加します。 サンプル フィールドの各金額が複数の独立した取引の合計を表す場合に、サンプリングした各合計金額のうち 1 つの取引に対してのみ監査手続きを実行したいときには、SUBSAMPLE フィールドの値を使って任意に個々の取引を選択できます。 詳細については、金額単位サンプリングの実行を参照してください。
NOREPLACEMENT 省略可能	同じレコードが複数回選択されることはありません。このため、サンプルには SIZE コマンドで計算されたレコード数よりも少ないレコードが含まれるようになる可能性があります。 NOREPLACEMENT を省略した場合、または REPLACEMENT を指定した場合は、レコードを複数回選択できます。
ORDER 省略可能	**メモ** ランダム選択方法の場合のみ。 FIELDS を指定する場合には、ORDER のみを使用できます。 出力結果に ORDER フィールドを追加します。 このフィールドには、各レコードの選択順序がランダムであることが表示されます。
RECORD | FIELDS *フィールド名 <...n>*	◦ **RECORD** - レコード全体が出力テーブルに含まれます。 ◦ **FIELDS** - レコード全体でなく、個々のフィールドが出力テーブルに含まれます。

名前	説明
	含めるフィールドまたは式を指定します。複数のフィールドを指定する場合、空白によってそれぞれのフィールドを区切る必要があります。
TO *テーブル名*	コマンドの結果を送信する場所： ◦ ***テーブル名*** -は、結果の保存先となる Analytics テーブルのことです。 *テーブル名*には、ファイル拡張子 .FIL を付けた文字列を引用符で囲んで指定する必要があります。例：TO "Output.FIL" デフォルトでは、テーブル データ ファイル(.fil) は、Analytics プロジェクトが入っているフォルダーに保存されます。 既存の異なるフォルダーにデータ ファイルを保存するには、絶対または相対ファイル パスを指定します。 • TO "C:\Output.FIL" • TO "Results\Output.FIL" **メモ** テーブル名は 64 文字の英数字(.FIL 拡張子を含まない) に制限されています。名前にはアンダースコア文字(_) を使用できますが、他の特殊文字やスペースは使用できません。名前の先頭を数字にすることはできません。
OPEN 省略可能	コマンドを実行した後、コマンドによって作成されたテーブルを開きます。コマンドが出力テーブルを作成する場合にのみ有効です。
APPEND 省略可能	コマンドの出力を既存ファイルに上書きしないで、そのファイルの末尾に追加します。 **メモ** コマンドの出力と既存のファイルの構造が同一であるようにする必要があります。 • 同じフィールド • 同じフィールド順序 • 一致するフィールドが同じ長さ • 一致するフィールドが同じデータ型 出力は、既存ファイルとの間でファイル構造が違っている場合でも、Analytics によって既存ファイルに追加されます。出力と既存のファイルの構造が一致しない場合は、データが混在、不足、不正確になります。
MERSENNE_TWISTER 省略可能	**メモ** セルおよびランダム選択方法の場合のみ。 Analytics の乱数ジェネレーターでは、メルセンヌ ツイスター アルゴリズムが使用されます。 このパラメーターを省略した場合は、Analytics のデフォルトのアルゴリズムが使用されます。 **メモ** バージョン 12 より前の Analytics で作成された Analytics スクリプトまたはサンプリングの結果との後方互換性が必要な場合は、単純にデフォルトの Analytics アルゴリズムを使用してください。
LOCAL 省略可能	Analytics プロジェクトと同じ場所に出力ファイルを保存します。

名前	説明
	メモ Analytics テーブルである出力ファイルを含むサーバー テーブルに対してコマンドを実行するときにのみ適用されます。

例

金額単位サンプルの抽出

金額単位サンプリングを使用して、請求書を含む勘定の金額虚偽表示の合計金額を推定します。

統計的に有効なサンプル サイズを計算した後、サンプルを抽出する準備ができています。固定間隔選択方法を使用します。

以下の例：

- 取引金額フィールドに基づいて、開いている Analytics テーブルからサンプルを抽出する
- 間隔値 $6,283.33 で固定間隔選択方法を使用する
- 選択した最初のレコードに 100,000 ($1,000 に含まれるセント数) 番目の金額単位を指定する
- 最上層のカットオフとして $5,000 を使用する
- レコード全体を出力テーブルに含める

```
SAMPLE ON Amount INTERVAL 6283.33 FIXED 1000.00 CUTOFF 5000.00 RECORD TO "Ar_monetary_unit_sample" OPEN
```

備考

メモ

このコマンドの動作の詳細については、Analytics のヘルプを参照してください。

SAVE コマンド

Analytics テーブルのコピーを作成してそれを別の名前で保存したり、Analytics プロジェクトを保存します。

構文

Analytics テーブルのコピーを作成し、それを別の名前で保存するには

```
SAVE 新規テーブルFORMAT ACLテーブル
```

現在のプロジェクトへの変更を保存するには

```
SAVE
```

パラメーター

名前	説明
新規テーブル	作成または保存する新しい Analytics テーブルの名前。 **メモ** テーブルの名前は、64 文字までの英数字に制限されます。名前にはアンダースコア文字 (_) を使用できますが、他の特殊文字やスペースは使用できません。名前の先頭を数字にすることはできません。
FORMAT *ACLテーブル*	既存の Analytics テーブルの名前。テーブルレイアウトの名前を使用します。関連するデータファイルの名前ではありません。

例

既存のテーブルを基に新しいテーブルを作成する

下記の例では、既存のテーブル **Payables_master** に基づいて、**Payables_March** という名前の新しいテーブルを作成しています。その後で、**Payables_March** を 3 月の買掛金データファイルにリンクすることもできます。

```
SAVE Payables_March FORMAT Payables_master
```

備考

機能の仕組み

SAVE FORMAT は、**ナビゲーター**の［**総覧**］タブで Analytics テーブルをコピーして貼り付けるのと同様の結果を生成します。新しい Analytics テーブルが作成され、元のテーブルと同じデータファイルまたはデータソースに関連付けられます。

必要に応じて、新しく作成されたテーブルを別のデータソースにリンクすることができます。

SAVE を使用して入力要求を回避する

特定の時点において、現在のプロジェクトへの変更を保存するように、Analytics によって要求されます。スクリプトの実行が中断されないようにするために、Analytics から変更の保存を要求される前に SAVE コマンドを使用して変更を保存しておくことができます。

SAVE LAYOUT コマンド

Analytics テーブル レイアウトを外 部 のテーブル レイアウト ファイル(.layout) に保 存 するため、またはテーブル レイアウト のメタデータを Analytics テーブルに保 存 します。

> **メモ**
>
> バージョン 11 より前 の Analytics では、外 部 テーブル レイアウト ファイルのファイル拡 張 子 には .fmt を使 用 していました。拡 張 子 を手 動 で指 定 すれば、拡 張 子 .fmt のテーブル レイアウト ファイルを今 後 も保 存 することができます。

構 文

```
SAVE LAYOUT {FILE|TABLE} TO {ファイル名|テーブル名}
```

パラメーター

名 前	説 明
FILE | TABLE	◦ **FILE** - Analytics テーブル レイアウトを外 部 テーブル レイアウト ファイル(.layout) に保 存 します。 ◦ **TABLE** - テーブル レイアウト のメタデータを ACLテーブルに保 存 する場 合 は、TABLE を指 定 します。
TO *ファイル名* | *テーブル名*	出 力 ファイルの名 前 、出 力 場 所 : ◦ ***ファイル名*** - .layout ファイルの名 前 *ファイル名*を引 用 された文 字 列 として指 定 します。例 : TO "Ap_Trans.layout". .layout ファイル拡 張 子 はデフォルト で使 用 されるため、指 定 は省 略 可 能 です。 デフォルト では、出 力 ファイルは、Analytics プロジェクト が入 っているフォルダーに保 存 されます。 既 存 の異 なるフォルダーにファイルを保 存 するには、絶 対 または相 対 ファイル パスを指 定 します。 • TO "C:\Ap_Trans.layout" • TO "Table Layouts\Ap_Trans.layout" > **メモ** > テーブル レイアウト名 を 64 文 字 の英 数 字 (.layout 拡 張 子 を含 まない) に制 限 し、テーブル レイアウト が Analytics にインポート されるときに名 前 が切 り捨 てられないようにします。 > 名 前 にはアンダースコア文 字 (_) を使 用 できますが、他 の特 殊 文 字 やスペースは使 用 できません。名 前 の先 頭 を数 字 にすることはできません。 ◦ ***テーブル名*** -Analytics テーブルと .fil ファイルの名 前 *テーブル名*を引 用 された文 字 列 として指 定 します。例 : TO "Ap_Trans_layout_

名前	説明
	metadata.fil". .fil ファイル拡張子はデフォルトで使用されるため、指定は省略可能です。 デフォルトでは、テーブル データ ファイル(.fil) は、Analytics プロジェクトが入っているフォルダーに保存されます。 既存の異なるフォルダーにデータ ファイルを保存するには、絶対または相対ファイル パスを指定します。 • TO "C:\Ap_Trans_layout_metadata.fil" • TO "Layout Metadata\Ap_Trans_layout_metadata.fil" **メモ** テーブル名は 64 文字の英数字(.FIL 拡張子を含まない) に制限されています。名前にはアンダースコア文字(_) を使用できますが、他の特殊文字やスペースは使用できません。名前の先頭を数字にすることはできません。

例

テーブル レイアウトを外部テーブル レイアウト ファイル(.layout) に保存する

次の例では、開いているテーブルで使用されているテーブル レイアウトを、**Ap_Trans.layout** という名前の外部テーブル レイアウト ファイルに保存しています。

この場合、テーブル レイアウト ファイルは Analytics プロジェクト フォルダーに保存されます。

```
SAVE LAYOUT FILE TO Ap_Trans.layout
```

次の例では、テーブル レイアウト ファイルは指定したフォルダーに保存されます。

```
SAVE LAYOUT FILE TO "C:\ACL_DATA\AP Audit 2013\Ap_Trans.layout"
```

テーブル レイアウトのメタデータのコピーを新しい Analytics テーブルに保存する

次の例では、開いているテーブルで使用されているテーブル レイアウトのメタデータのコピーを、**Ap_Trans_layout_metadata** という名前の新しい Analytics テーブルに保存しています。

次の例では、新しい Analytics テーブルは Analytics プロジェクト フォルダーに保存されます。

```
SAVE LAYOUT TABLE TO Ap_Trans_layout_metadata
```

次の例では、Analytics テーブルを指定したフォルダーに保存しています。

```
SAVE LAYOUT TABLE TO "C:\ACL_DATA\AP Audit 2013\Ap_Trans_layout_metadata"
```

備考

SAVE LAYOUT FILE と SAVE LAYOUT TABLE の比較

SAVE LAYOUT コマンドは、次の 2 つの用途に使用されます。

- **FILE** - 開いている Analytics テーブルのテーブルレイアウトを、拡張子 `.layout` の付いた外部テーブルレイアウト ファイルに保存します。
- **TABLE** - 開いている Analytics テーブルのテーブルレイアウトからメタデータを抽出し、それを新しい Analytics テーブルに保存します。

SAVE LAYOUT FILE

機能の仕組み

SAVE LAYOUT FILE は、開いている Analytics テーブルのテーブルレイアウトを外部のテーブルレイアウト ファイルに、拡張子 .layout で保存します。

テーブルレイアウトにはメタデータが含まれており、これにより、関連付けられたソース データ ファイルの生データの構造化された解釈が提供されます。テーブルレイアウトにソース データ自体は一切含まれていません。

SAVE LAYOUT FILE(テーブル レイアウトの .layout ファイルとしての保存) の用途

テーブルレイアウトを .layout ファイルとして保存すると、テーブルレイアウトとそのメタデータが移動可能かつ再利用可能になります。

.layout ファイルは、Analytics プロジェクトにインポートして、対応するソース データ ファイルに関連付けることができます。つまり、ソース データ ファイル内のデータ要素は、テーブルレイアウトのメタデータによって規定されるフィールド定義に対応している必要があります。

たとえば、3 月と 4 月のソース データ ファイル内のデータの構造は同一であると仮定して、取引ファイルの 3 月のテーブルレイアウトを保存し、それを 4 月の取引を含んでいるソース データ ファイルに関連付けることができます。.layout ファイルをこのように使用すると、新しいテーブルレイアウトを一から作成する手間を省くことができます。

Analytics テーブルの構造の詳細については、Analytics のヘルプを参照してください。

SAVE LAYOUT TABLE

機能の仕組み

SAVE LAYOUT TABLE は、開いている Analytics テーブルのテーブルレイアウトからメタデータを抽出し、それを新しい Analytics テーブルに保存します。

新しいテーブルは、テーブルレイアウトそのものではなくむしろ標準の Analytics テーブルで、元のテーブルのテーブルレイアウト メタデータの要約を含んでいます。Analytics スクリプトでこの要約にアクセスすることで、その情報に基づいてスクリプト内で意思決定することが可能になります。

元のテーブル内の各フィールドについて、テーブル レイアウト メタデータのうち以下の部分が新しいテーブルに抽出されます。

> **メモ**
>
> 新しいテーブル内のこれらのフィールド名は、どのローカライズ版の Analytics を使用しているかに関係なく、常に英語で生成されます。

新しいテーブルのフィールド名	テーブル レイアウト メタデータ
フィールド名	フィールドの名前
データ型	フィールドのデータ型
カテゴリ	フィールドのデータ カテゴリ
開始位置	フィールドの開始位置
フィールド長	フィールドの長さ
小数位	フィールドの小数点以下の桁数(数値フィールドのみ)
書式	フィールドの書式(日付時刻フィールドと数値フィールドのみ)
代替タイトル	フィールドの代替列見出し
列幅	ビュー内での列の幅

追加の詳細

演算フィールド	抽出されたメタデータに演算フィールドは含まれますが、演算フィールドで使用される式や、条件はどれも記録されません。また、演算フィールドには、開始位置、フィールド長、および小数点以下の桁数も記録されません。
関連付けられたフィールド	関連フィールドは、テーブル レイアウトの一部ではないので含まれません。
フィールド レベルのフィルター フィールド ノート	フィールド レベルのフィルターおよびフィールド ノートは含まれません。
代替列見出し 列幅	代替列見出しおよび列の幅に記録される値は、テーブル レイアウトで指定された値です。列に対して指定できる、ビュー レベルの値ではありません。

SAVE LOG コマンド

コマンド ログ全 体 、または現 在 の Analytics セッションのログ エントリを、外 部 ファイルに保 存します。

構 文

```
SAVE LOG <SESSION> AS ファイル名 {<ASCII>|HTML} <OK>
```

パラメーター

名 前	説 明
SESSION 省 略 可 能	現 在 の Analytics セッションのログ エントリだけが保 存されます。
AS *ファイル名*	出 力ファイルの名 前 。 *ファイル名* の値 は引 用 符 で囲んだ文 字 列として指 定します。 例 : AS "コマンド ログ" ファイル拡 張 子 (.txt、.htm または .html) を指 定することはできますが、必 須 ではありません。 デフォルトでは、出 力ファイルは、Analytics プロジェクトが入っているフォルダーに保 存されます。 既 存 の異なるフォルダーにファイルを保 存 するには、絶 対または相 対ファイル パスを指 定します。 ◦ AS "C:\Command Log.TXT" ◦ AS "Results\Command Log.TXT"
ASCII | HTML	出 力結 果 の書 式 : ◦ **ASCII(またはキーワードなし)** - ASCII プレーン テキスト ファイル ◦ **HTML** - HTML ファイル
OK 省 略 可 能	*ファイル名* と同じ名 前 のファイルが既 に存 在 している場 合 でも、確 認なしで上 書きします。

例

買 掛 金 分 析 のコマンド のログを保 存 する

3 月 の買 掛 金 ファイルに対してデータ分 析 を実 行して、対 応 するコマンド のログを調 書 の一 部 として保 存したいとします。

次 の例 では現 在 の Analytics セッションの項 目 を HTML ファイルに保 存します。 同じ名 前 のファイルが存 在 している場 合 でも、確 認なしで上 書きします。

```
SAVE LOG SESSION AS "C:\Payables_March_Log.htm" HTML OK
```

SAVE TABLELIST コマンド

Analytics プロジェクトのすべてのテーブルのリストを Analytics テーブルまたは CSV ファイルに保存します。

構文

```
SAVE TABLELIST {FILE|TABLE} TO {テーブル名|ファイル名}
```

パラメーター

名前	説明
FILE | TABLE	◦ **FILE** - テーブルリストを CSV ファイル(.csv) に保存します。 ◦ **TABLE** - Analytics テーブルにテーブルリストを保存します。
TO *テーブル名* | *ファイル名*	テーブルリストを保存する場所: ◦ ***テーブル名*** - TABLE を使用するときに出力される Analytics テーブルおよびそれに関連付けられた .fil ファイルの名前 .fil ファイル拡張子はデフォルトで使用され、指定する必要はありません。このテーブルは、作業中の Analytics プロジェクトと同じフォルダーに保存され、別のフォルダーには保存できません。 **メモ** Analytics テーブルの名前は、64 文字までの英数字に制限されます。名前にはアンダースコア文字(_) を使用できますが、他の特殊文字やスペースは使用できません。名前の先頭を数字にすることはできません。 ◦ ***ファイル名*** -FILE を使用するときの .csv ファイルの名前 .csv ファイル拡張子はデフォルトで使用されるので、指定する必要はありません。絶対パスまたは相対パスを指定して、Analytics プロジェクトが入っているフォルダーとは別の既存のフォルダーに CSV ファイルを保存することができます。相対パスを指定する場合は、Analytics の作業ディレクトリに対して相対です。 スペースを含む場合は、引用された文字列として値を指定する必要があります。

例

新しいテーブルを作成する

次の例では、Analytics プロジェクトに **Table_list_complete** という新しいテーブルを作成しています。

```
SAVE TABLELIST TABLE TO Table_list_complete
```

CSV ファイルを作成する

次の例では、`C:\ACL Data` フォルダーに `Table_list_complete.csv` という新しい CSV ファイルを作成しています。

```
SAVE TABLELIST FILE TO "C:\ACL Data\Table_list_complete"
```

備考

出力の列

出力される Analytics テーブルまたは CSV ファイルには 3 つの列があります。

- **テーブル名** - Analytics テーブルレイアウトの名前。
- **タイプ** - Analytics テーブルがローカルテーブルとサーバーテーブルのどちらを表すかを示します。
- **データ ファイル パス** - ソースデータファイルへの完全パス。

SAVE WORKSPACE コマンド

ワークスペースを作成して保存します。

構文

```
SAVE WORKSPACE ワークスペース名 {フィールド名 <...n>}
```

パラメーター

名前	説明
ワークスペース名	現在の Analytics プロジェクトで作成および追加するワークスペースの名前。
フィールド名 <...n>	ワークスペースに追加するフィールドの名前。複数のフィールド名をスペースで区切って指定することができます。

例

ワークスペースのアクティブ化

Metaphor_Inventory_2002 テーブルにある 2 つの演算フィールドを用いて **Inventory_margin** というワークスペースを作成します。次に、そのワークスペースをアクティブにして、演算フィールドが **Inventory** テーブルで利用できるようにします。

```
OPEN Metaphor_Inventory_2002
SAVE WORKSPACE Inventory_margin Gross_unit_margin Percent_unit_margin
OPEN Inventory
ACTIVATE WORKSPACE Inventory_margin OK
```

備考

演算フィールドの作成に使用するフィールド名の一致要件

ワークスペースに保存する演算フィールドを作成する式の中で使用するあらゆるフィールド名は、そのワークスペースを使用するテーブルのフィールド名と一致していなければなりません。

たとえば、ワークスペース内に`Value = Sale_price * Quantity`という演算フィールドがある場合は、アクティブテーブルにも**Sale_price**フィールドと**Quantity**フィールドが存在している必要があります。

SEEK コマンド

インデックス付き文字フィールドで、指定した文字式または文字列と一致する最初の値を検索します。

構文

```
SEEK 検索式
```

パラメーター

名前	説明
検索式	検索する文字式。 任意の有効な文字式、文字変数、あるいは引用符で囲まれた文字列として指定することができます。*検索式*は、大文字と小文字が区別されます。先頭にスペースを含むことができ、文字のように扱われます。

例

文字変数と一致するフィールドの最初の値を見つける

Card_Number フィールドは文字フィールドとして定義されており、昇順でインデックス付けされています。

次の例は、*v_card_num* 変数の値と完全に一致するか、先頭が一致するフィールドの最初の値を検索します。

```
INDEX ON Card_Number TO "CardNum" OPEN
SET INDEX TO "CardNum"
SEEK v_card_num
```

文字文字列と一致するフィールドの最初の値を見つける

Card_Number フィールドは文字フィールドとして定義されており、昇順でインデックス付けされています。

次の例は、文字リテラル"AB-123" 値と完全に一致するか、先頭が一致するフィールドの最初の値を検索します。

```
INDEX ON Card_Number TO "CardNum" OPEN
SET INDEX TO "CardNum"
SEEK "AB-123"
```

備考

> **メモ**
> このコマンドの動作の詳細については、Analytics のヘルプを参照してください。

機能の仕組み

SEEK コマンドを使用すると、テーブル内で、インデックス付きの文字フィールドに指定した*検索式*を含んでいる最初のレコードに直接移動できます。

- ***検索式が見つかった場合*** -、テーブル内で最初の一致するレコードが選択されます。
- ***検索式が見つからない場合***、-"キーと一致するインデックスがありません" というメッセージが表示され、テーブルは、検索式よりも大きい値を持つ最初のレコードに位置付けられます。

  検索式よりも大きい値がインデックス付きフィールドに存在しない場合は、テーブルは最初のレコードに位置付けられます。

インデックスの必要性

SEEK を使用して、文字フィールドを検索するには、最初に昇順でフィールドにインデックスを作成する必要があります。複数の昇順の文字フィールドを基にインデックスが作成されている場合は、そのインデックスに指定されている最初のフィールドだけが検索されます。

SEEK で、文字以外のインデックスフィールドや、降順でインデックス付けされた文字フィールドを検索することはできません。

部分一致がサポートされる場合とは

部分一致がサポートされます。インデックス付きフィールドに含まれる長い値の一部を検索式に指定できるのです。ただし、検索式は、一致を成すフィールドの先頭に現れる必要があります。

SEEK コマンドは、［**正確な文字比較を行う**］オプション(SET EXACT ON/OFF) の影響を受けません。

SEQUENCE コマンド

Analytics テーブル内の 1 つ以上のフィールドが順番どおりに整列されているかどうかを確認したり、順序が正しくない項目を識別したりします。

構文

SEQUENCE <ON> {<FIELDS> *フィールド* <D> <...*n*><FIELDS> ALL} <UNFORMATTED> <ERRORLIMIT *n*> <IF *テスト*> <WHILE *テスト*> <FIRST *範囲*|NEXT *範囲*> <TO {SCREEN|*ファイル名*|PRINT}> <APPEND> <HEADER *ヘッダー テキスト*> <FOOTER *フッター テキスト*> <PRESORT> <LOCAL> <ISOLOCALE *ロケール コード*>

パラメーター

名前	説明
ON FIELDS *フィールド* D <...*n*> | FIELDS ALL	順番を検査するフィールドや式。ALL を指定すると、Analytics テーブルのすべてのフィールドがチェックされます。 キー フィールドを降順に並べ替える D を含めます。デフォルトのソート順は昇順です。
UNFORMATTED 省略可能	結果をファイルに出力する場合、ページ見出しや改ページは除去されます。
ERRORLIMIT *n* 省略可能	コマンドが停止するまでに許容されるエラー数。デフォルト値は 10 です。
IF *テスト* 省略可能	各レコードを処理するために真である必要がある条件式。コマンドは、その条件を満たすレコードに対してのみ実行されます。 **メモ** IF パラメーターは、任意の範囲パラメーター(WHILE、FIRST、NEXT) が適用された後に、テーブルに残るレコードに対してのみ評価されます。
WHILE *テスト* 省略可能	各レコードを処理するために真である必要がある条件式。条件が false と評価するか、テーブルの最後に達したら、コマンドは実行を中止します。 **メモ** WHILE を FIRST または NEXT とともに使用する場合は、1 つの制限に達するとすぐに、レコードの処理が停止します。
FIRST *範囲* | NEXT *範囲* 省略可能	処理するレコード数: ◦ **FIRST** - 指定されたレコード数に達するまで、最初のレコードから処理を開始します ◦ **NEXT** - 指定されたレコード数に達するまで、現在選択されているレコードから処理を開始し

名前	説明
	ます *範囲*は処理するレコード数を指定します。 FIRST とNEXT を省略すると、すべてのレコードがデフォルトで処理されます。
TO SCREEN | *ファイル名* | PRINT 省略可能	コマンドの結果を送信する場所： ◦ **SCREEN** - はAnalytics の表示領域に結果を表示します ◦ ***ファイル名*** -は結果の保存先となるファイルです。 *ファイル名*には、適切なファイル拡張子を付けた文字列を引用符で囲んで指定します。 例：TO "Output.TXT" デフォルトでは、テーブルファイルは、Analytics プロジェクトが入っているフォルダーに保存されます。 既存の異なるフォルダーにファイルを保存するには、絶対または相対ファイルパスを指定します。 • TO "C:\Output.TXT" • TO "Results\Output.TXT" ◦ **印刷** -- 通常使うプリンターに結果を送信します
APPEND 省略可能	コマンドの出力を既存ファイルに上書きしないで、そのファイルの末尾に追加します。 **メモ** コマンドの出力と既存のファイルの構造が同一であるようにする必要があります。 • 同じフィールド • 同じフィールド順序 • 一致するフィールドが同じ長さ • 一致するフィールドが同じデータ型 出力は、既存ファイルとの間でファイル構造が違っている場合でも、Analytics によって既存ファイルに追加されます。出力と既存のファイルの構造が一致しない場合は、データが混在、不足、不正確になります。
HEADER *ヘッダーテキスト* 省略可能	レポートの各ページの最上部に挿入されるテキスト。 *ヘッダーテキスト*は引用符で囲んだ文字列として指定する必要があります。この値は、Analytics のシステム変数であるHEADER の値よりも優先されます。
FOOTER *フッターテキスト* 省略可能	レポートの各ページの最下部に挿入されるテキスト。 *フッターテキスト*は引用符で囲んだ文字列として指定する必要があります。この値は、Analytics のシステム変数であるFOOTER の値よりも優先されます。
PRESORT 省略可能	コマンドを実行する前にキーフィールドでテーブルを並べ替えます。 **メモ** GROUP コマンドの内部ではPRESORT を使用することができません。
LOCAL 省略可能	Analytics プロジェクトと同じ場所に出力ファイルを保存します。 **メモ** Analytics テーブルである出力ファイルを含むサーバーテーブルに対してコマンドを実行するときにのみ適用されます。

名前	説明
ISOLOCALE *ロケールコード* 省略可能	**メモ** Analytics の Unicode 版にのみ適用されます。 システム ロケールは「*言語-国*」の形式で入力します。たとえば、カナダ フランス語はコード「fr_ca」を入力します。 次のコードを使用します。 ◦ **言語** - ISO 639 標準言語コード ◦ **国** - ISO 3166 標準国コード 国コードを指定しない場合は、言語のデフォルト国が使用されます。 ISOLOCALE を使用しない場合は、デフォルト システム ロケールが使用されます。

Analytics の出力変数

名前	含む
WRITE*n*	コマンドによって検出された順番検査エラーの合計数。

例

従業員 ID と採用日を順番検査する

次の例は、**EmployeeID** と **HireDate** のフィールドから検出されたすべての順番検査エラーをテキスト ファイルに書き込んでいます。

```
SEQUENCE ON EmployeeID HireDate ERRORLIMIT 10 TO "SequenceErrors.txt"
```

備考

SEQUENCE を GROUP 内で使用する

SEQUENCE コマンドは、GROUP コマンド内で実行しても、グループの処理に影響を及ぼさないようになっています。ただし、GROUP コマンド以降のデータ順番検査エラーを報告することはありません。

SET コマンド

構成可能な Analytics オプションを設定します。

> **メモ**
>
> SET コマンドは、Analytics セッション中のみの Analytics オプションを設定します。この動作は、SET コマンドの使用場所が Analytics コマンド ラインであろうと Analytics スクリプトであろうと適用されます。
>
> Analytics オプションが Analytics セッション間で存続されるようにするには、［**オプション**］ダイアログ ボックスを使用する必要があります。詳細については、ACL オプションの構成を参照してください。

構文

構文	例および備考
SET BEEP *値*	SET BEEP 2 コマンド処理が完了したときにビープ音を鳴らす回数を指定します。 *値* パラメーターは 1 から 255 の間である必要があります。
SET CENTURY *値*	SET CENTURY 40 世紀を解釈する開始年を 2 桁の年で指定します。 *値* パラメーターは 0 から 99 までである必要があります。 世紀解釈の開始年の値を 40 に設定すると、2 桁の年の 40 から 99 までは 1940 年から 1999 年と解釈され、2 桁の年の 00 から 39 までは 2000 年から 2039 年として解釈されます。
SET CLEAN {ON|OFF}	SET CLEAN ON このオプションをオンにすると、Analytics は無効な文字データは空白に、無効な数値データはゼロに置き換えます。
SET DATE <TO> {0|1|2|*文字列*}	SET DATE "YYYY/MM/DD" Analytics での、ビュー、レポート、およびエクスポート ファイルにおける日付、日付時刻の日付部分の表示方法を指定します。 ◦ SET DATE 0 は、日付を MM/DD/YYYY 書式に設定します ◦ SET DATE 1 は、日付を MM/DD/YY 書式に設定します ◦ SET DATE 2 は、日付を DD/MM/YY 書式に設定します

構文	例および備考
	◦ SET DATE "<*文字列*>" は、日付を独自に指定するカスタム書式に設定します [**オプション**]ダイアログ ボックスで日、月、年にそれぞれ 'D'、'M'、'Y' 以外を指定している場合でも、SET DATE コマンドを使ってカスタム日付書式を指定するときには、日、月、年に 'D'、'M'、'Y' を使用する必要があります。例: `SET DATE "DD MMM YYYY"`
SET DELETE_FILE {ON|OFF}	`SET DELETE_FILE ON` デフォルト設定: OFF オンにすると、テーブル レイアウトを削除したときに、関連付けられているデータ ファイルが自動的に削除されるようになります。 オフにすると、テーブル レイアウトを削除したときに、関連付けられているデータ ファイルが自動的に削除されるようになります。 アンダースコア(_) を DELETE_FILE に含める必要があります。 コマンド ラインで、SET DELETE_FILE をパラメーターなしで指定すると、DELETE_FILE が現在オンであるかオフであるかが表示されます。 **注意** このオプションをオンにするときには注意してください。テーブルと共に元のデータ ファイルが削除される場合もあります。 データ ファイルはすぐに削除されます。Windows のごみ箱には送られません。
SET DESIGNATION *値*	`SET DESIGNATION "Produced by ABC Corporation"` *値* パラメーターは、印刷された各ページの一番上に表示されるラベルを指定する、引用符で囲まれた文字列です。
SET ECHO {ON|NONE}	`SET ECHO NONE` `COM スクリプト内のコマンドと結果は、ログから除外されます。` `SET ECHO ON` スクリプト内のコマンドと結果を Analytics コマンド ログに書き込むのを停止するには、NONE を指定します。ON を指定すると、ログ記録が再開します。 SET ECHO コマンドは、スクリプト内のコマンドと結果のログ記録に対してのみ適用されます。ユーザー インターフェイスを用いて実行されたコマンドや、コマンド ラインから発行されたコマンド、およびそれらが生成するあらゆる結果は、ECHO が設定されている方法にかかわらず、常にログに記録されます。 SET ECHO NONE/ON コマンドは、スクリプト内でもコマンド ラインからでも発行できますが、コマンドの発行場所に関係なく、それはスクリプト内のコマンドと結果のログ記録にのみ影響します。 コマンド ラインで、SET ECHO をパラメーターなしで指定すると、スクリプト内のコマンドと結果のログ記録が現在オンであるかオフであるかが表示されます。

<table>
<tr><th>構文</th><th>例および備考</th></tr>
<tr><td>SET EXACT {ON|OFF}</td><td>
<pre>SET EXACT ON</pre>
デフォルト設定：OFF
文字型フィールド、識別子、リテラル値を Analytics で比較する方法を制御できます。
メモ
スペースは文字のように扱われます。
◦ SET EXACT OFF - 異なる長さの 2 つの文字列を比較した場合、Analytics により、短い方の文字列が採用されます。比較は左から右へ行われます。
たとえば、"AB" は、"AB" と一致するほか、"ABC" とも一致すると見なされます。
◦ SET EXACT ON - 一致を成すには、比較文字列同士が全く同じでなければなりません。異なる長さの 2 つの文字列を比較する場合、Analytics により、長い方の文字列の長さに一致するよう、短い方の文字列の末尾にスペースが埋め込まれます。
たとえば、"AB" は "AB" と等しいですが、"ABC" と等しいとは見なされません。
SET EXACT のその他の例については、[テーブル]タブ([オプション]ダイアログ ボックス内) の[正確な文字比較を行う]オプションのヘルプを参照してください。
先頭および末尾のスペースを削除することで、文字列と文字列内部のスペースのみが比較されるようにするには、ALLTRIM( ) 関数を使用します。
たとえば、値に対して ALLTRIM( ) を使用した ALLTRIM(" AB") = ALLTRIM("AB") は True になりますが、この関数を使用しない場合の比較の結果は False になります。
Analytics のコマンドや関数は、SET EXACT の影響を受けるものもあれば、受けないものもあります。
<table>
<tr><th>影響を受ける</th><th>影響を受けない</th></tr>
<tr><td>◦ LOCATE コマンド
◦ MATCH( ) 関数
◦ BETWEEN( ) 関数</td><td>◦ JOIN コマンド
◦ DEFINE RELATION コマンド
◦ FIND( ) 関数
◦ FINDMULTI( ) 関数</td></tr>
</table>
</td></tr>
<tr><td>SET FILTER <TO> {テスト|フィルター名}</td><td>
<pre>SET FILTER TO ProdNo = "070104347"</pre>
<pre>SET FILTER TO ProdNoFilter</pre>
開いているテーブルでグローバル フィルター(ビュー フィルター) を作成し、論理テストを指定するか、または保存されている既存のフィルターの名前を指定します。
SET FILTER をパラメーターなしで指定し、開いているファイルからフィルターを削除します。
</td></tr>
<tr><td>SET FOLDER フォルダーパス</td><td>コマンド出力のために、 [総覧]タブ内の Analytics プロジェクト フォルダーを指定します。デフォルトの出力フォルダーは、アクティブなテーブルを含んでいるフォルダーです。
これは DOS 形式のパスで、「/フォルダー名/サブフォルダー名」の形式を用います。最初のスラッシュ(/) は [総覧]タブのルート階層を表します。ファイルの絶対パスを指定する必要があります。</td></tr>
</table>

構文	例および備考
	◦ SET FOLDER /Tables/Results は、出力フォルダーを Results サブフォルダーに設定します。Results サブフォルダーが存在しなければ、作成されます。 ◦ SET FOLDER / は、出力フォルダーを [**総覧**] タブのルート階層に設定します。 ◦ SET FOLDER は、出力フォルダーをデフォルト設定 (アクティブなテーブルを含んでいるフォルダー) にします。 出力フォルダーは、リセットするか、プロジェクトを閉じない限り、設定したままの状態であり続けます。プロジェクトを開くときに、出力フォルダーはデフォルトのアクティブなテーブル フォルダーに戻ります。
SET FORMAT {ON|OFF}	SET FORMAT ON デフォルト設定 : OFF ON パラメーターを使用すると、新しいテーブルを開いたときに、Analytics は現在のテーブル レイアウトと演算フィールド定義を自動的に表示します。この表示内容はコマンド ログにも出力されます。
SET FUZZYGROUPSIZE <TO> *数*	SET FUZZYGROUPSIZE TO 10 出力結果のあいまい重複グループに表示できる項目の最大数を指定します。*数* パラメーターは 2 未満にすることや、100 を超えることはできません。デフォルト サイズは 20 です。指定されたサイズは、Analytics セッションの間は有効となります。
SET GRAPH *グラフの種類*	SET GRAPH LINE 以降に生成されるすべてのグラフで使用するグラフの種類を指定します。コマンドの実行は、指定されたグラフの種類と互換性がある必要があります。たとえば、BENFORD コマンドで PIE2D または PIE3D チャートを作成することはできません。互換性がないグラフの種類が指定された場合は、デフォルトのグラフの種類が使用されます(BAR3D) 。 *グラフの種類* パラメーターは下記のうちの 1 つである必要があります。 ◦ PIE2D ◦ PIE3D ◦ BAR2D ◦ BAR3D - これがデフォルトのグラフの種類です。 ◦ STACKED2D ◦ STACKED3D ◦ LAYERED ◦ LINE ◦ BENFORD - 2D 棒グラフと 2D 折れ線グラフを組み合わせます。
SET HISTORY <TO> *値*	SET HISTORY TO 50

構文	例および備考
	保持するテーブル履歴エントリの最大数を指定します。*値*パラメーターは 1 ～ 100 である必要があります。
SET INDEX <TO> *値*	SET INDEX TO "CustomerCode.INX" アクティブなテーブルに適用するインデックスを指定します。
SET LEARN <TO> *スクリプト*	SET LEARN TO InventoryRec **スクリプト レコーダー**でコマンドを記録するために使用されるスクリプト ファイルの名前を指定します。
SET LOG <TO> {*ファイル*|OFF}	SET LOG TO "analysis.log" SET LOG OFF 最初のコマンドはログインを指定したログに切り替えます。指定したログが存在していない場合は、作成されます。 2 番目のコマンドはログを元の Analytics コマンド ログに復元します。 **メモ** Analytics プロジェクト パスとログ名の最大長は 259 文字です。これには、ファイル パス、ログ名、ファイル拡張子(.log) が含まれます。
SET LOOP <TO> *数*	SET LOOP TO 20 LOOP コマンドで実行されるコマンドの最大数を指定します。指定数のコマンドが実行されると、このコマンドは終了します。 *数*の範囲は 0 ～ 32767 です。0 でループ テストを停止します。
SET MARGIN *側面* <TO> *値*	SET MARGIN TOP TO 100 *側面*パラメーターには LEFT、RIGHT、TOP、または BOTTOM を指定します。四方の余白を変更する場合は、個別の SET MARGIN コマンドで各余白を指定する必要があります。*値*に 100 を指定すると、1 インチの余白が作られます。
SET MATH <TO> {FIRST|LAST|MIN|MAX}	SET MATH TO MIN デフォルト設定 : MAX 2 つのオペランドが数式で評価されるときに小数点精度が動作する方法を指定します。 ◦ **FIRST** - オペランドのペアの最初のオペランドの小数点桁数を使用します ◦ **LAST** - オペランドのペアの最後のオペランドの小数点桁数を使用します

構文	例および備考
	◦ **MIN** - オペランドのペアで最も少ない小数点桁数を使用します ◦ **MAX** - オペランドのペアで最も多い小数点桁数を使用します 複数のオペランドの式では、SET MATH 設定はペアで動作します。標準の演算順序 (BOMDAS) で評価されるときに、指定された設定を各オペランドのペアに適用し、必要に応じて端数処理します。 SET MATH 設定で、結果の小数点桁数が減る場合、結果は切り捨てられず、端数処理されます。 詳細については、数式の端数処理と小数点精度の制御を参照してください。 **メモ** Analytics テーブルが開いている間には、SET MATH を使用できません。
SET MONTHS <TO> *文字列*	3 文字形式のデフォルトの月名を指定します。*文字列*パラメーターは、カンマで区切られた月名の省略形のリストです。
SET NOTIFYFAILSTOP {ON|OFF}	SET NOTIFYFAILSTOP ON デフォルト設定 : OFF ◦ **NOTIFYFAILSTOP が OFF** - の場合 スクリプト内の NOTIFY コマンドが失敗しても、スクリプトの続行が Analytics によって許可されます。 ◦ **NOTIFYFAILSTOP が ON** - の場合 スクリプト内の NOTIFY コマンドが失敗すると、Analytics により、スクリプトの処理が停止されると共にメッセージがログに書き込まれます。スクリプトは、最初の失敗後に停止するか、または NOTIFYRETRYATTEMPTS に指定された試行回数後、どの試行も成功しなかった場合に停止します。
SET NOTIFYRETRYATTEMPTS <TO> *数*	SET NOTIFYRETRYATTEMPTS TO 10 NOTIFY コマンドが最初の試行に失敗した後、電子メールの送信を試行する回数を指定します。0 から 255 までの数値を入力します。0 を入力すると、最初の失敗後に追加の試行は行われません。デフォルトは 5 です。 NOTIFY コマンドが電子メールの送信に失敗することについて考えられる理由の 1 つは、電子メール サーバーを利用できないということです。
SET NOTIFYRETRYINTERVAL <TO> *秒数*	SET NOTIFYRETRYINTERVAL TO 30 NOTIFYRETRYATTEMPTS 間の秒数を指定します。1 から 255 までの数値を入力します。デフォルトは 10 秒です。
SET ORDER <TO> *値*	文字フィールドの並べ替え順を指定します。*値*パラメーターは、選択した並べ替え順にすべての文字を列挙します。
SET OVERFLOW {ON|OFF}	SET OVERFLOW OFF デフォルト設定 : ON

構文	例および備考
	OFF が指定されている場合、Analytics はオーバーフロー エラーが発生しても処理を停止しません。
SET PASSWORD *番号* <TO> *文字列*	SET PASSWORD 1 TO "password123" パスワード定義を作成するとともに、スクリプトの無人実行用にパスワード値を指定するために使用します。 *番号*パラメーターはパスワード定義を一意に識別するものであり、1 から 10 までの値である必要があります。パスワード値は、引用符で囲んだ値として指定します。
SET PERIODS <TO> *値* <,...*n*>	SET PERIODS TO "0,30,90,180,10000" AGE コマンドによって使用される年齢調べ間隔のデフォルトを指定します。
SET PICTURE *書式*	SET PICTURE "(9,999,999.99)" 数値のデフォルト書式を指定します。
SET READAHEAD <TO> *サイズ*	読み取るデータ ブロックのサイズを指定します。この設定の変更は、サポートからアドバイスされていない限り、行わないでください。
SET RETRY <TO> *数* SET RETRYIMPORT <TO> *num*	SET RETRY TO 50 最初の試行が失敗した場合、Analytics がデータのインポートまたはエクスポートを試行する回数を指定します。0 から 255 までの数値を入力します。0 を入力すると、最初の失敗後に追加の試行は行われません。デフォルトは 0 です。 再試行の間の待ち時間はありません。各連続した試行は、直前の失敗直後に行われます。 再試行を指定する機能は、データベースまたはクラウド データ サービスに接続する場合に便利ですが、一時的に利用できない可能性があります。 以下のコマンドに適用されます: ◦ ACCESSDATA ◦ IMPORT GRCPROJECT ◦ IMPORT GRCRESULTS ◦ IMPORT SAP ◦ RETRIEVE ◦ REFRESH (ACCESSDATA または IMPORT SAP を使って最初に作成されたテーブルの場合のみ) ◦ EXPORT . . . ACLGRC (HighBond のリザルトへのエクスポート)

構文	例および備考
	メモ SET RETRYIMPORT は旧バージョンとの互換性を保つために引き続きサポートされます。SET RETRYIMPORT と SET RETRY では、同じアクションが実行されます。
SET SAFETY {ON|OFF}	SET SAFETY OFF 次のいずれかを上書きするときに確認のダイアログ ボックスを表示するには、ON を指定します。 ◦ テーブル レイアウトにおけるフィールド ◦ Analytics テーブル ◦ Analytics データ ファイル(.fil) などのファイル ダイアログ ボックスが表示されないようにするには、OFF を指定します。 コマンド ラインで、SET SAFETY をパラメーターなしで指定すると、SAFETY が現在オンであるかオフであるかが表示されます。
SET SEPARATORS <TO> *値*	SET SEPARATORS TO ".,," Analytics で使用されるデフォルトの小数点の記号、桁区切り文字、およびリストの区切り文字を指定します。SET SEPARATORS の値は、3 つの有効な区切り文字を次の順序で指定する必要があります。 ◦ 小数点の記号(ピリオド、カンマ、またはスペース) ◦ 桁区切り記号(ピリオド、カンマ、またはスペース) ◦ リストの区切り文字(セミコロン、カンマ、またはスペース) 3 つの区切り文字のうち、小数点の記号は一意である必要があります。このコマンドを使用する場合は、3 つの区切り文字をすべて指定してください。リストの区切り文字は主に、関数のパラメーターを区切るために使用されます。
SET SESSION <*セッション名*>	SET SESSION SET SESSION "Analysis" Analytics コマンド ログ内に新しいセッションを出力します。セッションは、現在の自国スタンプによって識別されます。 オプションの *セッション名* では、30文字までの追加識別情報を追加できます。引用符は許可されていますが、必要ありません。
SET SORTMEMORY *数字*	SET SORTMEMORY 800 ソート処理とインデックス処理に割り当てるメモリの最大量を指定します。 *数字*パラメーターは 0 から 2000 メガバイト(MB) で、20 MB 単位で入力します。ソート メモリに 0 を設定した場

構文	例および備考
	合、Analytics により、現在空いているメモリが使用されます。
SET SUPPRESSTIME {ON|OFF}	SET SUPPRESSTIME ON デフォルト設定：OFF ODBC データソースを使用する(IMPORT ODBC コマンド) 、またはデータベースに直接アクセスする(DEFINE TABLE DB コマンド) Analytics テーブルを定義する場合にのみ使用します。 ON パラメーターを使用している場合にテーブルを定義すると、Analytics によって日付時刻値の時刻部分が出力されなくなります。たとえば、20141231 235959 を読み取って、ビューに表示したら、その後は 20141231 として処理されます。 以前の Analytics スクリプト(v.10.0 より前) の日付時刻では、日付時刻データの時刻部分は切り捨てられることが前提となっていましたが、このコマンドを含めると、日付時刻型対応バージョンの Analytics でそのスクリプトを実行できるようになります。 Analytics は、日付時刻書式の日付部分のみを使用するので、時刻部分を出力しません。時刻データは依然として、.fil ファイルやデータベーステーブルに存在します。必要であれば、データの時刻部分を含めるように、フィールドを定義し直したり、新しいフィールドを定義したりすることができます。 SET SUPPRESSTIME = OFF とした場合、ODBC またはデータベースへの直接アクセスを使用して定義された Analytics テーブルには、完全な日付時刻値が含まれます。 SET SUPPRESSTIME ON/OFF コマンドは、スクリプト内またはコマンド ラインから発行できます。 コマンド ラインで、SET SUPPRESSTIME をパラメーターなしで指定すると、日付時刻データの時刻部分の出力抑制が現在オンであるかオフであるかが表示されます。
SET SUPPRESSXML {ON|OFF}	SET SUPPRESSXML ON デフォルト設定：OFF コマンド出力は書式設定されたテキストではなくプレーン テキストであることを指定します。
SET TEST {ON|OFF}	SET TEST ON GROUP コマンドに関連する IF、WHILE、FOR、および NEXT の各テストの結果をログに記録するかどうかを指定します。
SET TIME <TO> *文字列*	SET TIME "hh:mm:ss PM" ビュー、レポート、およびエクスポート ファイルにおける日付時刻の時刻部分、または時刻値のみを Analytics が表示する方法を指定します。 [**オプション**]ダイアログ ボックスで別の時刻書式文字を指定した場合でも、SET TIME コマンドを使用して時刻書式を指定するときは、時間には 'h'、分には 'm'、秒には 's' を使用する必要があります。例：

構文	例および備考
	SET TIME TO "hh:mm"
SET UTCZONE {ON|OFF}	SET UTCZONE OFF デフォルト設定：ON ◦ **UTCZONE が ON の場合** - Analytics により、時刻の表示が、UTC オフセット付きのローカル時間から、そのローカル時間に相当する UTC に変更されます (UTC は協定世界時で、経度 0 度地点における時刻) 。 ◦ **UTCZONE が OFF の場合** - Analytics により、UTC オフセット付きのローカル時間が表示され、UTC には変更されません。 例： ◦ 01 Jan 2015 04:59:59 (SET UTCZONE ON) ◦ 31 Dec 2014 23:59:59-05:00 (SET UTCZONE OFF) ローカル時間の UTC への変換は、表示目的のためだけであり、ソースデータには影響しません。この 2 つの異なる表示モードは、いつでも変更して切り替えることができます。
SET VERIFY {ON|OFF|BLANK}	SET VERIFY ON ON が指定されている場合、Analytics はテーブルが開かれるたびに、データフィールドの内容がテーブルレイアウトのフィールドのデータ型に対応しているかどうかを自動的にチェックします。BLANK が指定されている場合、Analytics は ON パラメータで説明した検証に加えて、無効な文字データを空白に、無効な数値データをゼロに置き換えます。
SET WIDTH <TO> *文字*	SET WIDTH TO 20 数値演算フィールドや一時的に作成した数式の表示幅のデフォルト値 (文字) を指定します。Analytics は最大幅を指定することはできません。

SIZE コマンド

金額単位のサンプリングまたはレコードのサンプリングに対し、統計的に有効なサンプル サイズおよびサンプル間隔を計算します。

レコードのサンプリング 金額単位のサンプリング

構文

SIZE RECORD CONFIDENCE *信頼レベル*POPULATION *母集団の大きさ*PRECISION *許容逸脱率* <ERRORLIMIT *予想誤謬率*> <TO {SCREEN|*ファイル名*>

パラメーター

> **メモ**
> 値を指定する際、3 桁の区切り記号やパーセント記号は含めないでください。

名前	説明
RECORD	レコード サンプルのサンプル サイズを計算する ATTRIBUTE は、RECORD と同じ操作を行う、廃止されたパラメーターです。
CONFIDENCE *信頼度*	必要な信頼度。この信頼度で、結果のサンプルが母集団全体を表します。 たとえば、95 を指定した場合は、サンプルが実際に 95% の確率で母集団を代表しているとお客様が信頼したいということを意味します。信頼度は "サンプリング リスク" の補数です。信頼度が 95% ということはサンプリング リスクが 5% ということと同じです。
POPULATION *母集団の大きさ*	テーブルからサンプリングするレコード数
PRECISION *許容逸脱率*	許容逸脱率。許容逸脱率とは、発生し得るが、発生した場合でも統制が有効であると見なすことのできる、規定した統制からの逸脱率の上限のことです。 たとえば、5 を指定するとは、逸脱率が 5% を超えたときに統制を無効と見なすことができるということです。
ERRORLIMIT *予想誤謬率* 省略可能	母集団における推定逸脱率。検出する規定の統制からの逸脱率です。 たとえば、1 を指定するとは、この逸脱率が 1% 以内であることを指定することです。 このパラメーターを省略した場合は、母集団における推定逸脱率として 0% が使用されます。
TO SCREEN | *ファイル名*	コマンドの結果を送信する場所： ◦ **SCREEN** - は Analytics の表示領域に結果を表示します ◦ ***ファイル名*** -は結果の保存先となるファイルです。

名前	説明
	*ファイル名*には、適切なファイル拡張子を付けた文字列を引用符で囲んで指定します。 例：TO "Output.TXT" デフォルトでは、テーブル ファイルは、Analytics プロジェクトが入っているフォルダーに保存されます。 既存の異なるフォルダーにファイルを保存するには、絶対または相対ファイル パスを指定します。 • TO "C:\Output.TXT" • TO "Results\Output.TXT"

Analytics の出力変数

名前	含む
SAMPINT*n*	コマンドによって計算された、必須のサンプル間隔。
SAMPSIZE*n*	コマンドによって計算された、必須のサンプル サイズ。

例

レコード サンプルのサンプル サイズおよび間隔を計算する

レコード サンプルを使用して、請求書を含む勘定の規定された統制からの逸脱率を推定することを決定しました。

サンプルを抽出する前に、各層の統計的に有効なサンプル サイズを計算します。

Analytics によって抽出されるサンプルの 95% の時間が全体として母集団を表す信頼度が必要です。

指定された信頼度を使用し、下記の例では、レコード サンプルを抽出する場合に使用するためのサンプル サイズおよびサンプル間隔値がそれぞれ 95 および 8.12 として計算されます。

```
SIZE RECORD CONFIDENCE 95 POPULATION 40000 PRECISION 5 ERRORLIMIT 2
```

備考

> **メモ**
>
> このコマンドの動作の詳細については、Analytics のヘルプを参照してください。

構文

SIZE MONETARY CONFIDENCE *信頼度* POPULATION *母集団の大きさ* MATERIALITY *許容虚偽表示* <ERRORLIMIT *推定虚偽表示額*> <TO {SCREEN|*ファイル名*>

パラメーター

> **メモ**
> 値を指定する際、3 桁の区切り記号やパーセント記号は含めないでください。

名前	説明
MONETARY	金額単位サンプルのサンプル サイズを計算する
CONFIDENCE *信頼度*	必要な信頼度。この信頼度で、結果のサンプルが母集団全体を表します。 たとえば、95 を指定した場合は、サンプルが実際に 95% の確率で母集団を代表しているとお客様が信頼したいということを意味します。信頼度は "サンプリング リスク" の補数です。信頼度が 95% ということはサンプリング リスクが 5% ということと同じです。
POPULATION *母集団の大きさ*	数値サンプル フィールドの合計絶対値
MATERIALITY *許容虚偽表示*	許容虚偽表示。許容虚偽表示とは、サンプル フィールドの値として許容する最大の虚偽表示合計金額のことです。重大な虚偽表示額とまでは見なされません。 たとえば、29000 を指定するとは、虚偽表示合計金額が $29,000 を上回る場合に重大な虚偽表示額と見なすということです。
ERRORLIMIT *推定虚偽表示額* 省略可能	推定虚偽表示額。サンプル フィールドの値として許容する最大の虚偽表示合計金額のことです。 たとえば、5800 を指定するとは、虚偽表示合計金額として $5,800 までを許容するということです。 このパラメーターを省略した場合は、推定予想虚偽表示額として $0.00 が使用されます。
TO SCREEN | *ファイル名*	コマンドの結果を送信する場所： ◦ **SCREEN** - は Analytics の表示領域に結果を表示します ◦ ***ファイル名*** -は結果の保存先となるファイルです。 *ファイル名*には、適切なファイル拡張子を付けた文字列を引用符で囲んで指定します。 例：TO "Output.TXT" デフォルトでは、テーブル ファイルは、Analytics プロジェクトが入っているフォルダーに保存されます。 既存の異なるフォルダーにファイルを保存するには、絶対または相対ファイル パスを指定します。

名前	説明
	• TO "C:\Output.TXT" • TO "Results\Output.TXT"

Analytics の出力変数

名前	含む
SAMPINT*n*	コマンドによって計算された、必須のサンプル間隔。
SAMPSIZE*n*	コマンドによって計算された、必須のサンプル サイズ。

例

金額単位サンプルのサンプル サイズおよび間隔を計算する

金額単位サンプリングを使用して、請求書を含む勘定の金額虚偽表示の合計金額を推定することを決定しました。

サンプルを抽出する前に、各層の統計的に有効なサンプル サイズを計算します。

Analytics によって抽出されるサンプルの 95% の時間が全体として母集団を表す信頼度が必要です。

指定された信頼度を使用し、下記の例では、金額単位サンプルを抽出する場合に使用するためのサンプル サイズおよびサンプル間隔値がそれぞれ 93 および 6,283.33 として計算されます。

```
SIZE MONETARY CONFIDENCE 95 POPULATION 585674.41 MATERIALITY 29000 ERRORLIMIT 5800 TO SCREEN
```

備考

> **メモ**
>
> このコマンドの動作の詳細については、Analytics のヘルプを参照してください。

SORT コマンド

Analytics テーブルのレコードを、指定されたキーに基づいて昇順または降順に並べ替えます。結果は新しい物理的に並べ替えられた Analytics テーブルに出力されます。

構文

SORT ON {*キーフィールド* <D> <...*n*>|ALL} <FIELDS *フィールド名* <AS *表示名*> <...*n*>|FIELDS ALL> TO *テーブル名* <IF *テスト*> <WHILE *テスト*> <FIRST *範囲*|NEXT *範囲*> <APPEND> <OPEN> <ISOLOCALE *ロケールコード*>

パラメーター

名前	説明
ON *キーフィールド* D <...*n*> | ALL	並べ替えで使用するキーフィールドまたはフィールド、または式。 演算フィールドや一時的に作成した式など、データ型に関係なく、あらゆる種類のフィールドを並べ替えることができます。 ◦ ***キーフィールド*** -では、指定した 1 つまたは複数のフィールドが使用されます。 複数のフィールドで並べ替えを行う場合は、出力テーブルは入れ子で並べ替えられます。入れ子でのフィールド間の順序は、フィールドを指定した順になります。 キーフィールドを降順に並べ替える D を含めます。デフォルトのソート順は昇順です。 ◦ **ALL** - では、テーブル内のすべてのフィールドが使用されます。 テーブル内のすべてのフィールドを基準にして並べ替えを行うと、出力テーブルが入れ子で並べ替えられます。入れ子でのフィールド間の順序は、フィールドを指定した順になります。 ALL では昇順でしか並べ替えることができません。
FIELDS *フィールド名* <...*n*> | FIELDS ALL 省略可能	**メモ** キーフィールドは、自動的に出力テーブルに追加されるため、FIELDS に指定する必要はありません。 出力に含めるフィールド： ◦ **FIELDS *フィールド名*** - 指定されたフィールドを使用します フィールドは一覧の順序で使用されます。 演算フィールドを出力先テーブル内の適切なデータ型(ASCII 型または Unicode 型(Analytics のエディションによる)、ACL 型(ネイティブの数値データ型)、日付時刻型、あるいは論理型) の物理フィールドに変換します。演算された実際の値を物理フィールドに設定します。 ◦ **FIELDS ALL** - テーブルのすべてのフィールドを使用します。 フィールドは、テーブルレイアウトに現れる順序と同じ並びで使用されます。

名前	説明
	演算フィールドを出力先テーブル内の適切なデータ型(ASCII 型または Unicode 型(Analytics のエディションによる) 、ACL 型(ネイティブの数値データ型) 、日付時刻型、あるいは論理型) の物理フィールドに変換します。演算された実際の値を物理フィールドに設定します。 ◦ **FIELDS を省略する場合** - レコード全体、つまりすべてのフィールドとレコードの未定義部分すべてが、並べ替え後の出力テーブルに追加されます。 フィールドは、テーブル レイアウトに定義されているとおりの順に使用されます。 演算フィールドは保持されます。 **ヒント** レコードに含まれるデータの一部のみが必要な場合は、並べ替え後の出力テーブルにすべてのフィールド、つまりレコード全体が含まれないようにしてください。必要なフィールドのみを選択します。多くの場合、これにより、並べ替え処理が高速化します。
AS *表示名* 省略可能	FIELDS を指定した並べ替えを行う場合にのみ使用されます。 新しい Analytics テーブルのビューにおけるフィールドの表示名(代替列見出し) 。表示名をフィールド名、またはソース テーブル内の既存の表示名と同じにしたい場合は、AS を使用しないでください。 *表示名*の値は引用符で囲まれた文字列。列見出しを改行したい場合は、語句の間にセミコロン(;) を入れます。 **メモ** AS は、新しいテーブルへの出力時にのみ使用します。既存のテーブルに追加している場合は、既存テーブルの代替列見出しが優先されます。
TO *テーブル名*	コマンドの結果を送信する場所: ◦ ***テーブル名*** -は、結果の保存先となる Analytics テーブルのことです。 *テーブル名*には、ファイル拡張子 .FIL を付けた文字列を引用符で囲んで指定する必要があります。例: `TO "Output.FIL"` デフォルトでは、テーブル データ ファイル(.fil) は、Analytics プロジェクトが入っているフォルダーに保存されます。 既存の異なるフォルダーにデータ ファイルを保存するには、絶対または相対ファイル パスを指定します。 • `TO "C:\Output.FIL"` • `TO "Results\Output.FIL"` **メモ** テーブル名は 64 文字の英数字(.FIL 拡張子を含まない) に制限されています。名前にはアンダースコア文字(_) を使用できますが、他の特殊文字やスペースは使用できません。名前の先頭を数字にすることはできません。
IF *テスト* 省略可能	各レコードを処理するために真である必要がある条件式。コマンドは、その条件を満たすレコードに対してのみ実行されます。 **メモ** IF パラメーターは、任意の範囲パラメーター(WHILE、FIRST、NEXT) が適用された後に、テーブルに残るレコードに対してのみ評価されます。

名前	説明
WHILE *テスト* 省略可能	各レコードを処理するために真である必要がある条件式。条件がfalseと評価するか、テーブルの最後に達したら、コマンドは実行を中止します。 **メモ** WHILEをFIRSTまたはNEXTとともに使用する場合は、1つの制限に達するとすぐに、レコードの処理が停止します。
FIRST *範囲* | NEXT *範囲* 省略可能	処理するレコード数： ◦ **FIRST** - 指定されたレコード数に達するまで、最初のレコードから処理を開始します ◦ **NEXT** - 指定されたレコード数に達するまで、現在選択されているレコードから処理を開始します *範囲*は処理するレコード数を指定します。 FIRSTとNEXTを省略すると、すべてのレコードがデフォルトで処理されます。
APPEND 省略可能	コマンドの出力を既存ファイルに上書きしないで、そのファイルの末尾に追加します。 **メモ** コマンドの出力と既存のファイルの構造が同一であるようにする必要があります。 • 同じフィールド • 同じフィールド順序 • 一致するフィールドが同じ長さ • 一致するフィールドが同じデータ型 出力は、既存ファイルとの間でファイル構造が違っている場合でも、Analyticsによって既存ファイルに追加されます。出力と既存のファイルの構造が一致しない場合は、データが混在、不足、不正確になります。
OPEN 省略可能	テーブルを開き、インデックスをテーブルに適用します。
ISOLOCALE *ロケールコード* 省略可能	**メモ** Analyticsの Unicode 版にのみ適用されます。 システムロケールは「*言語-国*」の形式で入力します。たとえば、カナダフランス語はコード「fr_ca」を入力します。 次のコードを使用します。 ◦ **言語** - ISO 639 標準言語コード ◦ **国** - ISO 3166 標準国コード 国コードを指定しない場合は、言語のデフォルト国が使用されます。 ISOLOCALEを使用しない場合は、デフォルト システム ロケールが使用されます。

例

単一のフィー津おで並べ替え、レコード全体を出力する

サンプル **Inventory** テーブルのレコードを製品番号で並べ替えるとします。並べ替えられたレコードは、新しい Analytics テーブル **Inventory_Product_Number** に抽出されます。

レコード全体が出力テーブルに含まれます。

```
SORT ON ProdNo TO "Inventory_Product_Number"
```

デフォルトの昇順を降順に変更するには、キーフィールド名の後に D を追加します。

```
SORT ON ProdNo D TO "Inventory_Product_Number"
```

単一のフィールドで並べ替え、フィールドのサブセットを出力する

サンプル **Inventory** テーブルのレコードを製品番号で並べ替えるとします。キーフィールドと指定された非キーフィールドのみが新しい Analytics テーブル **Inventory_Quantity_on_Hand** に抽出されます。

3 番目の非キーフィールド **QtyOH** に対し、出力テーブルでの表示名として **Qty on Hand**(在庫数量) が付けられます。

```
SORT ON ProdNo FIELDS ProdDesc ProdStat QtyOH AS "Qty on Hand" TO "Inventory_Quantity_on_Hand"
```

単一のフィールドで並べ替え、すべてのフィールドを出力する

サンプル **Inventory** テーブルのレコードを製品番号で並べ替えるとします。すべてのフィールドは、新しい Analytics テーブル **Inventory_Product_Number** に抽出されます。

FIELDS ALL がレコード全体の出力と異なる点は、**FIELDS ALL** では、ソーステーブルのすべての演算フィールドが出力テーブルの物理フィールドに変換され、フィールドに実際の演算された値が入力されることです。

```
SORT ON ProdNo FIELDS ALL TO "Inventory_Product_Number"
```

複数のフィールドで並べ替える(ネストされた並べ替え)

サンプル **Inventory** テーブルのレコードをロケーション、製品クラス、製品番号の順で並べ替えるとします。並べ替えられたレコードを、新しい Analytics テーブル **Inventory_Location_Class_Number** に抽出します。以上を行うコマンドの例は次のようになります。

```
SORT ON Location ProdCls ProdNo TO "Inventory_Location_Class_Number"
```

関連するフィールドを使用して並べ替える

サンプル **Ap_Trans** テーブルのレコードを次のフィールドで並べ替えるとします。

- Vendor_State(業者が拠点とする州。関連 : **Vendor** テーブル)
- Vendor_City(業者が拠点とする市区町村。関連 : **Vendor** テーブル)
- Vendor_No(業者番号。**Ap_Trans** テーブル)

すべての 3 つのキー フィールド、および関 連 するフィールド **Vendor.Vendor_Name** を含 む指 定 された非 キー フィールドを、新しい Analytics テーブル **Ap_Trans_State_City** に抽 出 する例を次 に示します。

```
SORT ON Vendor.Vendor_State Vendor.Vendor_City Vendor_No FIELDS Vendor.Vendor_Name
Invoice_No Invoice_Date Invoice_Amount Prodno Quantity Unit_Cost TO "Ap_Trans_State_City"
```

備 考

> **メモ**
>
> このコマンドの動 作 の詳 細 については、Analytics のヘルプを参 照 してください。

関 連 するフィールドで並 べ替 える

関 連 するフィールドで並 べ替 え、並 べ替 えられた出 力 テーブルに、非 キー フィールドとして関 連 するフィールドを含 めることができます。SORT コマンドで関 連 するフィールドを参 照 するには、*子 テーブル名 .フィールド 名* を指 定 します。

固 定 長 データ ファイルと可 変 長 データ ファイルの比 較

SORT コマンドは固 定 長 、可 変 長 いずれのデータ ファイルでも使 用 することができます。

STATISTICS コマンド

Analytics テーブルで、1 つ以上の数値または日付時刻フィールドについて統計計算します。

構文

STATISTICS <ON> *フィールド* <...*n*>|ALL} <STD> <MODMEDQ> <NUMBER *n*> <TO SCREEN|*ファイル名*|PRINT> <IF *テスト*> <WHILE *テスト*> <FIRST *範囲*|NEXT *範囲*> <APPEND>

パラメーター

名前	説明
ON *フィールド* <...*n*> | ALL	合計する 1 つ以上の数値または日付時刻フィールドを指定するか、あるいは、Analytics テーブル内の数値フィールドと日付時刻フィールドすべてについて統計を生成する場合は ALL を指定します。
STD 省略可能	STD を指定すると、指定したフィールドの標準偏差もほかの統計情報と共に計算されます。
MODMEDQ 省略可能	指定したフィールド群の最頻値、中央値、上位四分位数、および 3 番目の四分位数値を他の統計のほかに計算します。
NUMBER *n* 省略可能	処理の間に保持しておく高値と低値の数。デフォルト値は 5 です。
TO SCREEN | *ファイル名* | PRINT 省略可能	コマンドの結果を送信する場所： ◦ **SCREEN** - は Analytics の表示領域に結果を表示します ◦ ***ファイル名*** -は結果の保存先となるファイルです。 *ファイル名*には、適切なファイル拡張子を付けた文字列を引用符で囲んで指定します。 例：TO "Output.TXT" デフォルトでは、テーブル ファイルは、Analytics プロジェクトが入っているフォルダーに保存されます。 既存の異なるフォルダーにファイルを保存するには、絶対または相対ファイル パスを指定します。 • TO "C:\Output.TXT" • TO "Results\Output.TXT" ◦ **印刷** -- 通常使うプリンターに結果を送信します
IF *テスト* 省略可能	各レコードを処理するために真である必要がある条件式。コマンドは、その条件を満たすレコードに対してのみ実行されます。

名前	説明
	メモ IF パラメーターは、任意の範囲パラメーター(WHILE、FIRST、NEXT) が適用された後に、テーブルに残るレコードに対してのみ評価されます。
WHILE *テスト* 省略可能	各レコードを処理するために真である必要がある条件式。条件が false と評価するか、テーブルの最後に達したら、コマンドは実行を中止します。 **メモ** WHILE を FIRST または NEXT とともに使用する場合は、1 つの制限に達するとすぐに、レコードの処理が停止します。
FIRST *範囲* | NEXT *範囲* 省略可能	処理するレコード数: ◦ **FIRST** - 指定されたレコード数に達するまで、最初のレコードから処理を開始します ◦ **NEXT** - 指定されたレコード数に達するまで、現在選択されているレコードから処理を開始します *範囲*は処理するレコード数を指定します。 FIRST と NEXT を省略すると、すべてのレコードがデフォルトで処理されます。
APPEND 省略可能	コマンドの出力を既存ファイルに上書きしないで、そのファイルの末尾に追加します。 **メモ** コマンドの出力と既存のファイルの構造が同一であるようにする必要があります。 • 同じフィールド • 同じフィールド順序 • 一致するフィールドが同じ長さ • 一致するフィールドが同じデータ型 出力は、既存ファイルとの間でファイル構造が違っている場合でも、Analytics によって既存ファイルに追加されます。出力と既存のファイルの構造が一致しない場合は、データが混在、不足、不正確になります。

Analytics の出力変数

メモ

テーブル内の複数のフィールドを対象として統計を生成した場合、システムで生成される出力変数には、最初に列挙したフィールドの値が含まれています。

名前	含む
ABS*n*	コマンドによって計算された絶対値。
AVERAGE*n*	コマンドによって計算された平均値。
COUNT*n*	コマンドによって計算されたレコード数。 ◦ 変数名が COUNT1 の場合、それは最近実行されたコマンドのレコード数を格納しています。

名前	含む
	◦ 変数名 COUNT*n* の *n* が 1 より大きい場合、変数は GROUP コマンドの内で実行されたコマンドのレコード数を格納しています。 *n* の値は GROUP 内のコマンドの行番号に基づいて割り当てられます。たとえば、コマンドが GROUP コマンドの 1 行下にある場合、値は COUNT2 が割り当てられます。コマンドが GROUP コマンドの 4 行下にある場合、値は COUNT5 が割り当てられます。
HIGH*n*	このコマンドによって確認された 5 番目に高い値。 5 番目に高い値はデフォルト設定です。この設定を変更するには NUMBER パラメーターを使用します。たとえば、NUMBER 3 は、3 番目に高い値を格納することを指定するものです。 **メモ** Analytics で最も高い値が確認される際、重複値は除外されません。たとえば、降順の値リストが 100、100、99、98 の場合、3 番目に高い値は 98 でなく 99 になります。
LOW*n*	このコマンドによって確認された 5 番目に低い値。 5 番目に低い値はデフォルト設定です。この設定を変更するには NUMBER パラメーターを使用します。たとえば、NUMBER 3 は、3 番目に低い値を格納することを指定するものです。 **メモ** Analytics で最も低い値が確認される際、重複値は除外されません。たとえば、昇順の値リストが 1、1、2、3 の場合、3 番目に低い値は 3 でなく 2 になります。
MAX*n*	コマンドによって確認された最大値。
MEDIAN*n*	コマンドによって確認された中央値。
MIN*n*	コマンドによって確認された最小値。
MODE*n*	コマンドによって確認された最頻値。
Q25*n*	コマンドによって計算された最初の四分位数値(下位四分位数値)。
Q75*n*	コマンドによって計算された 3 番目の四分位数値(上位四分位数値)。
RANGE*n*	コマンドによって計算された最大値と最小値の差。
STDDEV*n*	コマンドによって計算された標準偏差値。
TOTAL*n*	コマンドによって計算された最初の合計値。 TOTAL コマンドが GROUP コマンド内になければ、*n* の値は 1 です。*n* の値は GROUP コマンド内の TOTAL コマンドの行番号に相当します。 詳細については、"GROUP コマンド" ページ 216を参照してください。

例

条件付き統計を生成する

次の例では、製品クラスID(ProdCls) が'01' であるレコードの、**数量**(Qty) フィールドの統計を生成しています。

```
STATISTICS ON Quantity IF ProdCls = "01"
```

STRATIFY コマンド

数値フィールドの値に基づいて、レコードを数値間隔でグループ化します。各間隔のレコード数をカウントし、指定した数値フィールドの小計を間隔ごとに求めます。

構文

STRATIFY <ON> *数値フィールド* MINIMUM *値* MAXIMUM *値* {<INTERVALS *数値*>|FREE *間隔値* <*...n*> *最後の間隔*} <SUPPRESS> <SUBTOTAL *数値フィールド* <*...n*>|SUBTOTAL ALL> <KEY *内訳フィールド*> <TO SCREEN|*テーブル名*|*ファイル名*|GRAPH|PRINT}> <IF *テスト*> <FIRST *範囲*|NEXT *範囲*> <WHILE *テスト*> <APPEND> <OPEN> <HEADER *ヘッダーテキスト*> <FOOTER *フッターテキスト*> <LOCAL> <STATISTICS>

パラメーター

名前	説明
ON *数値フィールド*	階層化する数値フィールドまたは式。
MINIMUM *値*	数値型のフィールドにのみ適用されます。最初の数値間隔の最小値。 FREE を使用している場合、MINIMUM は省略可能です。それ以外の場合は必要になります。
MAXIMUM *値*	数値型のフィールドにのみ適用されます。最後の数値間隔の最大値。 FREE を使用している場合、MAXIMUM は省略可能です。それ以外の場合は必要になります。
INTERVALS *数* 省略可能	数値型のフィールドにのみ適用されます。 MINIMUM 値と MAXIMUM 値によって指定された範囲の間に Analytics が生成する、均等な間隔の数。間隔の数を指定しない場合は、デフォルトの数が使用されます。 デフォルトは、[**オプション**]ダイアログ ボックスの [**コマンド**]タブの [**間隔の数**]によって決定されます。
FREE *間隔値* <*...n*> *最終間隔* 省略可能	数値型のフィールドにのみ適用されます。 各間隔の開始点と最後の間隔の終了点を指定することにより、カスタム サイズの間隔を作成することができます。 MINIMUM 値と MAXIMUM 値を指定した場合は、これらの値がそれぞれ最初の間隔の開始点と最後の間隔の終了点となり、各 *間隔値* が範囲内に追加の間隔を生成します。指定する間隔値は、MINIMUM 値より大きく、かつ MAXIMUM 値以下である必要があります。 間隔値は、数値順でなければならず、重複値を含めることはできません。次に例を示します。 たとえば、FREE -1000, 0, 1000, 2000, 3000 のように指定します。

名前	説明
	FREE とINTERVALS の両方を指定する場合は、INTERVALS が無視されます。
SUPPRESS 省略可能	MAXIMUM 値より大きい値とMINIMUM 値より小さい値をコマンド出力から除外します。
SUBTOTAL *数値フィールド <...n>* | SUBTOTAL ALL 省略可能	グループごとに小計を計算する 1 つ以上の数値フィールドまたは式。 複数のフィールドはスペースで区切る必要があります。テーブル内のすべての数値フィールドについて小計を求める場合はALL を指定します。 小計フィールドを選択しないと、階層化の対象とするフィールドの小計が自動的に計算されます。 階層化するフィールドと、それ以外の 1 つ以上のフィールドの小計を出したい場合、または小計された階層化対象フィールドの統計を含めたい場合は、階層化するフィールドを明示的に指定する必要があります。
KEY *ブレークフィールド* 省略可能	小計計算をグループ化するフィールドまたは式。*ブレークフィールド*の値が変わるたびに、小計が計算されます。 *ブレークフィールド*は、文字フィールドか式である必要があります。指定できるフィールドは 1 つだけですが、1 つ以上のフィールドを含んでいる式を使用することができます。
TO SCREEN | *テーブル名* | *ファイル名* | GRAPH | PRINT	コマンドの結果を送信する場所： ◦ **SCREEN** - はAnalytics の表示領域に結果を表示します ◦ ***テーブル名*** -は、結果の保存先となるAnalytics テーブルのことです。 *テーブル名*には、ファイル拡張子 .FIL を付けた文字列を引用符で囲んで指定する必要があります。例：TO "Output.FIL" デフォルトでは、テーブル データ ファイル(.fil) は、Analytics プロジェクトが入っているフォルダーに保存されます。 既存の異なるフォルダーにデータ ファイルを保存するには、絶対または相対ファイル パスを指定します。 • TO "C:\Output.FIL" • TO "Results\Output.FIL" **メモ** テーブル名は 64 文字の英数字(.FIL 拡張子を含まない) に制限されています。名前にはアンダースコア文字(_) を使用できますが、他の特殊文字やスペースは使用できません。名前の先頭を数字にすることはできません。 ◦ ***ファイル名*** -は結果の保存先となるファイルです。 *ファイル名*には、適切なファイル拡張子を付けた文字列を引用符で囲んで指定します。 例：TO "Output.TXT" デフォルトでは、テーブル ファイルは、Analytics プロジェクトが入っているフォルダーに保存されます。 既存の異なるフォルダーにファイルを保存するには、絶対または相対ファイル パスを指定します。 • TO "C:\Output.TXT" • TO "Results\Output.TXT" ◦ **GRAPH** - は結果をグラフに表示し、それをAnalytics の表示領域に表示します

名前	説明
	◦ **印刷** -- 通常使うプリンターに結果を送信します
IF *テスト* 省略可能	各レコードを処理するために真である必要がある条件式。コマンドは、その条件を満たすレコードに対してのみ実行されます。 **メモ** IF パラメーターは、任意の範囲パラメーター(WHILE、FIRST、NEXT) が適用された後に、テーブルに残るレコードに対してのみ評価されます。
FIRST *範囲* | NEXT *範囲* 省略可能	処理するレコード数: ◦ **FIRST** - 指定されたレコード数に達するまで、最初のレコードから処理を開始します ◦ **NEXT** - 指定されたレコード数に達するまで、現在選択されているレコードから処理を開始します *範囲*は処理するレコード数を指定します。 FIRST とNEXT を省略すると、すべてのレコードがデフォルトで処理されます。
WHILE *テスト* 省略可能	各レコードを処理するために真である必要がある条件式。条件が false と評価するか、テーブルの最後に達したら、コマンドは実行を中止します。 **メモ** WHILE を FIRST または NEXT とともに使用する場合は、1 つの制限に達するとすぐに、レコードの処理が停止します。
APPEND 省略可能	コマンドの出力を既存ファイルに上書きしないで、そのファイルの末尾に追加します。 **メモ** コマンドの出力と既存のファイルの構造が同一であるようにする必要があります。 • 同じフィールド • 同じフィールド順序 • 一致するフィールドが同じ長さ • 一致するフィールドが同じデータ型 出力は、既存ファイルとの間でファイル構造が違っている場合でも、Analytics によって既存ファイルに追加されます。出力と既存のファイルの構造が一致しない場合は、データが混在、不足、不正確になります。
OPEN 省略可能	コマンドを実行した後、コマンドによって作成されたテーブルを開きます。コマンドが出力テーブルを作成する場合にのみ有効です。
HEADER *ヘッダーテキスト* 省略可能	レポートの各ページの最上部に挿入されるテキスト。 *ヘッダーテキスト*は引用符で囲んだ文字列として指定する必要があります。この値は、Analytics のシステム変数である HEADER の値よりも優先されます。
FOOTER *フッターテキスト* 省略可能	レポートの各ページの最下部に挿入されるテキスト。 *フッターテキスト*は引用符で囲んだ文字列として指定する必要があります。この値は、Analytics のシステム変数である FOOTER の値よりも優先されます。
LOCAL 省略可能	Analytics プロジェクトと同じ場所に出力ファイルを保存します。

名前	説明
	メモ Analytics テーブルである出力ファイルを含むサーバー テーブルに対してコマンドを実行するときにのみ適用されます。
STATISTICS 省略可能	**メモ** SUBTOTAL も指定されていない場合は使用できません。 すべての SUBTOTAL フィールドの平均値、最小値、および最大値を計算します。

例

請求金額で階層化する

Ar(売掛金)テーブルを **Invoice_Amount**(請求金額)フィールドで階層化する必要があるとします。請求金額の小計は自動的に求められます。

出力は以下のような $1000 間隔でグループ化されます。

- $0 ～ $999.99
- $1,000 ～ $1,999.99
- といった具合です。

各間隔の請求金額合計も求められます。

```
OPEN Ar
STRATIFY ON Invoice_Amount MINIMUM 0 MAXIMUM 10000 INTERVALS 10 TO "Stratified_
invoices.FIL"
```

備考

メモ

このコマンドの動作の詳細については、Analytics のヘルプを参照してください。

機能の仕組み

STRATIFY は、数値フィールドの値に基づいて、レコードを均等な数値間隔またはカスタム サイズの数値間隔にグループ化します。

出力には間隔ごとに、ソース テーブルのうち、その間隔に属するレコードの数が記録された特別なレコードが含まれます。

MINIMUM および MAXIMUM 値を自動的に入力する

STRATIFY コマンドを実行する前に、階層化の対象フィールドで STATISTICS コマンドまたは PROFILE コマンドを実行して、フィールド内の最小値と最大値を MINIMUM パラメーターと MAXIMUM パラメーターに自動的に設定することができます。

自動生成された小計と統計フィールドの名前

STATISTICS を使用して、1 つ以上の SUBTOTAL フィールドで統計演算を実行し、結果を Analytics テーブルに出力する場合は、パラメーターによって自動生成されたフィールドの名前は次のようになります。

自動生成されたフィールドの説明	出力テーブルのフィールド名	出力テーブルの列見出し(表示名)
小計フィールド	*集計対象となる、ソース テーブルのフィールド名*	**Total** + *集計対象となる、ソース テーブルの代替列見出し*
平均フィールド	**a_***集計対象となる、ソース テーブルのフィールド名*	**Average** + *集計対象となる、ソース テーブルの代替列見出し*
最小フィールド	**m_***集計対象となる、ソース テーブルのフィールド名*	**Minimum** + *集計対象となる、ソース テーブルの代替列見出し*
最大フィールド	**x_***集計対象となる、ソース テーブルのフィールド名*	**Maximum** + *集計対象となる、ソース テーブルの代替列見出し*

SUMMARIZE コマンド

1 つ以上の文字フィールド、数値フィールド、または日付時刻フィールドの同じ値に基づいて、レコードをグループ分けします。各グループのレコード数をカウントし、指定した数値フィールドの小計をグループごとに求めます。

構文

SUMMARIZE ON *キーフィールド <...n>* <SUBTOTAL *数値フィールド <...n>*|SUBTOTAL ALL> <OTHER *フィールド <...n>*|OTHER ALL> <TO {SCREEN|*テーブル名*|PRINT> <IF *テスト*> <WHILE *テスト*> <FIRST *範囲*|NEXT *範囲*> <PRESORT> <APPEND> <OPEN> <LOCAL> <HEADER *ヘッダーテキスト*> <FOOTER *フッターテキスト*> <STATISTICS> <MODMEDQ> <STDEV> <CPERCENT> <ISOLOCALE *ロケールコード*>

パラメーター

名前	説明
ON *キーフィールド <...n>*	要約する 1 つ以上の文字、数値、日付時刻フィールド。複数のフィールドはスペースで区切る必要があります。また、異なるデータ型を指定できます。
SUBTOTAL *数値フィールド <...n>* | SUBTOTAL ALL 省略可能	グループごとに小計を計算する 1 つ以上の数値フィールドまたは式。 複数のフィールドはスペースで区切る必要があります。テーブル内のすべての数値フィールドについて小計を求める場合は ALL を指定します。
OTHER *フィールド <...n>* | OTHER ALL 省略可能	出力に含める 1 つ以上の追加フィールド。 ◦ ***フィールド <...n>*** - 指定した 1 つまたは複数のフィールドが含まれます。 ◦ **ALL** - キーフィールドまたは小計フィールドとして指定しなかった、テーブル内のすべてのフィールドが含まれます。 OTHER は、各要約グループ内のすべてのレコードが同じ値を含んでいるフィールドに対してのみ使用してください。要約されるグループに対し異なった値を含んでいるフィールドを指定すると、グループ内の最初のレコードの値しか表示されず、これでは意味がありません。 例： ◦ **テーブルを顧客番号で要約する** -場合には、適切な「OTHER(その他の) フィールド」の例としては**顧客名**を指定できます。通常、顧客名は、同じ顧客番号を持つすべてのレコードで同じです。 ◦ **州で業者テーブルを要約する** -不適切「OTHER(その他の) フィールド」は**都市**です。各州の一覧の最初の都市のみが出力に表示されます。このような場合、州と市の両方をキーフィールドとして(州、市の順番で) 要約を行うことをお勧めします。
TO SCREEN | *テーブル名* | PRINT	コマンドの結果を送信する場所： ◦ **SCREEN** - は Analytics の表示領域に結果を表示します

名前	説明
	◦ ***テーブル名*** -は、結果の保存先となる Analytics テーブルのことです。 *テーブル名*には、ファイル拡張子 .FIL を付けた文字列を引用符で囲んで指定する必要があります。例：TO "Output.FIL" デフォルトでは、テーブル データ ファイル(.fil) は、Analytics プロジェクトが入っているフォルダーに保存されます。 既存の異なるフォルダーにデータ ファイルを保存するには、絶対または相対ファイル パスを指定します。 • TO "C:\Output.FIL" • TO "Results\Output.FIL" **メモ** テーブル名は 64 文字の英数字(.FIL 拡張子を含まない) に制限されています。名前にはアンダースコア文字(_) を使用できますが、他の特殊文字やスペースは使用できません。名前の先頭を数字にすることはできません。 ◦ **印刷** -- 通常使うプリンターに結果を送信します
IF テスト 省略可能	各レコードを処理するために真である必要がある条件式。コマンドは、その条件を満たすレコードに対してのみ実行されます。 **メモ** IF パラメーターは、任意の範囲パラメーター(WHILE、FIRST、NEXT) が適用された後に、テーブルに残るレコードに対してのみ評価されます。
WHILE テスト 省略可能	各レコードを処理するために真である必要がある条件式。条件が false と評価するか、テーブルの最後に達したら、コマンドは実行を中止します。 **メモ** WHILE を FIRST または NEXT とともに使用する場合は、1 つの制限に達するとすぐに、レコードの処理が停止します。
FIRST *範囲* | NEXT *範囲* 省略可能	処理するレコード数： ◦ **FIRST** - 指定されたレコード数に達するまで、最初のレコードから処理を開始します ◦ **NEXT** - 指定されたレコード数に達するまで、現在選択されているレコードから処理を開始します *範囲*は処理するレコード数を指定します。 FIRST と NEXT を省略すると、すべてのレコードがデフォルトで処理されます。
PRESORT 省略可能	コマンドを実行する前にキー フィールドでテーブルを並べ替えます。 **メモ** GROUP コマンドの内部では PRESORT を使用することができません。 **PRESORT を使用する場合** **PRESORT** を使用すると、出力は並べ替えられ、キー フィールド内の同一値セットごと、または値の等しい組み合わせごとに 1 つの一意のグループが含まれます。

名前	説明
	ヒント 入力テーブルがすでに並べ替えられている場合は、PRESORT を指定しないことで処理時間を短縮できます。 **Presort を使用しない場合** PRESORT を使用しない場合、出力結果は、入力テーブルの並べ替え順序を使用します。 キーフィールドに入っている同一の値が連続して並んでいない場合、出力結果には、同一値セットごと、または値の等しいの組み合わせごとに 2 つ以上のグループが含まれることになります。 **メモ** コンテキストによっては、同一値セットごと、または値の等しい組み合わせに 2 つ以上のグループが含まれることが、要約の目的よりも優先されることがあります。
APPEND 省略可能	コマンドの出力を既存ファイルに上書きしないで、そのファイルの末尾に追加します。 **メモ** コマンドの出力と既存のファイルの構造が同一であるようにする必要があります。 • 同じフィールド • 同じフィールド順序 • 一致するフィールドが同じ長さ • 一致するフィールドが同じデータ型 出力は、既存ファイルとの間でファイル構造が違っている場合でも、Analytics によって既存ファイルに追加されます。出力と既存のファイルの構造が一致しない場合は、データが混在、不足、不正確になります。
OPEN 省略可能	コマンドを実行した後、コマンドによって作成されたテーブルを開きます。コマンドが出力テーブルを作成する場合にのみ有効です。
LOCAL 省略可能	Analytics プロジェクトと同じ場所に出力ファイルを保存します。 **メモ** Analytics テーブルである出力ファイルを含むサーバーテーブルに対してコマンドを実行するときにのみ適用されます。
HEADER *ヘッダーテキスト* 省略可能	レポートの各ページの最上部に挿入されるテキスト。 *ヘッダーテキスト*は引用符で囲んだ文字列として指定する必要があります。この値は、Analytics のシステム変数である HEADER の値よりも優先されます。
FOOTER *フッターテキスト* 省略可能	レポートの各ページの最下部に挿入されるテキスト。 *フッターテキスト*は引用符で囲んだ文字列として指定する必要があります。この値は、Analytics のシステム変数である FOOTER の値よりも優先されます。
STATISTICS 省略可能	**メモ** SUBTOTAL も指定されていない場合は使用できません。

名前	説明
	すべての SUBTOTAL フィールドの平均値、最小値、および最大値を計算します。
MODMEDQ 省略可能	**メモ** SUBTOTAL も指定されていない場合は使用できません。 最頻値、中央値、最初の四分位数、および3 番目の四分位数値がすべての SUBTOTAL フィールドに対して計算されます。
STDEV 省略可能	**メモ** SUBTOTAL も指定されていない場合は使用できません。 標準偏差と合計の割合がすべての SUBTOTAL フィールドに対して計算されます。
CPERCENT 省略可能	レコード数における各グループの割合を計算します。
ISOLOCALE 省略可能	**メモ** Analytics の Unicode 版にのみ適用されます。 システム ロケールは「*言語-国*」の形式で入力します。たとえば、カナダ フランス語はコード「fr_ca」を入力します。 次のコードを使用します。 ◦ **言語** - ISO 639 標準言語コード ◦ **国** - ISO 3166 標準国コード 国コードを指定しない場合は、言語のデフォルト国が使用されます。 ISOLOCALE を使用しない場合は、デフォルト システム ロケールが使用されます。

例

顧客ごとの総取引額

Customer_Number(顧客番号) フィールドに基づいて Ar(売掛金) テーブルを要約し、**Trans_Amount**(取引額) フィールドの小計を求めたいとします。出力は、顧客ごとでグループ化され、各顧客の総取引額が含まれます。

```
OPEN Ar
SUMMARIZE ON Customer_Number SUBTOTAL Trans_Amount TO "Customer_total.FIL"
PRESORT
```

顧客ごとの取引日別の総取引額

Customer_Number(顧客番号) と**Trans_Date**(取引日) フィールドに基づいて Ar(売掛金) テーブルを要約したいとします。**Trans_Amount** フィールドの小計を求めます。

出力は、顧客ごとと、顧客内の取引日ごとにグループ化され、各顧客の取引日別に総取引額が含まれます。

```
OPEN Ar
SUMMARIZE ON Customer_Number Trans_Date SUBTOTAL Trans_Amount TO "Customer_total_
by_date.FIL" PRESORT
```

顧客ごと、取引日別の取引額の合計、平均、最小、および最大

直前の例にSTATISTICS を追加します。

顧客ごとに、その顧客が取引をした日付別の取引額の小計に加え、日付別の取引額の平均値、最小値、および最大値も計算することができます。次のように指定します。

```
OPEN Ar
SUMMARIZE ON Customer_Number Trans_Date SUBTOTAL Trans_Amount TO "Customer_
stats_by_date.FIL" PRESORT STATISTICS
```

同じ取引額、同じ日付

Trans_Date(取引日) フィールドと**Trans_Amount(取引金額)** フィールドに基づいて、クレジット カード取引テーブルを集計する場合を考えます。

出力は日付別にグループ化され、その日付内で金額別にグループ化されます。関連付けられたカウントを使用して、同じ金額と同じ日付の取引を特定できます。以上を行うコマンドの例を次に示します。

```
OPEN CC_Trans
SUMMARIZE ON Trans_Date Trans_Amount TO "Transactions_by_date_amount.FIL" OPEN
PRESORT
SET FILTER TO COUNT > 1
```

備考

> **メモ**
>
> このコマンドの動作の詳細については、Analytics のヘルプを参照してください。

機能の仕組み

SUMMARIZE は、1 つ以上の文字フィールド、数値フィールド、または日付時刻フィールドで、同じ値または同じ値の組み合わせを持つレコードをグループ化します。出力にはグループごとに、ソース テーブルのうち、そのグループに属するレコードの数が記録された特別なレコードが含まれます。

小計と統計：出力結果の計算とフィールド名

1つ以上の任意のパラメーターを使用して、指定するSUBTOTALフィールドで統計演算を実行することもできます。統計計算は、出力においてグループ別に内訳が示されます。

省略可能なパラメーター	出力テーブルの列見出し(表示名)	出力テーブルのフィールド名	小計フィールドで実行された計算
SUBTOTAL	**Total** + *小計対象となる代替列見出し*	*小計対象となるフィールド名*	各グループの小計された値
STATISTICS	**Average** + *小計対象となる代替列見出し*	**a_***小計対象となるフィールド名*	各グループの平均値
	Minimum + *小計対象となる代替列見出し*	**m_***小計対象となるフィールド名*	各グループの最小値
	Maximum + *小計対象となる代替列見出し*	**x_***小計対象となるフィールド名*	各グループの最大値
MODMEDQ	**Median** + *小計対象となる代替列見出し*	**c_***小計対象となるフィールド名*	各グループの中央値 ◦ 奇数の値セット：中央値 ◦ 偶数の値セット：中央にある2つの値の平均
	Mode + *小計対象となる代替列見出し*	**o_***小計対象となるフィールド名*	各グループの最も頻繁に発生する値 ◦ 2回以上出現する値がない場合は、"N/A" が表示される ◦ 関連付けの場合は、最も低い値が表示される
	Q25 + *小計対象となる代替列見出し*	**q_***小計対象となるフィールド名*	各グループの最初の四分位数値(下四分位数値) ◦ 結果はAnalyticsのアルゴリズムによって計算された補間値である ◦ Microsoft ExcelのQUARTILEおよびQUARTILE.INC関数と同じ結果を生成する
	Q75 + *小計対象となる代替列見出し*	**p_***小計対象となるフィールド名*	各グループの3番目の四分位数(下四分位数) ◦ 結果はAnalyticsのアルゴリズムによって計算された補間値である ◦ Microsoft ExcelのQUARTILEおよびQUARTILE.INC関数と

省略可能なパラメーター	出力テーブルの列見出し(表示名)	出力テーブルのフィールド名	小計フィールドで実行された計算
			同じ結果を生成する
STDEV	**STDDEV** + *小計対象となる代替列見出し*	**d_***小計対象となるフィールド名*	各グループの標準偏差
	% Field + *小計対象となる代替列見出し*	**f_***小計対象となるフィールド名*	フィールドの合計に対する割合として表される各グループの小計
CPERCENT	**パーセントカウント**	COUNT_PERCENTAGE	各グループに属するソーステーブルレコードの割合 **メモ** 小計フィールドは必要ありません。

TOP コマンド

Analytics テーブルの最 初 のレコードに移 動します。

構 文

```
TOP
```

パラメーター

このコマンドにはパラメーラーがありません。

備 考

TOP を使 用する場 面

直 前 に実 行 されたコマンド(FIND など) により、テーブル内 で先 頭 以 外 のレコードが選 択 されていた場 合 に先 頭 レコードに移 動 するのに、TOP を使 用 することができます。

TOTAL コマンド

Analytics テーブルの 1 つ以上のフィールドの合計値を計算します。

構文

TOTAL {<FIELDS> *数値フィールド* <...*n*>|<FIELDS> ALL} <IF *テスト*> <WHILE *テスト*> <FIRST *範囲*|NEXT *範囲*>

パラメーター

名前	説明
FIELDS *数値フィールド* <...*n*> | FIELDS ALL	値の合計を求める数値フィールド。テーブル内の数値フィールドの各々について合計を求める場合は ALL を指定します。
IF *テスト* 省略可能	各レコードを処理するために真である必要がある条件式。コマンドは、その条件を満たすレコードに対してのみ実行されます。 **メモ** IF パラメーターは、任意の範囲パラメーター(WHILE、FIRST、NEXT) が適用された後に、テーブルに残るレコードに対してのみ評価されます。
WHILE *テスト* 省略可能	各レコードを処理するために真である必要がある条件式。条件が false と評価するか、テーブルの最後に達したら、コマンドは実行を中止します。 **メモ** WHILE を FIRST または NEXT とともに使用する場合は、1 つの制限に達するとすぐに、レコードの処理が停止します。
FIRST *範囲* | NEXT *範囲* 省略可能	処理するレコード数: ◦ **FIRST** - 指定されたレコード数に達するまで、最初のレコードから処理を開始します ◦ **NEXT** - 指定されたレコード数に達するまで、現在選択されているレコードから処理を開始します *範囲*は処理するレコード数を指定します。 FIRST と NEXT を省略すると、すべてのレコードがデフォルトで処理されます。

Analytics の出力変数

> **メモ**
>
> テーブル内の複数のフィールドを合計した場合、システムで生成される出力変数には、最初に列挙したフィールドの合計が含まれています。

名前	含む
TOTAL*n*	コマンドによって計算された最初の合計値。 TOTAL コマンドが GROUP コマンド内になければ、*n* の値は 1 です。*n* の値は GROUP コマンド内の TOTAL コマンドの行番号に相当します。 詳細については、"GROUP コマンド" ページ 216を参照してください。

例

最初の 25 レコードの合計を求める

次の例は、テーブル内の最初の 25 レコードの **MKTVAL** フィールドの総量を計算しています。

```
TOTAL FIELDS MKTVAL FIRST 25
```

備考

TOTAL の用途

TOTAL は、コントロール合計を作成して、ソースデータの完全性や正確性を検証する場合に使用できます。このコマンドにより、指定されたフィールドあるいは式の算術合計が計算されます。

TRAIN コマンド

自動的な機械学習を使用して、トレーニング データ セットに対する最適の予測モデルを作成します。

構文

TRAIN {CLASSIFIER|REGRESSOR} <ON> *キー フィールド* <...*n*> TARGET *ラベル付きフィールド* SCORER {ACCURACY|AUC|F1|LOGLOSS|PRECISION|RECALL|MAE|MSE|R2} SEARCHTIME *分* MAXEVALTIME *分* MODEL *モデル名* TO *テーブル名* <IF *テスト*> <WHILE *テスト*> <FIRST *範囲* |NEXT *範囲*> FOLDS *分割数* <SEED *シード値*> <LINEAR> <NOFP>

> **メモ**
>
> TRAIN コマンドで使用されるデータセットのサポートされる最大サイズは 1 GB です。

パラメーター

名前	説明
CLASSIFIER | REGRESSOR	予測モデルを学習するときに使用する予測タイプ。 ◦ **CLASSIFIER** - 分類アルゴリズムを使用して、モデルを学習します レコードが属するクラスまたはカテゴリを予測する場合は分類を使用します。 ◦ **REGRESSOR** - 回帰アルゴリズムを使用して、モデルを学習します レコードに関連付けられた数値を予測する場合は回帰を使用します。
ON *キー フィールド* <...*n*>	1 つ以上学習入力フィールド。 フィールドは、文字、数値、または論理型を使用できます。複数のフィールドはスペースで区切る必要があります。 > **メモ** > 文字フィールドは「分類的」である必要があります。つまり、カテゴリまたはクラスを特定し、最大数の一意の値を含む必要があります。 > 最大値は [**最大カテゴリ**] オプション([**ツール > オプション > コマンド**]) で指定されます。
TARGET *ラベル付きフィールド*	学習入力フィールドに基づいて予測するようにモデルが学習されているフィールド。 別の予測タイプ(分類または回帰) は別のフィールド データ型で動作します。 CLASSIFIER で有効 — 文字または論理対象フィールド REGRESSOR で有効 — 数値対象フィールド

<table>
<tr><th>名前</th><th>説明</th></tr>
<tr><td>SCORER ACCURACY | AUC | F1 | LOGLOSS | PRECISION | RECALL | MAE | MSE | R2</td><td>生成されたモデルのスコアを決定(調整およびランク付け)するときに使用するメトリクス。
生成されたモデルのうち、このメトリクスの最善値を有するモデルが保持され、そうでないモデルは破棄されます。
使用している予測タイプ(分類または回帰)に応じて、メトリクスの別のサブセットが有効となります。<table><tr><td>CLASSIFIER で有効</td><td>SCORER ACCURACY | AUC | F1 | LOGLOSS | PRECISION | RECALL | MAE | MSE | R2</td></tr><tr><td>REGRESSOR で有効</td><td>MAE | MSE | R2</td></tr></table></td></tr>
<tr><td>SEARCHTIME 分</td><td>予測モデルの学習および最適化にかかる合計時間(分)。
学習および最適化は、異なるパイプライン構成の検索を伴います(異なるモデル、プリプロセッサー、およびハイパーパラメーターの組み合わせ)。
メモ
TRAIN コマンドの合計実行時間は SEARCHTIME と最大で MAXEVALTIME の 2 倍です。
ヒント
MAXEVALTIME の 10 倍である SEARCHTIME を指定します。
この時間割り当ては、処理時間と多様なモデルタイプの評価を可能にすることの間で、合理的なバランスを取っています。</td></tr>
<tr><td>MAXEVALTIME 分</td><td>最大実行時間は、モデル評価ごとの分数です。
ヒント
100 MB の学習データごとに 45 分を割り当てます。
この時間割り当ては、処理時間と多様なモデルタイプの評価を可能にすることの間で、合理的なバランスを取っています。</td></tr>
<tr><td>MODEL モデル名</td><td>学習プロセスで生成されたモデルファイル。
モデルファイルには、学習データセットに最適なモデルが含まれます。モデルを PREDICT コマンドに入力し、新しい未確認のデータセットに関する予測を生成します。
引用された文字列としてモデル名を指定します。例: <code>TO "Loan_default_prediction"</code>
ファイル拡張子 <code>*.model</code> を指定するか、それを Analytics で自動で指定させます。
デフォルトでは、モデルファイルは、Analytics プロジェクトが入っているフォルダーに保存されます。
既存の異なるフォルダーにモデルファイルを保存するには、絶対または相対ファイルパスを指定します。
TO "C:\Loan_default_prediction"
TO "ML Train output\Loan_default_prediction.model"</td></tr>
<tr><td>TO テーブル名</td><td>学習プロセスで生成されたモデル評価テーブルの名前。
モデル評価テーブルには、以下の 2 つの異なるタイプの情報が格納されています。</td></tr>
</table>

名前	説明
	◦ **スコアラー/メトリクス** -。これらは、学習プロセスによって生成されたモデル ファイルの予測パフォーマンスの定量的な推定、分類メトリクスまたは回帰メトリクスを意味します。 メトリクスが異なれば、提供される推定のタイプも異なります。**スコアラー**には、SCORER に指定したメトリクスが表示されます。**メトリクス**には、指定しなかったメトリクスが表示されます。 ◦ **重要性/係数** -(降順) : モデルによって生成された予測に対する各機能(予測印子) の寄与度を示す値。 *テーブル名*、.FIL ファイル拡張子を持つ引用符で囲まれた文字列として指定します。例 : TO "Model_evaluation.FIL" デフォルトでは、テーブル データ ファイル(.FIL) は、Analytics プロジェクトが入っているフォルダーに保存されます。 既存の異なるフォルダーにデータ ファイルを保存するには、絶対または相対ファイル パスを指定します。 ◦ TO "C:\Model_evaluation.FIL" ◦ TO "ML Train output\Model_evaluation.FIL" **メモ** テーブル名は 64 文字の英数字(.FIL 拡張子を含まない) に制限されています。名前にはアンダースコア文字(_) を使用できますが、他の特殊文字やスペースは使用できません。名前の先頭を数字にすることはできません。
IF *テスト* 省略可能	各レコードを処理するために真である必要がある条件式。コマンドは、その条件を満たすレコードに対してのみ実行されます。 **メモ** IF パラメーターは、任意の範囲パラメーター(WHILE、FIRST、NEXT) が適用された後に、テーブルに残るレコードに対してのみ評価されます。
WHILE *テスト* 省略可能	各レコードを処理するために真である必要がある条件式。条件が false と評価するか、テーブルの最後に達したら、コマンドは実行を中止します。 **メモ** WHILE を FIRST または NEXT とともに使用する場合は、1 つの制限に達するとすぐに、レコードの処理が停止します。
FIRST *範囲* | NEXT *範囲* 省略可能	処理するレコード数 : ◦ **FIRST** - 指定されたレコード数に達するまで、最初のレコードから処理を開始します ◦ **NEXT** - 指定されたレコード数に達するまで、現在選択されているレコードから処理を開始します *範囲*は処理するレコード数を指定します。 FIRST と NEXT を省略すると、すべてのレコードがデフォルトで処理されます。
FOLDS *分割数*	モデルを評価および最適化するときに使用する交差検証分割数。 分割は学習データセットの下位分割であり、交差検証プロセスで使用されます。 一般的に、モデルの学習時に、5 ～ 10 分割を使用すると、適切な結果が得られます。許可された最小分割数は 2 です。最大数は 10 です。

名前	説明
	ヒント 学習データセットが小さく、分割数が増えると、モデルの予測性能の推定値が改善されることがありますが、全体的な実行時間も長くなります。
SEED *シード値* 省略可能	Analytics の乱数ジェネレーターを初期化するために使用するシード値。 SEED を省略した場合は、シード値がランダムに選択されます。 明示的にシード値を指定し、将来に学習プロセスを同じデータセットで複製したい場合は、それを記録します。
LINEAR 省略可能	学習して、線形モデルのみのスコアを決定します。 学習プロセスに線形モデルのみを含めると、出力の係数が保証されます。 LINEAR が省略される場合、分類または再帰に関連するすべてのモデルタイプが評価されます。
NOFP 省略可能	機能選択とデータ前処理を学習プロセスから除外します。 機能選択は、予測モデルを最適化する際に最も有用な学習データセットで、自動化されたフィールドの選択です。自動化された選択は予測性能を改善することがありますが、モデル最適化に関連するデータ量が減ります。 データ前処理は、学習データセットでの調整や標準化などの変換を実行し、学習アルゴリズムにより適したものにします。 **注意** 理由がある場合にかぎり、機能選択とデータ前処理を除外してください。

例

分類モデルの学習

債務不履行になる融資申請者を予測するために後続のプロセスで使用できる分類モデルに学習させる必要があるとします。

顧客が債務不履行になったかどうかなど各融資の確認済みの結果が含まれる、融資データ履歴セットに関する学習をモデルに行わせます。

直後の予測プロセスで、TRAIN コマンドで生成されたモデルを使用して、現在の融資申請者データを処理します。

```
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
```

回帰モデルに学習させる

将来の住宅販売価格を予測するために後続のプロセス内で使用できる回帰モデルに学習させる必要があるとします。

販売価格を含む最近の住宅販売データのセットを当該のモデルに学習させます。

直後の予測プロセスで、TRAINコマンドで生成されたモデルを使って住宅価格評価を生成します。

```
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
```

備考

> **メモ**
>
> このコマンドの動作の詳細については、Analytics のヘルプを参照してください。

VERIFY コマンド

Analytics テーブル中 のデータがテーブル レイアウト のフィールド定 義とー 致していることを確 認 することにより、1 つ以 上 の Analytics テーブルのデータの妥 当 性 エラーをチェックします。

構 文

VERIFY {<FIELDS> *フィールド* <...*n*>|<FIELDS> ALL} <IF *テスト*> <WHILE *テスト*> <FIRST *範 囲* |NEXT *範 囲*> <ERRORLIMIT *n*> <TO {SCREEN|*ファイル名*|PRINT}> <APPEND>

パラメーター

名 前	説 明
FIELDS *フィールド* <...*n*> | FIELDS ALL	検 証 するフィールド または式 。 ALL を指 定 すると、テーブルのすべてのフィールド を検 証 します。 **メモ** 定 義 では、演 算 フィールド 、アド ホック式 、および バイナリ フィールド は常 に有 効 です。
IF *テスト* 省 略 可 能	各 レコード を処 理 するために真 である必 要 がある条 件 式 。コマンド は、その条 件 を満 たすレコー ド に対してのみ実 行 されます。 **メモ** IF パラメーターは、任 意 の範 囲 パラメーター(WHILE、FIRST、NEXT) が適 用 され た後 に、テーブルに残 るレコード に対してのみ評 価 されます。
WHILE *テスト* 省 略 可 能	各 レコード を処 理 するために真 である必 要 がある条 件 式 。条 件 が false と評 価 するか、テーブル の最 後 に達 したら、コマンド は実 行 を中 止 します。 **メモ** WHILE を FIRST または NEXT とともに使 用 する場 合 は、1 つの制 限 に達 すると すぐに、レコード の処 理 が停 止 します。
FIRST *範 囲* | NEXT *範 囲* 省 略 可 能	処 理 するレコード 数 : ◦ **FIRST** - 指 定 されたレコード 数 に達 するまで、最 初 のレコード から処 理 を開 始 します ◦ **NEXT** - 指 定 されたレコード 数 に達 するまで、現 在 選 択 されているレコード から処 理 を開 始 し ます *範 囲* は処 理 するレコード 数 を指 定 します。 FIRST と NEXT を省 略 すると、すべてのレコード がデフォルト で処 理 されます。
ERRORLIMIT *n* 省 略 可 能	コマンド が停 止 するまでに許 容 されるエラー数 。デフォルト 値 は 10 です。

名前	説明
TO SCREEN | *ファイル名* | PRINT 省略可能	コマンドの結果を送信する場所： ◦ **SCREEN** - はAnalyticsの表示領域に結果を表示します ◦ ***ファイル名*** -は結果の保存先となるファイルです。 *ファイル名*には、適切なファイル拡張子を付けた文字列を引用符で囲んで指定します。 例：TO "Output.TXT" デフォルトでは、テーブルファイルは、Analyticsプロジェクトが入っているフォルダーに保存されます。 既存の異なるフォルダーにファイルを保存するには、絶対または相対ファイルパスを指定します。 • TO "C:\Output.TXT" • TO "Results\Output.TXT" ◦ **印刷** -- 通常使うプリンターに結果を送信します
APPEND 省略可能	コマンドの出力を既存ファイルに上書きしないで、そのファイルの末尾に追加します。 **メモ** コマンドの出力と既存のファイルの構造が同一であるようにする必要があります。 • 同じフィールド • 同じフィールド順序 • 一致するフィールドが同じ長さ • 一致するフィールドが同じデータ型 出力は、既存ファイルとの間でファイル構造が違っている場合でも、Analyticsによって既存ファイルに追加されます。出力と既存のファイルの構造が一致しない場合は、データが混在、不足、不正確になります。

Analyticsの出力変数

名前	含む
WRITE*n*	コマンドによって確認された妥当性エラーの合計数。

例

データの検証と誤謬限度の指定

テーブル内のすべての列を検証し、誤謬上限を10に設定するとします。次のコマンドは、10個のデータ検証エラーが検出されると処理を停止します。

```
VERIFY ALL ERRORLIMIT 10 TO "ImportErrors.txt"
```

備考

機能の仕組み

VERIFY は、1 つ以上のフィールドの値をテーブルレイアウトの各フィールドに指定されているデータ型と比較し、エラーを報告します。このコマンドは、次のことを確認します。

- **文字フィールド** -が有効な文字だけを含んでおり、印刷できない文字を含んでいない。
- **数値フィールド** -が有効な数字データだけを含んでいる。数値フィールドは、数字のほかに、数字の前に 1 つのプラス記号またはマイナス記号を含んでいたり、1 つの小数点を含んでいたりしても構いません。
- **日付時刻フィールド** -が、有効な日付、日付時刻、または時刻を含んでいる。

識別される各エラーについては、レコード番号とフィールド名が、16 進形式の無効な値と共に出力されます。

関数

ABS( ) 関数

数式の絶対値を返します。数の絶対値は符号のない数です。

構文

```
ABS(数値)
```

パラメーター

名前	種類	説明
数値	数値	絶対値を確認する値。

出力

数値。

例

基本的な例

7.2 が返されます。

```
ABS(7.2)
```

7.2 が返されます。

```
ABS(-7.2)
```

AGE( ) 関数

指定された日付を指定された締切日または現在のオペレーティングシステムの日付と比較して、経過日数を返します。

構文

AGE(*日付/日付時刻/文字列* <,*締切日*>)

パラメーター

名前	種類	説明
日付/日付時刻/文字列	文字 日付時刻	年齢を調べるフィールド、式、またはリテラル値。
締切日 省略可能	文字 日付時刻	*日付/日付時刻/文字列*と比較するフィールド、式、またはリテラル値。これを省略した場合は、締切日として、現在のオペレーティングシステム日付が使用されます。

> **メモ**
>
> *日付/日付時刻/文字列*と*締切日*は日付時刻値を受け入れることはできますが、値の時刻部分は無視されます。時刻値だけで AGE( ) を使用することはできません。

出力

数値。

例

基本的な例

締切日がない

2014 年 12 月 31 日と現在の日付の間の日数を返します。

- 正の値が返された場合、値は、過去 2014 年 12 月 31 日からこれまでの日数に相当します。
- 負の値が返された場合、値は、将来 2014 年 12 月 31 日になるまでの日数に相当します。
- '0' が返された場合、2014 年 12 月 31 日は現在の日付です。

```
AGE(`20141231`)
```

期日フィールドの日付と現在の日付の間の日数が返されます。

```
AGE(期日)
```

データ型の混在

以下のいずれによっても、指定した2つの日付間の日数である518が返されます。

```
AGE(`20130731`,`20141231`)
```

```
AGE("20130731","20141231")
```

```
AGE(`20130731`,"20141231")
```

```
AGE(`20130731 235959`,`20141231`)
```

締切日とフィールドの使用

期日フィールドの日付と、締切日である2014年12月31日の間の日数が返されます。

- 締切日より前の日付を指定した場合は、その日付から締切日までの日数に等しい正の値が返されます。
- 締切日より後の日付を指定した場合は、締切日からその日付が発生するまでの日数に等しい負の値が返されます。

```
AGE(期日, `20141231`)
```

2014年12月31日と**期日**フィールドの日付の間の日数が返されます。結果の絶対値はすぐ上の例と同じですが、戻り値の符号(正または負)が逆になります。

```
AGE(`20141231`, 期日)
```

フィールド同士の日付の比較

支払日フィールドの日付と、**期日**フィールドの日付の間の日数が返されます。

- 期日前の支払日を指定した場合は、期日までの支払いを示す正の値が返されます。
- 期日後の支払日は、支払遅延を示す負の値を返します。

```
AGE(支払日, 期日)
```


支払日フィールドの日付と、**期日**フィールドの日付の間の日数に猶予期間の15日を加算した数値が返されます。

- 期日前、または期日後15日までの支払日を指定した場合は、正の値が返されます。
- 期日後15日を超えている支払日を指定した場合は、猶予期間外の支払遅延を示す負の値が返されます。

```
AGE(<支払日>, <期日>+15)
```

高度な例

期限が過ぎた支払いを抽出する

締切日を2014年12月31日として、請求書(Invoice_Date)の経過日数が180日を超えている各レコードについて、名前、金額、請求日付を抽出します。

```
EXTRACT FIELDS Name Amount 請求日 TO "Overdue" IF AGE(請求日,`20141231`) > 180
```

備考

機能の仕組み

AGE( ) 関数は、2つの日付間の日数を計算します。

AGE( ) の使用に適する場面

AGE( ) を使用できる場面は、2つの日付を比較して期日を過ぎた取引先を特定する場合や、残高の経過日数を分析する場合など、2つの日付の間の経過日数が必要な任意の作業を実行する場合です。

負の戻り値

*日付/日付時刻/文字列*に指定した値が*締切日*として指定された日付より後の日付であるか、オペレーティングシステム日付(*締切日*が指定されていない場合)より後の日付である場合は、負数が返されます。

締切日フィールドの使用

締切日のリテラル日付値を必要とするAGEコマンドとは異なり、AGE( ) 関数では締切日フィールドを使用できます。

例:

```
AGE(支払日, 期日)
```


このようにAGE( ) 関数を使用することは、式で2つの日付フィールドを減算して日付間の差を求めることと同等です。

例：

```
期日 - 支払日
```

パラメーターの詳細

*日付/日付時刻/文字列*または*締切日*に指定された日付時刻フィールドは、フィールド定義で正しく書式を定義してさえいれば、どのような日付書式または日付時刻書式でも使用することができます。

リテラル日付または日付時刻値の指定

*日付/日付時刻/文字列*または*締切日*にリテラルの日付値または日付時刻値を指定する場合は、次の表内の書式に制限されます。また、`20141231` や"20141231" のように、値を逆引用符、一重引用符、または二重引用符で囲む必要があります。

日付または時刻の個々の要素の間に、スラッシュ(/) やコロン(:) のような区切り文字をいっさい使用しないでください。文字の時刻値ではコロンを使用できます。

- **日付時刻値** - 以下の表に示す日付、区切り文字、および時刻の書式を任意に組み合わせて使用することができます。日付を時刻の前に置く必要があり、かつ、2つの間に区切り文字を使用する必要があります。有効な区切り文字は空白スペース1つ、あるいは文字 't' または 'T' です。

形式の例	リテラル値の例
YYYYMMDD	`20141231` "20141231"
YYMMDD	`141231` "141231"
YYYYMMDD hhmmss	`20141231 235959` "20141231 235959"
YYMMDDthhmm	`141231t2359` "141231t2359"
YYYYMMDDThh	`20141231T23` "20141231T23"
YYYYMMDD hhmmss+/-hhmm (UTC オフセット)	`20141231 235959-0500` "20141231 235959-0500"
YYMMDD hhmm+/-hh	`141231 2359+01`

形式の例	リテラル値の例
(UTC オフセット)	"141231 2359+01"
メモ UTC オフセットが設定されているデータのメインの時刻書式で hh を単独で使用しないでください。たとえば、「hh+hhmm」という使い方は避けてください。信頼できない結果になる可能性があります。	

ALLTRIM( ) 関数

入力文字列から先頭と末尾のスペースを除去した文字列を返します。

構文

```
ALLTRIM(文字列)
```

パラメーター

名前	種類	説明
文字列	文字	先頭と末尾のスペースを除去する値。

出力

文字。

例

基本的な例

"Vancouver" が返されます。

```
ALLTRIM("  Vancouver  ")
```

"New York" が返されます。

```
ALLTRIM("  New York  ")
```

高度な例

文字フィールドの連結

ALLTRIM の使用に適する場面は、名前フィールドと姓フィールドなどの文字フィールドを連結した後のフィールドで、連結された値同士の間に複数の空白が入らないようにスペースを削除する場合です。

```
DEFINE FIELD Full_Name COMPUTED ALLTRIM(名前) + " " + ALLTRIM(姓)
```

改行なしスペースの削除

改行なしスペースはALLTRIM( )関数で削除されません。

先頭または末尾の改行なしスペースを削除する必要がある場合は、次の式を使用して演算フィールドを作成します。

```
DEFINE FIELD Description_cleaned COMPUTED ALLTRIM(REPLACE(説明, CHR(160), CHR(32)))
```

REPLACE( )関数がすべての改行なしスペースを標準のスペースに置換してから、ALLTRIM( )がすべての先頭または末尾の標準スペースを削除します。

備考

機能の仕組み

ALLTRIM() 関数は文字列から先頭と末尾のスペースを除去します。文字列内の、先頭と末尾にないスペースは、除去されません。

関連する関数

文字列から先頭のスペースのみを除去したい場合はLTRIM( )関数を、末尾のスペースのみを除去したい場合はTRIM( )関数を使用してください。

ASCII( ) 関数

指定された文字のASCIIコードを返します。

構文

```
ASCII(文字)
```

パラメーター

名前	種類	説明
文字	文字	ASCIIコードの確認対象となる文字。 引用符で囲まれた文字、あるいは複数文字の文字列、フィールド、または式を指定します。複数文字の列を指定した場合、最初の文字だけが評価されます。

出力

数値。

例

基本的な例

65 が返されます。

```
ASCII("A")
```

49 が返されます。

```
ASCII("1")
```

高度な例

タブ文字で始まるレコードの抽出

"Description" フィールドの先頭にタブ文字が入っているレコードを抽出します。タブ文字の ASCII コードは "9" です。

```
EXTRACT RECORD TO "Tab_Entries.acl" IF ASCII(Description) = 9
```

備考

出力不可能文字がないかどうかのテスト

以下の出力不可能文字がないかどうかをテストするには、ASCII( ) を使用します。

- **NULL** - ASCII "0"
- **タブ** - ASCII "9"
- **ライン フィード (LF)** - ASCII "10"
- **キャリッジ リターン(CR)** - ASCII "13"

関連する関数

ASCII( ) は CHR( ) の逆関数です。

AT( ) 関数

文字値における部分文字列の特定の出現の開始位置を示す数値を返します。

構文

AT(*出現番号*, *検索文字列*, *検索されるテキスト*)

パラメーター

名前	種類	説明
出現番号	数値	位置を返す*検索文字列*の出現箇所(インスタンス) たとえば、*検索文字列*の最初の出現箇所の開始位置を返させる場合は、1 を指定します。
検索文字列	文字	*検索されるテキスト*内から検索する部分文字列。この値では大文字と小文字が区別されます。 *検索文字列*が二重引用符を含んでいる場合は、検索文字列の値を一重引用符で囲む必要があります。 `AT(1,'"test"', Description)`
検索されるテキスト	文字	検索される値。 テーブル内の複数のフィールドで検索したい場合は、*検索されるテキスト* パラメーターで 2 つ以上のフィールドを連結することができます。 `AT(1,'"test"', Description+Summary)`

出力

数値。*検索文字列*の指定された出現の開始バイト位置を返します。文字列が検出されなかった場合はゼロ値を返します。

例

基本的な例

出現箇所が見つかった場合

4 が返されます。

```
AT(1, "-", "604-669-4225")
```

8 が返されます。

```
AT(2, "-", "604-669-4225")
```

出現箇所が見つらなかった場合

この値には 3 番目のハイフンがないため、0 が返されます。

```
AT(3, "-", "604-669-4225")
```

この値には 4 番目の小文字の "a" がないので、0 が返されます。

```
AT(4, "a", "Alabama")
```

文字のグループ

5 が返されます。

```
AT(2, "iss", "Mississippi")
```

フィールド内の検索

Invoice_Number(請求書番号) フィールド内の各値における最初のハイフンのバイト位置が返されます。

```
AT(1, "-", Invoice_Number)
```

高度な例

2 つ目のハイフンが 10 番目のバイト位置より後に現れる請求書番号を見つける

AT( ) 関数を使って以下のようなフィルターを作成することにより、テーブル内の請求書番号の一貫性を分析

することができます。請求書番号に2つ以上のハイフンが入っていて2つ目のハイフンが10バイト目の位置より後に現れる、すべてのレコードが、このフィルターにより抽出されます。

```
SET FILTER TO AT(2, "-", Invoice_Number) > 10
```

備考

AT( ) の使用に適する場面

この関数は、文字値における以下の開始位置を取得する場合に使用することができます。

- 部分文字列の開始位置
- 部分文字列の後続の出現箇所の開始位置

フィールド内に同じ部分文字列が複数回出現することを確認したいだけである場合は、OCCURS( ) 関数がより良い代替手段です。詳細については、"OCCURS( ) 関数" ページ 660を参照してください。

*出現番号*が出現箇所の数より大きい場合に、値が返されます。

*出現番号*が*検索されるテキスト*内における部分文字列の実際の出現数より大きい場合には、関数は部分文字列の出現を見つけられないので 0 を返します。

連結されたフィールドと戻り値

複数のフィールド内を検索した場合、見つかったインスタンスに対する戻り値は、指定したすべてのフィールドにおける*検索文字列*の開始位置になります。連結したフィールドは 1 つのフィールドのように扱われます。ただし、ALLTRIM( ) 関数を使用して個々のフィールドからスペースを除去しなければ、各フィールドの先頭と末尾のスペースを含むフィールドとなります。

たとえば、それぞれ幅が 8 バイトである 2 つのフィールドから文字列の最初の出現箇所を検索した場合に、その文字列が 2 番目のフィールドの先頭で見つかったときは、戻り値は 9 になります。

BETWEEN( ) 関数

指定された値が範囲内にあるかどうかを示す論理値を返します。

構文

BETWEEN(*値*, *最小値*, *最大値*)

パラメーター

名前	種類	説明
値	文字 数値 日付時刻	テストするフィールド、式、またはリテラル値。
最小値	文字 数値 日付時刻	範囲の最小値。
最大値	文字 数値 日付時刻	範囲の最大値。

> **メモ**
> T(true) と評価する範囲には、*最小値*と*最大値*が含まれます。
> 文字の範囲については、"SET EXACT の動作" 見開きページを参照してください。

出力

論理。*値*が*最小値*以上かつ*最大値*以下である場合は、T(true) を返します。そうでない場合は F(false) を返します。

例

基本的な例

数値型の入力値

T が返されます。

```
BETWEEN(500,400,700)
```

F が返されます。

```
BETWEEN(100,400,700)
```

文字型の入力値

T が返されます。

```
BETWEEN("B","A","C")
```

文字の比較で大文字と小文字が区別され、小文字 "b" は大文字 "A" と "C" の間には位置しないため、F が返されます。

```
BETWEEN("b","A","C")
```

日付時刻型の入力値

T が返されます。

```
BETWEEN(`141230`,`141229`,`141231`)
```

Login_time フィールドの値のうち、午前 7 時 0 分 0 秒から午前 9 時 0 分 0 秒までのすべての値を指定した場合は 'T' が返され、それ以外の値を指定した場合は 'F' が返されます。

```
BETWEEN(Login_time,`t070000`,`t090000`)
```

SET EXACT の動作

Last_Name フィールドの値のうち、"C" から "K" の文字で始まるすべての値を指定した場合は 'T' が返され、それ以外の値を指定した場合は 'F' が返されます(SET EXACT を OFF にしておく必要があります) 。

```
BETWEEN(Last_Name, "C", "K")
```

Last_Name フィールドの値のうち、"C" から "J" の文字で始まるすべての値を指定した場合は 'T' が返され、それ以外の値を指定した場合は 'F' が返されます(SET EXACT を ON にしておく必要があります) 。単一の文字 "K" を指定した場合には、T が返されます。

```
BETWEEN(Last_Name, "C", "K")
```

フィールドへの入力値

Invoice_Date フィールドの値のうち、2014 年 9 月 30 日から 2014 年 10 月 30 日までのすべての値を指定した場合は 'T' が返され、それ以外の値を指定した場合は 'F' が返されます。

```
BETWEEN(Invoice_Date, `20140930`, `20141030`)
```

請求日 (Invoice_Date) が発注日 (PO_Date) と支払日 (Paid_Date) の間にないすべてのレコードに対しては 'T' が返され、それ以外のレコードに対しては 'F' が返されます。

```
NOT BETWEEN(Invoice_Date, PO_Date, Paid_Date)
```

Invoice_Amount フィールドの値のうち、2014 年 9 月 30 日から 2014 年 10 月 30 日までのすべての値を指定した場合は 'T' が返され、それ以外の値を指定した場合は 'F' が返されます。

```
BETWEEN(Invoice_Amount, 1000, 5000)
```

高度な例

給与の範囲を表示するためのフィルターの作成

次の例は、**Employee_List** サンプルテーブルを開き、給与が 40,000.00 ドル以上、50,000.00 ドル以下の従業員のみが含まれるように、表示されるレコードを制限するフィルターを適用しています。

```
OPEN Employee_List
SET FILTER TO BETWEEN(Salary, 40000.00, 50000.00)
```

備考

サポートされているデータ型

BETWEEN( ) 関数への入力には数値、文字、日付時刻のデータを指定できます。データ型を混在させることはできません。3 つの入力はすべて同じデータ型に属している必要があります。

AND 演算子の代わりにBETWEEN( ) を使用する

AND 演算子を使用する式の代わりにBETWEEN( ) 関数を使用することができます。

例：

```
BETWEEN(Invoice_Amount, 1000, 5000)
```

上記の関数は、次の式と同等です。

```
Invoice_Amount >= 1000 AND Invoice_Amount <= 5000
```

*最小値と最大値*の順序

BETWEEN( ) 関数に指定する*最小値*と*最大値*の順序は重要ではありません。値の大小関係はAnalytics が自動的に識別します。

次の例ではいずれもT が返されます。

```
BETWEEN(2500, 1000, 5000)
```

```
BETWEEN(2500, 5000, 1000)
```

数値型入力値の小数点以下の精度

比較する数値型入力値の小数点以下の精度が異なる場合、比較は高い方の精度に合わせて行われます。

1.23 は 1.23 と等しいため、T が返されます。

```
BETWEEN(1.23, 1.23, 1.25)
```

小数点第 3 位まで考慮されるため、1.23 は 1.234 より小さいと判定されるので、F が返されます。

```
BETWEEN(1.23, 1.234, 1.25)
```

文字データ

大文字と小文字の区別

文字データが使用される場合、BETWEEN( ) 関数は大文字と小文字を区別します。文字を比較する場合、"a" は "A" と同等ではありません。

F が返されます。

```
BETWEEN("B","a","C")
```

大文字と小文字のどちらかに統一されていないデータを操作する場合は、BETWEEN( ) 関数を使用する前にUPPER( ) 関数を使用して、値をすべて大文字に変換することができます。

T が返されます。

```
BETWEEN(UPPER("B"), UPPER("a"), UPPER("C"))
```

部分一致

文字比較では、部分一致がサポートされます。

*値*が*最小値*より大きいと見なされる場合があります。

値 "AB" は*最小値* "ABC" より小さいように見えますが、T が返されます。

```
BETWEEN("AB", "ABC", "Z")
```

*最大値*が*値*より小さいと見なされる場合があります。

値 "ZZ" は*最大値* "Z" より大きいように見えますが、T が返されます。

```
BETWEEN("ZZ", "ABC", "Z")
```

> **メモ**
>
> 文字比較において、短い値は、一致を成す長い値の先頭に現れる必要があります。

部分一致とSET EXACT

部分一致は、Analytics のデフォルト設定である SET EXACT = OFF の場合に有効になります。SET EXACT = ON にすると、部分一致は無効になり、比較値が一致を成すには、正確に一致しなければなりません。

SET EXACT が ON の場合、上記の例はどちらも False になります。

SET EXACT([**正確な文字比較を行う**]オプション) の詳細については、"SET コマンド" ページ 404を参照してください。

SET EXACT の OFF または ON への設定

BETWEEN( ) 関数で [**正確な文字比較を行う**]オプションが使用されないようにしたい場合は、 [**オプション**] ダイアログボックスの [**テーブル**]タブ([**ツール > オプション**]) でこのオプションが選択されていないことを確認してください。

スクリプトを使用している場合は、BETWEEN( ) 関数が現れる前に SET EXACT OFF コマンドを追加します。必要に応じて、SET EXACT ON コマンドによって前の状態に戻すことができます。

日付時刻パラメーター

関数への入力として指定された日付、日付時刻、または時刻フィールドは、フィールド定義で正しく書式を定義してさえいれば、どのような日付書式、日付時刻書式、または時刻書式でも使用することができます。

日付型、日付時刻型、時刻型の入力値を混在させる

BETWEEN( ) 関数の3つの入力値に日付値、日付時刻値、および時刻値を混在させることは禁止されていませんが、これら日付時刻のサブタイプを混在させると、意味のない結果が返される可能性があります。

日付時刻値の日付部分のみに関心があっても、時刻部分はまだ計算の一部を構成しているため、Analyticsは対応するシリアル値を使用して、日付時刻の計算を処理しています。

次の例で考えてみましょう。

2014年12月31日は*最小値*と*最大値*によって指定された範囲に含まれるため、Tが返されます。

```
BETWEEN(`20141231`,`20141229`,`20141231`)
```

2014年12月31日は*最小値*と*最大値*によって指定された範囲に含まれるように見えますが、Fが返されます。

```
BETWEEN(`20141231 120000`,`20141229`,`20141231`)
```

これら2つの式の対応するシリアル値を見ると、どうして2番目の式がfalseと評価されるのかがわかります。

シリアル値である*値*がシリアル値である*最大値*と等しいため、Tが返されます。

```
BETWEEN(42003.000000, 42001.000000, 42003.000000)
```

シリアル値である*値*がシリアル値である*最大値*より大きいため、Fが返されます。

```
BETWEEN(42003.500000, 42001.000000, 42003.000000)
```

2つの日付が同一であっても、シリアル値の`42003.500000`は`42003.000000`より大きいため、範囲外になります。`0.500000`は12:00 PMに相当するシリアル値です。

日付時刻サブタイプ同士を一致させる

混在する日付時刻サブタイプによって発生する可能性のある問題を回避するには、サブタイプ同士を一致させる関数を使用します。

たとえば、次の式では、上記の2番目の式と同じ初期値が使用されていますが、FでなくTが返されます。

```
BETWEEN(CTOD(DATE(`20141231
120000`,"YYYYMMDD"),"YYYYMMDD"),`20141229`,`20141231`)
```

リテラル日付、日付時刻、または時刻値の指定

関数への入力のいずれかにリテラルの日付値、日付時刻値、または時刻値を指定する場合は、次の表内の書式に制限されます。また、`20141231` のように、値を逆引用符で囲む必要があります。

日付または時刻の個々の要素の間に、スラッシュ(/) やコロン(:) のような区切り文字をいっさい使用しないでください。

- **日付時刻値** - 以下の表に示す日付、区切り文字、および時刻の書式を任意に組み合わせて使用することができます。日付を時刻の前に置く必要があり、かつ、2 つの間に区切り文字を使用する必要があります。有効な区切り文字は空白スペース 1 つ、あるいは文字 't' または 'T' です。
- **時刻値** - 24 時間形式で時刻を指定する必要があります。UTC(Coordinated Universal Time: 協定世界時) からのオフセットは、プラス記号 (+) またはマイナス記号 (-) で始める必要があります。

形式の例	リテラル値の例
YYYYMMDD	`20141231`
YYMMDD	`141231`
YYYYMMDD hhmmss	`20141231 235959`
YYMMDDthhmm	`141231t2359`
YYYYMMDDThh	`20141231T23`
YYYYMMDD hhmmss+/-hhmm (UTC オフセット)	`20141231 235959-0500`
YYMMDD hhmm+/-hh (UTC オフセット)	`141231 2359+01`
thhmmss	`t235959`
Thhmm	`T2359`
メモ UTC オフセットが設定されているデータのメインの時刻書式で hh を単独で使用しないでください。たとえば、「hh+hhmm」という使い方は避けてください。信頼できない結果になる可能性があります。	

BINTOSTR( ) 関数

ZONED または EBCDIC 文字データから変換された Unicode 文字データを返します。"Binary to String" の省略形です。

> **メモ:**
>
> この関数は Analytics の Unicode 版に固有です。非 Unicode 版ではサポートされない関数です。

構文

```
BINTOSTR(文字列, 文字列のタイプ)
```

パラメーター

名前	種類	説明
文字列	文字	Unicode 文字エンコードに変換する ZONED 値または EBCDIC 値。
文字列のタイプ	文字	変換元の書式。次のいずれかの値を指定する必要があります。 ◦ "A" - 変換元は ZONED(ASCII) データです。 ◦ "E" - 変換元は EBCDIC データです。

出力

文字。

例

基本的な例

式 ZONED(-6448,4) は値 -6448 を文字型書式 "644Q" に変換していますが、Analytics の Unicode 版では BINTOSTR( ) を使って ZONED( ) の出力を Unicode 文字に変換する必要があります。

Unicode 書式の "644Q" が返されます。

```
BINTOSTR(ZONED(-6448,4), "A")
```

備考

BINTOSTR( ) の使用に適する場面

この関数は、ZONED( ) および EBCDIC( ) 関数からの戻り値を Unicode 値に変換する場合に使用することができます。

> **メモ**
>
> この関数を Analytics の Unicode 版の ZONED( ) および EBCDIC( ) の戻り値に適用しない場合は、エンコードが正確に解釈されないので、戻り値の表示が不正確になります。

BIT( ) 関数

現在のレコードにおける指定されたバイト位置のバイナリ表記を 8 文字の文字列として返します。

構文

```
BIT(バイト位置)
```

パラメーター

名前	種類	説明
バイト位置	数値	バイナリ値として返されるバイト位置。

出力

文字。

例

基本的な例

8 番目のバイトに"1" が含まれる場合に、"00110001" が返されます。

```
BIT(8)
```

9 番目のバイトに"A" が含まれる場合に、"01000001" が返されます。

```
BIT(9)
```

17 番目のバイトに"a" が含まれる場合に、"01100001" が返されます。

```
BIT(17)
```

高度な例

BIT ( ) および SUBSTRING ( ) を使用した値の抽出

バイト 17 に8 つの信用フラグが含まれているとします。

3 番目のビットが1 に設定されたすべての顧客("出荷対象外"の顧客) のレコードを抽出するには、次のように指定します。

```
EXTRACT IF SUBSTRING(BIT(17), 3, 1) = "1"
```

この例では、SUBSTRING( ) 関数を使って、3 番目のビットの値を抽出しています。

備考

機能の仕組み

BIT( ) は、指定されたバイト位置にあるバイトを1と0 で構成される8 文字の列に変換します。

BIT( ) の使用に適する場面

BIT( ) は、バイト中の個々のビットを調べる場合に使用できます。

関連する関数

指定されたバイト位置にある文字を読み出したい場合は、BYTE( ) 関数を使用してください。

BLANKS( ) 関数

指定された数の空白スペースを含んでいる文字列を返します。

構文

```
BLANKS(数)
```

パラメーター

名前	種類	説明
カウント	数値	挿入する空白スペースの数。

出力

文字。

例

基本的な例

"　" が返されます。

```
BLANKS(5)
```

"ABC Corporation" を返します。

```
"ABC" + BLANKS(1) + "Corporation"
```


備考

BLANKS( ) の使用に適する場面

BLANKS( ) 関数を使用できる場面は、フィールドを一致させる場合や、スクリプトで変数を初期化する場合、あるいはフィールドの書式設定や文字列の連結を行うときに空白を挿入する場合です。

BYTE( ) 関数

現在のレコード内で指定されたバイト位置に格納されている文字を返します。

構文

BYTE(*バイト位置*)

パラメーター

名前	種類	説明
バイト位置	数値	文字値として返されるバイト位置。 バイト位置はフィールド定義とは関係なく、レコード内での位置(1 から開始) を参照します。

出力

文字。

例

基本的な例

"1" を値とする ID フィールドで始まるレコードに対して、"1" が返されます。

```
byte(112)
```

高度な例

一貫した書式設定を使用した印刷ファイルまたは PDF の中のレコードを識別する

BYTE( ) を使用して、データファイル内のレコードのうち、特定の文字が特定のバイト位置に存在するレコードを識別することができます。これは、ドキュメント全体を通してデータが一貫した方法で書式設定されている、印刷イメージ(レポート) ファイルまたは Adobe Acrobat(PDF) ファイルで典型的なケースです。

たとえば、バイト位置 113 にピリオドのあるレコードを見つけて抽出するには、次のコマンドを使用します。

```
EXTRACT RECORD IF BYTE(113) = "." TO "Output.fil"
```

備考

BYTE( ) の使用に適する場面

BYTE( ) 関数を使用すると、フィールドを定義しなくても、レコード内の特定の位置にある内容を調べることができます。

EBCDIC データに対する BYTE( ) の使用

EBCDIC データに対してこの関数を使用した場合は、返される値も EBCDIC になります。この値を文字値と比較することはできないかもしれません。

関連する関数

指定されたバイト位置のバイナリ表記を取得したい場合は、BIT( ) 関数を使用してください。

CDOW( ) 関数

指定された日付または日付時刻の曜日を返します。"Character Day of Week" の省略形です。

構文

```
CDOW(日付/日付時刻, 長さ)
```

パラメーター

名前	種類	説明
日付/日付時刻	日付時刻	曜日を返すフィールド、式、またはリテラル値。
長さ	数値	出力文字列の長さを指定する 1 から 9 の間の値。曜日名の省略形を表示するには、より小さな値を指定してください。

出力

文字。

例

基本的な例

2014 年 12 月 31 日が水曜日 (Wednesday) に当たり、*長さ*が 9 のため、"Wednesday" が返されます。

```
CDOW(`20141231`, 9)
```

2014 年 12 月 31 日が水曜日 (Wednesday) に当たり、*長さ*が 9 のため、"Wed" が返されます。

```
CDOW(`20141231 235959`, 3)
```

Invoice_date フィールドの各値に対して曜日の正式名称が返されます。

```
CDOW(Invoice_date, 9)
```

Receipt_timestamp フィールドの各値に対して曜日の省略名称が返されます。

```
CDOW(Receipt_timestamp, 3)
```

高度な例

日付の曜日を識別するフィールドを追加する

CDOW( ) 関数を使用して、日付フィールドの全日付の曜日を識別するための演算フィールドを作成することができます。演算フィールドを作成したら、ビューの日付列の横にそれを追加できます。

```
DEFINE FIELD Name_of_Day COMPUTED CDOW(Trans_Date, 3)
```

フィルターを作成して、週末に発生した取引をテストする

CDOW( ) 関数を使用して、フィルターを作成し、週末に発生した取引を抽出することができます。

```
SET FILTER TO CDOW(Trans_Date, 3) = "Sat" OR CDOW(Trans_Date, 3) = "Sun"
```

備考

パラメーターの詳細

*日付/日付時刻*に指定されたフィールドは、フィールド定義で正しく書式を定義してさえいれば、どのような日付書式または日付時刻書式でも使用することができます。

*長さ*パラメーターが曜日名より短い場合、曜日名は指定された長さに切り詰められます。*長さ*パラメーターが曜日名より長い場合、曜日名は空白スペースで埋められます。

リテラル日付または日付時刻値の指定

*日付/日付時刻*にリテラルの日付値または日付時刻値を指定する場合は、次の表内の書式に制限されます。また、`20141231` のように、値を逆引用符で囲む必要があります。

日付または時刻の個々の要素の間に、スラッシュ(/) やコロン(:) のような区切り文字をいっさい使用しないでください。

- **日付時刻値** - 以下の表に示す日付、区切り文字、および時刻の書式を任意に組み合わせて使用することができます。日付を時刻の前に置く必要があり、かつ、2 つの間に区切り文字を使用する必要があります。有効な区切り文字は空白スペース 1 つ、あるいは文字 't' または 'T' です。
- **時刻値** - 24 時間形式で時刻を指定する必要があります。UTC(Coordinated Universal Time: 協定世界時) からのオフセットは、プラス記号 (+) またはマイナス記号 (-) で始める必要があります。

形式の例	リテラル値の例
YYYYMMDD	`20141231`
YYMMDD	`141231`
YYYYMMDD hhmmss	`20141231 235959`
YYMMDDthhmm	`141231t2359`
YYYYMMDDThh	`20141231T23`
YYYYMMDD hhmmss+/-hhmm (UTC オフセット)	`20141231 235959-0500`
YYMMDD hhmm+/-hh (UTC オフセット)	`141231 2359+01`
メモ UTC オフセットが設定されているデータのメインの時刻書式で hh を単独で使用しないでください。たとえば、「hh+hhmm」という使い方は避けてください。信頼できない結果になる可能性があります。	

関連する関数

曜日を数字(1 ～ 7) として返したい場合は、CDOW( ) でなくDOW( ) を使用してください。

CHR( ) 関数

指定されたASCIIコードに対応する文字を返します。

構文

```
CHR(数値)
```

パラメーター

名前	種類	説明
数値	数値	1 から 255 の間の有効な数式。

出力

文字。

例

基本的な例

"A" が返されます。

```
CHR(65)
```

"1" が返されます。

```
CHR(49)
```

高度な例

通貨フィールドの値に英国のポンド記号(£)を付加する

Invoice_Amount フィールドの金額の前にポンド記号(ASCII コード 163) を付加する演算フィールドを作成し

ます。数値型の **Invoice_Amount** フィールドがまず文字型のフィールドに変換されて、次いで先頭と末尾の空白額が削除されます。

```
DEFINE FIELD Currency_UK COMPUTED CHR(163)+ALLTRIM(STRING(Invoice_Amount, 12))
```

備考

CHR( ) の使用に適する場面

CHR( ) 関数は、キーボードから直接入力できない文字や画面に表示できない文字など、あらゆる ASCII コードに対応する文字を返す場合に使用できます。CHR( ) を使用して、フィールドやレコードの中からこれら特定の文字を見つけることができます。

NUL の参照

ASCII の NUL(ヌル) 文字は Analytics でテキスト修飾子として使用されるため、これに相当する `CHR(0)` を参照すると、予測できない結果をもたらす可能性があります。できれば参照を避けてください。

関連する関数

CHR( ) は ASCII( ) の逆関数です。

CLEAN( ) 関数

文字列内で、最初の無効な文字とそれ以降のすべての文字を空白に置き換えます。

構文

```
CLEAN(文字列 <,追加指定の無効文字>)
```

パラメーター

名前	種類	説明
文字列	文字	デフォルトで設定されている無効文字および追加指定の無効文字を削除する対象となる値
追加指定の無効文字 省略可能	文字	*文字列*から削除する対象となる、デフォルトで設定されている無効な文字以外の無効な文字。追加指定の無効文字は複数、指定できます。 `" ,;\"` タブ文字、ヌル文字、およびキャリッジ リターン/ライン フィード文字はデフォルトの無効な文字であり、自動的に削除されるため、指定する必要はありません。 二重引用符を追加指定の無効文字として指定するには、*追加指定の無効文字*に、一重引用符で囲んだ二重引用符を指定してください。 `'"'`

出力

文字。

例

基本的な例

"ABC　"("ABC" および 4 つの空白) が返されます。

```
CLEAN("ABC%DEF","%")
```

"1234.56　"("1234.56" および 6 つの空白) が返されます。

```
CLEAN("1234.56,111,2", ",")
```

備考

CLEAN( ) の使用に適する場面

この関数は、無効なデータを含むフィールドが正しく印刷されるようにするために使用します。また、この関数を使用すると、フィールドの一部を分離することができます。たとえば、姓と名前の両方が入っている顧客フィールドから姓を分離できます。

無効な一重引用符または二重引用符の指定

無効文字として一重引用符と二重引用符を両方とも指定するには、CLEAN( ) 関数を入れ子にする必要があります。

```
CLEAN(CLEAN(string, ""), """)
```

Automatic CLEAN( )

Analytics スクリプトにSET CLEAN ONを追加することによって、すべての文字フィールドに対して自動的に CLEAN( ) 関数を適用できます。このオプションを使用して、追加指定の無効文字を指定することはできません。

CMOY( ) 関数

指定された日付または日付時刻の月の名前を返します。"Character Month of Year" の省略形です。

構文

```
CMOY(日付/日付時刻, 長さ)
```

パラメーター

名前	種類	説明
日付/日付時刻	日付時刻	月の名前を返すフィールド、式、またはリテラル値。
長さ	数値	出力文字列の長さを指定する 1 から 9 の間の値。月の名前の省略形を表示するには、より小さな値を指定してください。

出力

文字。

例

基本的な例

"December" が返されます。

```
CMOY(`20141231`, 9)
```

"Dec" が返されます。

```
CMOY(`20141231 235959`, 3)
```

Receipt_timestamp フィールドの各値に対して月の省略名称が返されます。

```
CMOY(Receipt_timestamp, 3)
```

Invoice_date フィールドの各値に対して月の正式名称が返されます。

```
CMOY(Invoice_date, 9)
```

Invoice_date フィールドの各値の 15 日後に当たる月の正式名称が返されます。

```
CMOY(Invoice_date + 15, 9)
```

備考

パラメーターの詳細

*日付/日付時刻*に指定されたフィールドは、フィールド定義で正しく書式を定義してさえいれば、どのような日付書式または日付時刻書式でも使用することができます。

*長さ*パラメーターが月の名前より短い場合、月の名前は指定された長さに切り詰められます。*長さ*パラメーターが月の名前より長い場合、月の名前は空白スペースで埋められます。

リテラル日付または日付時刻値の指定

*日付/日付時刻*にリテラルの日付値または日付時刻値を指定する場合は、次の表内の書式に制限されます。また、`20141231` のように、値を逆引用符で囲む必要があります。

日付または時刻の個々の要素の間に、スラッシュ(/) やコロン(:) のような区切り文字をいっさい使用しないでください。

- **日付時刻値** - 以下の表に示す日付、区切り文字、および時刻の書式を任意に組み合わせて使用することができます。日付を時刻の前に置く必要があり、かつ、2 つの間に区切り文字を使用する必要があります。有効な区切り文字は空白スペース 1 つ、あるいは文字 't' または 'T' です。
- **時刻値** - 24 時間形式で時刻を指定する必要があります。UTC(Coordinated Universal Time：協定世界時) からのオフセットは、プラス記号 (+) またはマイナス記号 (-) で始める必要があります。

形式の例	リテラル値の例
YYYYMMDD	`20141231`
YYMMDD	`141231`
YYYYMMDD hhmmss	`20141231 235959`
YYMMDDthhmm	`141231t2359`
YYYYMMDDThh	`20141231T23`
YYYYMMDD hhmmss+/-hhmm	`20141231 235959-0500`

形式の例	リテラル値の例
(UTC オフセット)	
YYMMDD hhmm+/-hh (UTC オフセット)	`141231 2359+01`
メモ UTC オフセットが設定されているデータのメインの時刻書式で hh を単独で使用しないでください。たとえば、「hh+hhmm」という使い方は避けてください。信頼できない結果になる可能性があります。	

関連する関数

月を数字(1 ～ 12) として返したい場合は、CMOY( ) でなくMONTH( ) を使用してください。

COS( ) 関数

ラジアン単位で表された角度のコサインを、小数点以下 15 桁の精度で返します。

構文

```
COS(ラジアン)
```

パラメーター

名前	種類	説明
ラジアン	数値	ラジアン単位の角度。

出力

数値。

例

基本的な例

指定したラジアン数に対して 0.500000000000000 が返されます。

```
COS(1.047197551196598)
```

高度な例

入力値として度を使用する

60 度の余弦である 0.500000000000000 が返されます。

```
COS(60 * PI( )/180)
```

小数点第 3 位までの小数に丸める

60 度の余弦を小数点第 3 位に丸めた 0.500 が返されます。

```
DEC(COS(60 * PI( )/180),3)
```

備考

Mantissa Arc Test の実行

Analytics の 3 つの三角関数 SIN( )、COS( )、および TAN( ) は、ベンフォードの法則に関連する Mantissa Arc Test の実行をサポートします。

度のラジアンへの変換

入力値が度で表されている場合は、PI( ) 関数を使用して角度をラジアンに変換できます。(*度* * PI( )/180) = *ラジアン*となります。必要に応じて、DEC( ) 関数を使用して、戻り値を丸めたり切り捨てたりすることができます。

CTOD( ) 関数

文字または数値の日付値を日付に変換します。また、文字または数値の日付時刻値から日付を抽出し、それを日付として返すこともできます。"Character to Date" の省略形です。

構文

CTOD(*文字/数値* <,*書式*>)

パラメーター

名前	種類	説明
文字/数値	文字 数値	日付に変換する、あるいは日付を抽出するフィールド、式、またはリテラル値。
書式 省略可能	文字	*文字/数値*を日付書式にした値。YYYYMMDD または YYMMDD を除く日付書式を使用した値を指定する場合には、*書式*が必須です。書式の例："DD/MM/YYYY" **メモ** *書式*パラメーターを必要とする日付時刻値で CTOD 関数を使用する場合は、書式の日付部分のみを指定してください。時刻部分を指定してはいけません。例： CTODT("31/12/2014 23:59:59", "DD/MM/YYYY hh:mm:ss") 時刻部分を指定すると、結果が表示されなくなります。

出力

日付時刻。日付値は、現在 Analytics に設定されている日付の表示書式を使用して出力されます。

例

基本的な例

文字リテラルの入力値

`20141231` に対して現在のAnalytics日付表示書式であるDD MMM YYYYを適用した、31 Dec 2014が返されます。

```
CTOD("20141231")
```

```
CTOD("31/12/2014", "DD/MM/YYYY")
```

```
CTOD("20141231 235959")
```

数値リテラルの入力値

`20141231` に対して現在のAnalytics日付表示書式であるDD MMM YYYYを適用した、31 Dec 2014が返されます。

```
CTOD(20141231)
```

```
CTOD(31122014, "DDMMYYYY")
```

```
CTOD(20141231.235959)
```

フィールドへの文字型入力値

指定した文字フィールドの各値が現在のAnalytics日付表示形式を使用した日付として返されます。

```
CTOD(Invoice_date, "DD/MM/YYYY")
```

```
CTOD(Receipt_timestamp)
```

フィールドへの数値型入力値

指定した数値フィールドの各値が現在のAnalytics日付表示形式を使用した日付として返されます。

```
CTOD(Due_date, "DDMMYYYY")
```

```
CTOD(Payment_timestamp)
```

高度な例

文字フィールドまたは数値フィールドを日付と比較する

日付を表す値が格納されている文字フィールドまたは数値フィールドに対して日付を比較するには、CTOD( ) 関数を使用します。

下のフィルターでは、次の 2 つの値が比較されています。

- 書式 DDMMYYYY で数値として日付を格納する数値型の **Due_date** フィールド
- リテラル日付値、2014 年 7 月 1 日

```
SET FILTER TO CTOD(Due_date, "DDMMYYYY") < `20140701`
```

備考

必須の日付書式

日付値または日付時刻値を含んでいる文字フィールドおよび数値フィールドは、次の表内の書式と一致している必要があります。日付時刻値は、そのデータ型に有効な日付、区切り文字、および時刻の書式を任意に組み合わせて使用することができます。日付を時刻の前に置く必要があり、かつ、2 つの間に区切り文字を入れる必要があります。

日付、または日付時刻値の日付部分は、Analytics でサポートされており、そのデータ型に有効であれば、どのような日付書式でも使用することができます。ただし、YYYYMMDD と YYMMDD 以外の書式は、*書式*で正しく定義されてさえいれば使用できます。

日付書式	区切り文字形式	時刻書式
文字フィールド		
YYYYMMDD	単一の空白スペース	hhmmss hh:mm:ss
YYMMDD	文字 't'	hhmm hh:mm
*書式*で定義されている場合は、データ型に有効な、Analytics がサポートする任意の日付書式	文字 'T'	hh

日付書式	区切り文字形式	時刻書式
		+/-hhmm +/-hh:mm (UTC オフセット)
		+/-hh (UTC オフセット)

日付書式	区切り文字形式	時刻書式
		メモ UT-C オフセットが設定されているデータのメインの時刻書式で hh を単独で使用しないでください。たとえば、「h-h+-hh-m-m」という使い方は避けてください。信頼でき

日付書式	区切り文字形式	時刻書式
数値フィールド		
YYYYMMDD	小数点	hhmmss
YYMMDD		hhmm
書式で定義されている場合は、データ型に有効な、Analytics がサポートする任意の日付書式		hh

他の日付時刻変換関数

文字または数値から日付時刻への変換

関数	説明
CTODT( )	文字または数値の日付時刻値を日付時刻に変換します。"Character to Datetime" の省略形です。
CTOT( )	文字または数値の時刻値を時刻に変換します。また、文字または数値の日付時刻値から時刻を抽出し、それを時刻として返すこともできます。"Character to Time" の省略形です。

日付時刻から文字への変換

関数	説明
DATE( )	指定された日付または日付時刻から日付を抽出し、それを文字データとして返します。また、現在のオペレーティング システム日付を返すこともできます。
DATETIME( )	日付時刻を文字列に変換します。また、現在のオペレーティング システムの日付時刻を返すこともできます。
TIME( )	指定された時刻または日付時刻から時刻を抽出し、それを文字データとして返します。また、現在のオペレーティング システムの時刻を返すこともできます。

シリアルから日付時刻への変換

関数	説明
STOD( )	シリアル日付、つまり、整数で表される日付を日付値に変換します。"Serial to Date" の省略形です。
STODT( )	シリアル日付時刻、つまり、整数部分と24 時間の小数部分で表される日付時刻を日付時刻値に

関数	説明
	変換します。"Serial to Datetime" の省略形です。
STOT( )	シリアル時刻、つまり、24 時間を 1 として、24 時間が小数部分で表される時刻を時刻値に変換します。"Serial to Time" の省略形です。

CTODT( ) 関数

文字または数値の日付時刻値を日付時刻に変換します。"Character to Datetime" の省略形です。

構文

```
CTODT(文字/数値 <,書式>)
```

パラメーター

名前	種類	説明
文字/数値	文字 数値	日付時刻に変換するフィールド、式、またはリテラル値。
書式 省略可能	文字	*文字/数値*を日付書式にした値。値の日付部分に対し、YYYYMMDD または YYMMDD を除く日付書式を使用した値を指定する場合には、*書式*が必須です。書式の例："DD/MM/YYYY"

出力

日付時刻。日付時刻値は、現在 Analytics に設定されている日付と時刻の表示書式を使用して出力されます。

例

基本的な例

文字リテラルの入力値

`20141231t235959` が返されます。これは、現在の Analytics 日付時刻表示書式である DD MMM YYYY hh:mm:ss を適用した 31 Dec 2014 23:59:59 として表示されます。

```
CTODT("20141231 235959")
```

```
CTODT("31/12/2014 23:59:59", "DD/MM/YYYY hh:mm:ss")
```

数値リテラルの入力値

`20141231t235959` が返されます。これは、現在の Analytics 日付時刻表示書式である DD MMM YYYY hh:mm:ss を適用した 31 Dec 2014 23:59:59 として表示されます。

```
CTODT(20141231.235959)
```

```
CTODT(31122014.235959, "DDMMYYYY.hhmmss")
```

フィールドへの文字型入力値

Receipt_timestamp フィールドの各値が現在の Analytics の日付表示書式を適用した日付時刻として返されます。

```
CTODT(Receipt_timestamp, "DD/MM/YYYY hh:mm:ss")
```

フィールドへの数値型入力値

数値フィールド、**Payment_timestamp** の各値が、現在の Analytics の日付表示書式を適用した日付時刻として返されます。

```
CTODT(Receipt_timestamp, "DD/MM/YYYY hh:mm:ss")
```

高度な例

文字フィールドまたは数値フィールドを日付時刻と比較する

日付時刻を表す値が格納されている文字フィールドまたは数値フィールドに対して日付時刻を比較するには、CTODT( ) 関数を使用します。

下のフィルターでは、次の 2 つの値が比較されています。

- 日付時刻を書式 DD/MM/YYYY hh:mm:ss の文字データとして格納する文字フィールド、**Receipt_timestamp**
- リテラル日付時刻値、2014 年 7 月 1 日 13:30:00

```
SET FILTER TO CTODT(Receipt_timestamp, "DD/MM/YYYY hh:mm:ss") < '20140701t133000'
```

備考

必須の日付時刻書式

日付時刻値を含んでいる文字フィールドおよび数値フィールドは、次の表内の書式と一致している必要があります。日付時刻値は、そのデータ型に有効な日付、区切り文字、および時刻の書式を任意に組み合わせて使用することができます。日付を時刻の前に置く必要があり、かつ、2つの間に区切り文字を入れる必要があります。

値の日付部分は、Analyticsでサポートされており、そのデータ型に有効であれば、どのような日付書式でも使用することができます。ただし、YYYYMMDDとYYMMDD以外の書式は、*書式*で正しく定義されてさえいれば使用できます。*書式*を使用する場合は、時刻書式も指定する必要があり、次の表に示されている時刻書式のいずれかである必要があります。

Analyticsは、日付時刻値の日付部分と時刻部分の間の区切り文字を自動的に認識するため、*書式*に区切り文字を指定する必要はありません。指定したい場合は、区切り文字を指定してもかまいません。

日付書式	区切り文字形式	時刻書式
文字フィールド		
YYYYMMDD	単一の空白スペース	hhmmss hh:mm:ss
YYMMDD	文字 't'	hhmm hh:mm
*書式*で定義されている場合は、データ型に有効な、Analyticsがサポートする任意の日付書式	文字 'T'	hh
		+/-hhmm +/-hh:mm (UTC オフセット)
		+/-hh (UTC オフセット)

日付書式	区切り文字形式	時刻書式

メモ

UT-C オフセットが設定されているデータのメインの時刻書式で hh を単独で使用しないでください。たとえば、「h-h+-hh-m-m」という使い方は避けてくださ

日付書式	区切り文字形式	時刻書式
数値フィールド		
YYYYMMDD	小数点	hhmmss
YYMMDD		hhmm
書式で定義されている場合は、データ型に有効な、Analytics がサポートする任意の日付書式		hh

他の日付時刻変換関数

文字または数値から日付時刻への変換

関数	説明
CTOD( )	文字または数値の日付値を日付に変換します。また、文字または数値の日付時刻値から日付を抽出し、それを日付として返すこともできます。"Character to Date" の省略形です。
CTOT( )	文字または数値の時刻値を時刻に変換します。また、文字または数値の日付時刻値から時刻を抽出し、それを時刻として返すこともできます。"Character to Time" の省略形です。

日付時刻から文字への変換

関数	説明
DATE( )	指定された日付または日付時刻から日付を抽出し、それを文字データとして返します。また、現在のオペレーティング システム日付を返すこともできます。
DATETIME( )	日付時刻を文字列に変換します。また、現在のオペレーティング システムの日付時刻を返すこともできます。
TIME( )	指定された時刻または日付時刻から時刻を抽出し、それを文字データとして返します。また、現在のオペレーティング システムの時刻を返すこともできます。

シリアルから日付時刻への変換

関数	説明
STOD( )	シリアル日付、つまり、整数で表される日付を日付値に変換します。"Serial to Date" の省略形です。
STODT( )	シリアル日付時刻、つまり、整数部分と24 時間の小数部分で表される日付時刻を日付時刻値に

関数	説明
	変換します。"Serial to Datetime" の省略形です。
STOT( )	シリアル時刻、つまり、24 時間を 1 として、24 時間が小数部分で表される時刻を時刻値に変換します。"Serial to Time" の省略形です。

CTOT( ) 関数

文字または数値の時刻値を時刻に変換します。また、文字または数値の日付時刻値から時刻を抽出し、それを時刻として返すこともできます。"Character to Time" の省略形です。

構文

CTOT(*文字/数値*)

パラメーター

名前	種類	説明
文字/数値	文字 数値	時刻に変換する、あるいは時刻を抽出するフィールド、式、またはリテラル値。

出力

日付時刻。時刻値は、現在 Analytics に設定されている時刻の表示書式を使用して出力されます。

例

基本的な例

文字リテラルの入力値

`t235959` が返されます。これは、現在の Analytics 時刻表示書式である hh:mm:ss を適用した 23:59:59 として表示されます。

```
CTOT("t235959")
```

```
CTOT("23:59:59")
```

```
CTOT("20141231 235959")
```

数値リテラルの入力値

`t235959` が返されます。これは、現在の Analytics 時刻表示書式である hh:mm:ss を適用した 23:59:59 として表示されます。

```
CTOT(.235959)
```

```
CTOT(0.235959)
```

```
CTOT(20141231.235959)
```

フィールドへの文字型入力値

文字フィールド **Login_time** の各値が、現在の Analytics の時刻表示書式を適用した時刻として返されます。

```
CTOT(Login_time)
```

フィールドへの数値型入力値

数値フィールド、**Payment_timestamp** の各値が、現在の Analytics の時刻表示書式を適用した、日付部分なしの時刻として返されます。

```
CTOT(Payment_datetime)
```

高度な例

文字フィールドまたは数値フィールドを時刻と比較する

時刻を表す値が格納されている文字フィールドまたは数値フィールドに対して時刻を比較するには、CTOT( ) 関数を使用します。

下のフィルターでは、次の 2 つの値が比較されています。

- 時刻を数値データとして格納した数値型の **Login_time** フィールド
- リテラル時刻値 09:30:00

```
SET FILTER TO CTOT(Login_time) > `t093000`
```

備考

必須の日付時刻書式

時刻値または日付時刻値を含んでいる文字フィールドおよび数値フィールドは、次の表内の書式と一致している必要があります。

時刻値は、区切り文字と時刻の書式を任意に組み合わせて使用することができます。関数が正しく動作するためには、時刻値の前に区切り文字を置くか、あるいは時刻の要素の間にコロンを入れる必要があります。

日付時刻値は、そのデータ型に有効な日付、区切り文字、および時刻の書式を任意に組み合わせて使用することができます。日付を時刻の前に置く必要があり、かつ、2つの間に区切り文字を入れる必要があります。

文字または数値の日付値を日付に変換する場合、あるいは、文字または数値の日付時刻値から日付を抽出し、それを日付として返す場合は、CTOD( ) 関数を使用してください。

文字または数値の日付時刻値を日付時刻に変換する場合は、CTODT( ) 関数を使用してください。

日付書式	区切り文字形式	時刻書式
文字フィールド		
YYYYMMDD	単一の空白スペース	hhmmss hh:mm:ss
YYMMDD	文字 't'	hhmm hh:mm
	文字 'T'	hh
		+/-hhmm +/-hh:mm (UTC オフセット)
		+/-hh (UTC オフセット)
		メモ: UTC オフセットが設定されているデータのメインの時刻書式で hh を単独で使用しないでください。たとえば、「hh+hhmm」という使い方は避けてください。信頼できない結果になる可能性があります。)

日付書式	区切り文字形式	時刻書式
数値フィールド		
YYYYMMDD	小数点	hhmmss
YYMMDD		hhmm
		hh

他の日付時刻変換関数

文字または数値から日付時刻への変換

関数	説明
CTOD( )	文字または数値の日付値を日付に変換します。また、文字または数値の日付時刻値から日付を抽出し、それを日付として返すこともできます。"Character to Date" の省略形です。
CTODT( )	文字または数値の日付時刻値を日付時刻に変換します。"Character to Datetime" の省略形です。

日付時刻から文字への変換

関数	説明
DATE( )	指定された日付または日付時刻から日付を抽出し、それを文字データとして返します。また、現在のオペレーティング システム日付を返すこともできます。
DATETIME( )	日付時刻を文字列に変換します。また、現在のオペレーティング システムの日付時刻を返すこともできます。
TIME( )	指定された時刻または日付時刻から時刻を抽出し、それを文字データとして返します。また、現在のオペレーティング システムの時刻を返すこともできます。

シリアルから日付時刻への変換

関数	説明
STOD( )	シリアル日付、つまり、整数で表される日付を日付値に変換します。"Serial to Date" の省略形です。
STODT( )	シリアル日付時刻、つまり、整数部分と24 時間の小数部分で表される日付時刻を日付時刻値に変換します。"Serial to Datetime" の省略形です。
STOT( )	シリアル時刻、つまり、24 時間を 1 として、24 時間が小数部分で表される時刻を時刻値に変換しま

関数	説明
	す。"Serial to Time" の省略形です。

CUMIPMT( ) 関数

期間の範囲中に貸付金に対して支払う累積利息を返します。

構文

CUMIPMT(*利率*, *期間*, *金額*, *開始期*, *終了期* <,*種類*>)

パラメーター

名前	種類	説明
利率	数値	1 期あたりの利率。
期間	数値	支払期間の総数。
金額	数値	貸付金の元金。
開始期	数値	計算対象となる最初の期。 *開始期*には 0 は指定できません。
終了期	数値	計算対象となる終了の期。 *終了期*には、支払期間の総数を超える数を指定することはできません。
種類 省略可能	数値	支払いのタイミング: ◦ 0 - 期末払い ◦ 1 - 期首払い 支払いのタイミングが省略された場合は、デフォルト値の 0 が使用されます。

メモ

利率、*期間*を指定する際には、**1 期あたりの**利率を確実に指定するため、整合性のある期間を使用する必要があります。

例:

- 年利 5% の 2 年間の貸付金または投資に対して月*払い*する場合は、*利率*に 0.05/12、期間に 2 * 12 を指定します。
- 同じ貸付金または投資に対して年払いする場合は、*利率*に 0.05、*期間*に 2 を指定します。

出力

数値。

例

基本的な例

$275,000 の貸付金を年利 6.5 パーセントで 25 年間にわたって返済する場合 (支払期日は月末です) の、2 年目に支払う利息の合計額、($) 17437.23 が返されます。

```
CUMIPMT(0.065/12, 12*25, 275000, 13, 24, 0)
```

上記の貸付金について、初年度に支払う利息の合計額、($) 17741.31 が返されます。

```
CUMIPMT(0.065/12, 12*25, 275000, 1, 12, 0)
```

備考

関連する関数

CUMPRINC( ) 関数は CUMIPMT( ) 関数に対して補完的役割を果たします。

CUMIPMT( ) 関数は単一の期間に支払われた利息を計算します。

CUMPRINC( ) 関数

期間の範囲中に貸付金に対して支払う累積元金を返します。

構文

CUMPRINC(*利率*, *期間*, *金額*, *開始期*, *終了期* <,*種類*>)

パラメーター

名前	種類	説明
利率	数値	1 期あたりの利率。
期間	数値	支払期間の総数。
金額	数値	貸付金の元金。
開始期	数値	計算対象となる最初の期。 *開始期*には 0 は指定できません。
終了期	数値	計算対象となる終了の期。 *終了期*には、支払期間の総数を超える数を指定することはできません。
種類 省略可能	数値	支払いのタイミング: ◦ 0 - 期末払い ◦ 1 - 期首払い 支払いのタイミングが省略された場合は、デフォルト値の 0 が使用されます。

メモ

利率、*期間*を指定する際には、**1 期あたりの**利率を確実に指定するため、整合性のある期間を使用する必要があります。

例:

- 年利 5% の 2 年間の貸付金または投資に対して月*払い*する場合は、*利率*に 0.05/12、期間に 2 * 12 を指定します。
- 同じ貸付金または投資に対して年払いする場合は、*利率*に 0.05、*期間*に 2 を指定します。

出力

数値。

例

基本的な例

$275,000 の貸付金を年利 6.5 パーセントで 25 年間にわたって返済する場合 (支払期日は月末です) の、2 年目に支払う元金の合計額、$4844.61 が返されます。

```
CUMPRINC(0.065 / 12, 12*25, 275000, 13, 24)
```

上記の貸付金の最初の月に支払う元金の額、$367.24 が返されます。

```
CUMPRINC(0.065/12, 12*25, 275000, 1, 1, 0)
```

備考

関連する関数

CUMIPMT( ) 関数は CUMPRINC( ) 関数に対して補完的役割を果たします。

CUMPPMT( ) 関数は単一の期間に支払われた元金を計算します。

DATE( ) 関数

指定された日付または日付時刻から日付を抽出し、それを文字データとして返します。また、現在のオペレーティングシステム日付を返すこともできます。

構文

DATE(<*日付/日付時刻*> <,*書式*>)

パラメーター

名前	種類	説明
日付/日付時刻 省略可能	日付時刻	日付を抽出するフィールド、式、またはリテラル値。これを省略した場合は、現在のオペレーティングシステム日付が返されます。
書式 省略可能	文字	出力文字列に適用する書式。例: "DD/MM/YYYY" このパラメーターを省略した場合は、現在の Analytics の日付表示書式が使用されます。*日付/日付時刻*を省略した場合に、*書式*を指定することはできません。

出力

文字。

例

基本的な例

"20141231" に対して現在の Analytics 日付表示書式を適用した値が返されます。

```
DATE(`20141231 235959`)
```

"31-Dec-2014" が返されます。

```
DATE(`20141231 235959`, "DD-MMM-YYYY")
```

現在のオペレーティングシステム日付が、現在のAnalyticsの日付表示書式を適用した文字列として返されます。

```
DATE()
```

Receipt_timestamp フィールドの各値が、現在のAnalyticsの日付表示書式を適用した文字列として返されます。

```
DATE(Receipt_timestamp)
```

Receipt_timestamp フィールドの各値に対して、指定した日付表示書式を適用した文字列が返されます。

```
DATE(Receipt_timestamp, "DD/MM/YYYY")
```

備考

出力文字列の長さ

出力文字列の長さは常に12文字です。指定した出力*書式*、またはAnalyticsの日付の表示書式が12文字より短い場合は、出力文字列の末尾に空白が埋められます。

パラメーターの詳細

*日付/日付時刻*に指定されたフィールドは、フィールド定義で正しく書式を定義してさえいれば、どのような日付書式または日付時刻書式でも使用することができます。

*書式*を使用して出力文字列の表示方法を制御する場合は、サポートされている任意のAnalytics日付表示書式を使用することができます。例：

- DD/MM/YYYY
- MM-DD-YY
- DD MMM YYYY

*書式*は、一重引用符または二重引用符を使用して指定する必要があります。たとえば、`"DD MMM YYYY"` のように指定します。

リテラル日付または日付時刻値の指定

*日付/日付時刻*にリテラルの日付値または日付時刻値を指定する場合は、次の表内の書式に制限されます。また、`` `20141231` `` のように、値を逆引用符で囲む必要があります。

日付または時刻の個々の要素の間に、スラッシュ(/) やコロン(:) のような区切り文字をいっさい使用しないでください。

- **日付時刻値** - 以下の表に示す日付、区切り文字、および時刻の書式を任意に組み合わせて使用することができます。日付を時刻の前に置く必要があり、かつ、2つの間に区切り文字を使用する必要があります。有効な区切り文字は空白スペース1つ、あるいは文字 't' または 'T' です。
- **時刻値** - 24時間形式で時刻を指定する必要があります。UTC(Coordinated Universal Time: 協定世界時) からのオフセットは、プラス記号 (+) またはマイナス記号 (-) で始める必要があります。

形式の例	リテラル値の例
YYYYMMDD	`20141231`
YYMMDD	`141231`
YYYYMMDD hhmmss	`20141231 235959`
YYMMDDthhmm	`141231t2359`
YYYYMMDDThh	`20141231T23`
YYYYMMDD hhmmss+/-hhmm (UTC オフセット)	`20141231 235959-0500`
YYMMDD hhmm+/-hh (UTC オフセット)	`141231 2359+01`
メモ UTC オフセットが設定されているデータのメインの時刻書式で hh を単独で使用しないでください。たとえば、「hh+hhmm」という使い方は避けてください。信頼できない結果になる可能性があります。	

関連する関数

日付時刻値として現在のオペレーティングシステム日付を返すには、DATE( ) の代わりにTODAY( ) を使用してください。

他の日付時刻変換関数

日付時刻から文字への変換

関数	説明
DATETIME( )	日付時刻を文字列に変換します。また、現在のオペレーティングシステムの日付時刻を返すこともできます。

関数	説明
TIME( )	指定された時刻または日付時刻から時刻を抽出し、それを文字データとして返します。また、現在のオペレーティング システムの時刻を返すこともできます。

文字または数値から日付時刻への変換

関数	説明
CTOD( )	文字または数値の日付値を日付に変換します。また、文字または数値の日付時刻値から日付を抽出し、それを日付として返すこともできます。"Character to Date" の省略形です。
CTODT( )	文字または数値の日付時刻値を日付時刻に変換します。"Character to Datetime" の省略形です。
CTOT( )	文字または数値の時刻値を時刻に変換します。また、文字または数値の日付時刻値から時刻を抽出し、それを時刻として返すこともできます。"Character to Time" の省略形です。

シリアルから日付時刻への変換

関数	説明
STOD( )	シリアル日付、つまり、整数で表される日付を日付値に変換します。"Serial to Date" の省略形です。
STODT( )	シリアル日付時刻、つまり、整数部分と24 時間の小数部分で表される日付時刻を日付時刻値に変換します。"Serial to Datetime" の省略形です。
STOT( )	シリアル時刻、つまり、24 時間を 1 として、24 時間が小数部分で表される時刻を時刻値に変換します。"Serial to Time" の省略形です。

DATETIME( ) 関数

日付時刻を文字列に変換します。また、現在のオペレーティングシステムの日付時刻を返すこともできます。

構文

```
DATETIME(<日付時刻> <,書式>)
```

パラメーター

名前	種類	説明
日付時刻 省略可能	日付時刻	変換するフィールド、式、またはリテラル値。これを省略した場合は、現在のオペレーティングシステム日付が返されます。
書式 省略可能	文字	出力文字列に適用する書式。例：`"DD/MM/YYYY"` このパラメーターを省略した場合は、現在の Analytics の日付表示書式が使用されます。*日付/日付時刻*を省略した場合に、*書式*を指定することはできません。

出力

文字。

例

基本的な例

リテラル日付時刻の入力値

"20141231 235959" に対して現在の Analytics 日付時刻表示書式を適用した値が返されます。

```
DATETIME(`20141231 235959`)
```

"31-Dec-2014 11:59 P" が返されます。

```
DATETIME(`20141231 235959`, "DD-MMM-YYYY hh:mm A")
```

現在のオペレーティングシステム日付時刻に対して、現在のAnalyticsの日付時刻表示書式を適用した文字列が、返されます。

```
DATETIME()
```

フィールドへの入力値

Receipt_timestamp フィールドの各値が、現在のAnalyticsの日付時刻表示書式を適用した文字列として返されます。

```
DATETIME(Receipt_timestamp)
```

Receipt_timestamp フィールドの各値に対して、指定した日付時刻表示書式を適用した文字列が返されます。

```
DATETIME(Receipt_timestamp, "DD/MM/YYYY hh:mm:ss")
```

備考

出力文字列の長さ

出力文字列の長さは常に27文字です。指定した出力*書式*、またはAnalyticsの日付と時刻の表示書式が27文字より短い場合は、出力文字列の末尾に空白が埋められます。

パラメーターの詳細

*日付時刻*に指定されたフィールドは、フィールド定義で正しく書式を定義してさえいれば、どのような日付時刻書式でも使用することができます。

*書式*を使用して出力文字列の表示方法を制御する場合は、次の表内の書式に制限されます。

- 日付、時刻、およびAM/PMの書式を任意に組み合わせて使用することができます。
- 日付は時刻の前に置く必要があります。Analyticsは出力文字列の区切り文字として単一のスペースを自動的に使用するので、両者の間に区切り文字を置く必要はありません。
- AM/PM書式は省略可能で、最後に置きます。
- *書式*は、一重引用符または二重引用符を使用して指定する必要があります。

  例：`"DD-MMM-YYYY hh:mm:ss AM"`

日付書式	時刻書式	AM/PM書式	例
サポートされるすべてのAnalytics日付表示書式	hh:mm:ss	なし 24時間制	"DD/MM/YYYY hh:mm:ss"
	hhmmss	AMまたはPM	"MMDDYY hhmmss PM"

日付書式	時刻書式	AM/PM 書式	例
		12 時間制	
	hh:mm	A または P 12 時間制	"DD-MMM-YYYY hh:mm A"
	hhmm		
	hh		

リテラル日付時刻値の指定

*日付時刻*にリテラルの日付時刻値を指定する場合は、次の表内の書式に制限されます。また、`20141231 235959` のように、値を逆引用符で囲む必要があります。

日付または時刻の個々の要素の間に、スラッシュ(/) やコロン(:) のような区切り文字をいっさい使用しないでください。

- **日付時刻値** - 以下の表に示す日付、区切り文字、および時刻の書式を任意に組み合わせて使用することができます。日付を時刻の前に置く必要があり、かつ、2 つの間に区切り文字を使用する必要があります。有効な区切り文字は空白スペース 1 つ、あるいは文字 't' または 'T' です。
- **時刻値** - 24 時間形式で時刻を指定する必要があります。UTC(Coordinated Universal Time: 協定世界時) からのオフセットは、プラス記号(+) またはマイナス記号(-) で始める必要があります。

形式の例	リテラル値の例
YYYYMMDD hhmmss	`20141231 235959`
YYMMDDthhmm	`141231t2359`
YYYYMMDDThh	`20141231T23`
YYYYMMDD hhmmss+/-hhmm (UTC オフセット)	`20141231 235959-0500`
YYMMDD hhmm+/-hh (UTC オフセット)	`141231 2359+01`
メモ UTC オフセットが設定されているデータのメインの時刻書式で hh を単独で使用しないでください。たとえば、「hh+hhmm」という使い方は避けてください。信頼できない結果になる可能性があります。	

他の日付時刻変換関数

日付時刻から文字への変換

関数	説明
DATE( )	指定された日付または日付時刻から日付を抽出し、それを文字データとして返します。また、現在のオペレーティング システム日付を返すこともできます。
TIME( )	指定された時刻または日付時刻から時刻を抽出し、それを文字データとして返します。また、現在のオペレーティング システムの時刻を返すこともできます。

文字または数値から日付時刻への変換

関数	説明
CTOD( )	文字または数値の日付値を日付に変換します。また、文字または数値の日付時刻値から日付を抽出し、それを日付として返すこともできます。"Character to Date" の省略形です。
CTODT( )	文字または数値の日付時刻値を日付時刻に変換します。"Character to Datetime" の省略形です。
CTOT( )	文字または数値の時刻値を時刻に変換します。また、文字または数値の日付時刻値から時刻を抽出し、それを時刻として返すこともできます。"Character to Time" の省略形です。

シリアルから日付時刻への変換

関数	説明
STOD( )	シリアル日付、つまり、整数で表される日付を日付値に変換します。"Serial to Date" の省略形です。
STODT( )	シリアル日付時刻、つまり、整数部分と24 時間の小数部分で表される日付時刻を日付時刻値に変換します。"Serial to Datetime" の省略形です。
STOT( )	シリアル時刻、つまり、24 時間を 1 として、24 時間が小数部分で表される時刻を時刻値に変換します。"Serial to Time" の省略形です。

DAY( ) 関数

指定された日付または日付時刻から日にちを抽出し、それを数値(1 ～ 31)として返します。

構文

```
DAY(日付/日付時刻)
```

パラメーター

名前	種類	説明
日付/日付時刻	日付時刻	日にちを抽出するフィールド、式、またはリテラル値。

出力

数値。

例

基本的な例

31 が返されます。

```
DAY(`20141231`)
```

```
DAY(`20141231 235959`)
```

Invoice_date フィールドの各値に対して日付が返されます。

```
DAY(Invoice_date)
```

備考

パラメーターの詳細

*日付/日付時刻*に指定されたフィールドは、フィールド定義で正しく書式を定義してさえいれば、どのような日付書式または日付時刻書式でも使用することができます。

リテラル日付または日付時刻値の指定

*日付/日付時刻*にリテラルの日付値または日付時刻値を指定する場合は、次の表内の書式に制限されます。また、`20141231` のように、値を逆引用符で囲む必要があります。

日付または時刻の個々の要素の間に、スラッシュ(/) やコロン(:) のような区切り文字をいっさい使用しないでください。

- **日付時刻値** - 以下の表に示す日付、区切り文字、および時刻の書式を任意に組み合わせて使用することができます。日付を時刻の前に置く必要があり、かつ、2 つの間に区切り文字を使用する必要があります。有効な区切り文字は空白スペース 1 つ、あるいは文字 't' または 'T' です。
- **時刻値** - 24 時間形式で時刻を指定する必要があります。UTC(Coordinated Universal Time: 協定世界時) からのオフセットは、プラス記号(+) またはマイナス記号(-) で始める必要があります。

形式の例	リテラル値の例
YYYYMMDD	`20141231`
YYMMDD	`141231`
YYYYMMDD hhmmss	`20141231 235959`
YYMMDDthhmm	`141231t2359`
YYYYMMDDThh	`20141231T23`
YYYYMMDD hhmmss+/-hhmm (UTC オフセット)	`20141231 235959-0500`
YYMMDD hhmm+/-hh (UTC オフセット)	`141231 2359+01`
メモ UTC オフセットが設定されているデータのメインの時刻書式で hh を単独で使用しないでください。たとえば、「hh+hhmm」という使い方は避けてください。信頼できない結果になる可能性があります。	

関連する関数

以下の各値を返させたい場合は、それぞれ該当する関数を使用します。

- 曜日を数字(1 ～ 7)として返させる場合:DAY( ) でなくDOW( ) を使用
- 曜日の名前を返させる場合:DAY( ) でなくCDOW( ) を使用

DBYTE( ) 関数

レコード内の指定されたバイト位置にある Unicode 文字を返します。

> **メモ**
>
> この関数は Analytics の Unicode 版に固有です。非 Unicode 版ではサポートされない関数です。

構文

```
DBYTE(バイト位置)
```

パラメーター

名前	種類	説明
バイト位置	数値	文字値として返されるバイト位置。 意味のある値を返すために、2 バイト文字の開始場所を指定する必要があります。これは単に、*バイト位置*パラメーターには奇数を指定してくださいということです。

出力

文字。

例

基本的な例

次の例は、11 文字 (22 バイト) の値、**美丽 10072DOE**を含んでいる Unicode 値に適用されたときのこの関数の動作を示しています。

"丽" が返されます。

```
DBYTE(3)
```

"D" が返されます。

```
DBYTE(17)
```

"E" が返されます。

```
DBYTE(21)
```

備考

DBYTE( ) の使用に適する場面

DBYTE( ) 関数は、フィールドを定義しなくてもレコード内の特定の位置にある内容を調べる場合に使用できます。

DEC( ) 関数

指定された小数点以下の桁数で値または数式の結果を返します。

構文

```
DEC(数値, 小数位)
```

パラメーター

名前	種類	説明
数値	数値	小数位の数を調節する値または結果。 ◦ **整数** - 小数点桁数が末尾のゼロとして*数値*の最後に追加されます。 ◦ **小数** - 小数点桁数が減る場合、*数値*は切り捨てられずに、端数処理されます。小数点桁数が増える場合、末尾のゼロが*数値*の最後に追加されます。
小数位	数値	戻り値で使用する小数点以下の桁数。 **メモ** DEC( ) を使用して、結果の小数点精度を増やすことはできません。 小数点精度を増やす方法については、数式の端数処理と小数点精度の制御を参照してください。

出力

数値。

例

基本的な例

7.00 が返されます。

```
DEC(7, 2)
```

7.565 が返されます。

```
DEC(7.5647, 3)
```

7.56470 が返されます。

```
DEC(7.5647, 5)
```

高度な例

日歩の計算

Annual_rate というフィールドの日歩を小数点第 6 位まで求めるには、次のように指定します。

```
DEC(Annual_rate, 6) / 365
```

備考

DEC( ) の使用に適する場面

この関数は、フィールドで小数位を調整する場合や、値や計算結果の小数位を指定した桁数に丸める場合に使用できます。

DEC( ) は固定小数点端数処理を元に戻すことはできません

DEC( ) 関数を使用して、数式で固定小数点演算で実行される標準の端数処理を元に戻すことはできません。

例

Analytics で次の一連の式を検討します。

```
1.1 * 1.1 = 1.2
1.1 * 1.10 = 1.21
DEC(1.1 * 1.1, 2) = 1.20
```

固定小数点の端数処理は、`1.1 * 1.1` の結果は 1.2 で、1.21 ではありません。これは端数処理されていない結果です。DEC( ) を使用して、小数点 2 桁の結果を指定しても、小数点 2 桁の精度は作成されません。代わりに、末尾のゼロが追加され、精度を上げずに、指定された小数点桁数を作成します。

小数点精度を増やす方法については、数式の端数処理と小数点精度の制御を参照してください。

関連する関数

値を最も近い整数に丸めたい場合は、"ROUND( ) 関数" ページ 751 関数を使用してください。

DHEX( ) 関数

Unicode 文字列を 16 進数の文字列に変換します。

> **メモ:**
> この関数は Analytics の Unicode 版に固有です。非 Unicode 版ではサポートされない関数です。

構文

```
DHEX(フィールド)
```

パラメーター

名前	種類	説明
フィールド	文字	16 進数の文字列に変換する Unicode 文字列。

出力

文字。

例

基本的な例

"00410042004300310032003300" が返されます。

```
DHEX("ABC123")
```

備考

機能の仕組み

DHEX( ) 関数は、各 2 バイト文字をビッグエンディアンで、最上位の 2 バイトから順に格納して表示します。

各文字は4文字のコードで表されます。出力文字列は、*フィールド*値の4倍の長さで、16進数の値を構成する0から9までの数字とAからFまでの文字を含んでいます。

関連する関数

DHEX( ) は、16進数の文字列をUnicode文字列に変換するHTOU( ) 関数の逆関数です。

DICECOEFFICIENT( ) 関数

2 つの指定された文字列のダイス係数を返します。これは、2 つの文字列間の類似度を測定したものです。

構文

```
DICECOEFFICIENT(文字列 1, 文字列 2<,n-gram>)
```

パラメーター

名前	種類	説明
文字列1	文字	比較の最初の文字列。
文字列2	文字	比較の 2 番目の文字列。
n-gram 省略可能	数値	使用する *n*-gram の長さ。 1 以上の整数を指定します。*n-gram* の長さを大きくすると、2 つ文字列の間の類似度の基準が厳しくなります。 長さを指定しない場合、デフォルトの長さの 2 が使用されます。 *n*-gram は、ダイス係数計算の構成要素であり、比較対象となる 2 つの文字列にとって、構成要素であると同時に重なり合う、部分文字列(文字ブロック) です。 詳細については、"備考" ページ 532を参照してください。

出力

数値。値は 2 つの文字列のダイス係数です。これは、同一の 2 つの文字列の *n*-gram の合計数の割合を表します。範囲は 0.0000 ～ 1.0000 です。

例

基本的な例

n-gram の長さが結果に及ぼす影響

下の 3 つの例は、同じ 2 つの文字列を比較したものです。返される類似度は、指定した *n*-gram の長さによって変わります。

次の例では0.9167が返されます(デフォルトの*n*-gram長さ(2)を使用すると、2つの文字列間の*n*-gramの同一度が92%になる)。

```
DICECOEFFICIENT("125 SW 39TH ST, Suite 100","Suite 100, 125 SW 39TH ST")
```

次の例では1.0000が返されます(*n*-gram長さに1を使用すると、2つの文字列間の*n*-gramの同一度が100%になる)。

```
DICECOEFFICIENT("125 SW 39TH ST, Suite 100","Suite 100, 125 SW 39TH ST", 1)
```

次の例では、0.8261が返されます(*n*-gram長さに3を使用すると、2つの文字列間の*n*-gramの同一度が83%になる)。

```
DICECOEFFICIENT("125 SW 39TH ST, Suite 100","Suite 100, 125 SW 39TH ST", 3)
```

フィールドへの入力値

デフォルトの*n*-gram長さ2を使用した場合の、文字列「125 SW 39TH ST, Suite 100」と"**Address**"フィールドの各値との間のダイス係数が返されます。

```
DICECOEFFICIENT(Address,"125 SW 39TH ST, Suite 100")
```

高度な例

要素間の順序を入れ替えた文字列の使用

要素間の順序を入れ替えた文字列を検索する場合に、DICECOEFFICIENT( )を最適化するには、*n*-gram長さを小さくし、重要でない文字を削除します。

次の例では0.7368が返されます(デフォルトの*n*-gram長さ(2)を使用すると、2つの文字列間の*n*-gramの同一度が76%になる)。

```
DICECOEFFICIENT("John Smith","Smith, John")
```

次の例では、1.0000が返されます(姓と名の間のカンマを除外し、*n*-gram長さ1を使用すると、2つの文字列間の*n*-gramの同一度が100%になる)。

```
DICECOEFFICIENT("John Smith", EXCLUDE("Smith, John", ","), 1)
```

「125 SW 39TH ST, Suite 100」に対する類似度のランキング

次の例は、「125 SW 39TH ST, Suite 100」と"**Address**"フィールドの各値との間のダイス係数を表示する演算フィールド**Dice_Co**を作成しています。


```
DICECOEFFICIENT(Address,"125 SW 39TH ST, Suite 100")
```

「125 SW 39TH ST, Suite 100」との類似度に基づいてすべての "**Address**" フィールド値をランキングするには、演算フィールド **Dice_Co** をビューに追加し、そのフィールドの降順でクイックソートします。

「125 SW 39TH ST, Suite 100」に対するあいまい重複の抽出

次の例は、「125 SW 39TH ST, Suite 100」との指定した類似度以上の類似度を持つすべての "**Address**" フィールド値を抽出するフィルターを作成しています。

```
SET FILTER TO DICECOEFFICIENT(Address,"125 SW 39TH ST, Suite 100") > 0.5
```

式の数値を変更すると、フィルターリングされる値の類似度を調整することができます。

備考

DICECOEFFICIENT( ) の使用に適する場面

DICECOEFFICIENT( ) 関数は、ほぼ同一の値 (あいまい重複) を検出する場合に使用できます。また、同一またはほぼ同一の内容で、要素間の順序を入れ替えた文字列を検索する場合にも使用できます。例：

- 数字が入れ替わった電話番号、社会保障番号
- 形式が異なる複数の同じ住所

機能の仕組み

DICECOEFFICIENT( ) は 2 つの評価された文字列間のダイス係数を返します。ダイス係数は、文字列間の類似度を測定したもので、0.0000 ～ 1.0000 で示されます。返された値が大きいと、2 つの文字列間の類似度が高くなります。

- **1.0000** - 各文字列が同じ文字から構成されていることを示します。ただし、文字の順序と大文字小文字の表記は異なる場合があります。
- **0.7500** - 2 つの文字列の *n*-gram の同一度が 75% になることを意味します。
- **0.0000** - 2 つの文字列に共有 *n*-grams(以下で説明) がないか、計算で使用される *n*-gram の指定された長さが比較対象の 2 つの文字列のうちの短い方の文字列よりも長いことを意味します。

使用上のヒント

- **フィルターリングまたは並べ替え** - ダイス係数に基づいてフィールドの値をフィルターリングするか並べ替えると、比較文字列に最も類似している値が特定されます。
- **大文字と小文字の区別** - この関数では大文字と小文字が区別されないため、"SMITH" は "smith" と同じであると判断されます。
- **先頭と末尾にあるスペース** - フィールド内の先頭と末尾にあるスペースを自動的に除去するため、パラメーターとしてフィールドを指定するときに TRIM( ) および ALLTRIM( ) 関数を使う必要はありません。

- **一般的要素の除去** - OMIT( ) とEXCLUDE( ) 関数は、フィールド値から "Corporation" や "Inc."、カンマ、ピリオド、アンパサンド(&) 文字などの一般的要素を除去することによって、DICECOEFFICIENT( ) 関数の有効性を高めることができます。

  一般的要素と句読点の除去により、DICECOEFFICIENT( ) の文字列比較は、意味のある違いが発生する可能性のある文字列の部分だけに集中されます。

ダイス係数の計算方法

ダイス係数は、同一の2つの文字列の *n*-gram の合計数の割合を表します。

まず、ダイス係数は比較対象の2つの文字列を *n*-gram 文字ずつに分割して計算されます。*n*-grams(または *q*-grams) は長さが *n* の重なり合うサブ文字列または重なり合う文字ブロックです。*ngram* パラメーターを使用して *n* の長さを指定するか、デフォルトの長さの2を使用できます。

2 つの名前は *n*-grams に分割されます。

ここに名前、"John Smith" と "Smith, John D." があり、長さ2文字および3文字の *n*-gram に分割するとします。アンダースコアはスペースを示します。内部スペースと句読点は文字と見なされます。

n-gram の長さ	“John Smith” *n*-gram	"Smith, John D." *n*-gram
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.

ダイス係数の計算式

2つの比較対象の文字列に対して *n*-grams が確立されたら、次の式を使用して計算が完了します。

- *2 x 共有 n-grams の数 / 両方の文字列の n-grams の数*

"共有 *n*-grams" は両方の文字列に表示される *n*-grams です。たとえば、*n*-gram 長さ2(AB | **BC** と **BC** | CD) の場合、「ABC」と「BCD」は *n*-gram「BC」を共有します。

ダイス係数の計算例

次の表では、異なる *n*-gram の長さを使用して、2つの文字列「John Smith」および「John D. Smith」のダイス係数を計算する例を示します。

同じ文字列のペアの *n*-gram の長さが大きくなると、ダイス係数値が小さくなり、類似度が低いことを示します。文字列は同じですが、類似度の基準がより厳しくなります。文字列をより長い *n*-grams に分割する場合、共有と見なされるには、より長い連続文字が *n*-gram と一致する必要があります。

この点については、*n*-gram の長さを大きくすると、文字の相対位置の重みが大きくなると考えることもできます。対照的に、長さ1の *n*-gram を使用しているときには、文字の相対位置は考慮されません。相対位置は、文字列内での絶対的位置ではなく、文字列の、他の文字列からの相対的位置のことです。

> **ヒント**
> 特に入れ替えを検索している場合は、*n*-gram 長さ 1 を使用します。

n-gram の長さ	“John Smith” *n*-gram	"Smith, John D." *n*-gram	共有 *n*-grams	ダイス係数
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**
2 (デフォルト)	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( ) と ISFUZZYDUP( ) および LEVDIST( ) の比較

DICECOEFFICIENT( ) 関数とレーベンシュタイン距離を使用する ISFUZZYDUP( ) および LEVDIST( ) の主な違いの 1 つは、DICECOEFFICIENT( ) は比較対象の 2 つの文字列の文字または文字ブロックの相対位置を重視しないか、完全に無視するという点です。これに対して、レーベンシュタイン距離に基づく関数では相対位置が重要になります。

比較対象値 (要素入れ替えの考慮あり)

要素全体の配置が入れ替わる可能性がある住所などの文字列を比較する場合は、DICECOEFFICIENT( ) を使用した方がよいことがあります。たとえば、同じ住所で「Suite」要素が入れ替わっている場合は、DICECOEFFICIENT( ) を使用すると高い類似度が示されますが、LEVDIST( ) では高い相違性が示されます。

住所ペア	ダイス係数 (デフォルトの *n*-gram 2)	レーベンシュタイン距離
◦ 125 SW 39TH ST, Suite 100 ◦ Suite 100, 125 SW 39TH ST	0.9167	22 (レーベンシュタイン距離が大きくなれば、2 つの文字列間の相違も大きくなります)

比較対象値(要素入れ替えの考慮なし)

入れ替えが問題ではない場合、LEVDIST( ) はより有用な結果を提供する可能性があります。たとえば、同じ会社名に異なる句読点が含まれている場合、DICECOEFFICIENT( ) を使用すると高い相違度が示されますが、LEVDIST( ) では高い類似度が示されます。

会社名の組み合わせ	ダイス係数 (デフォルトの *n*-gram 2)	レーベンシュタイン距離
◦ AVS, Inc ◦ A.V.S. Inc	0.3750	3

DIGIT( ) 関数

指定された位置にある Packed データ型のバイトから、上位または下位の数字を返します。

構文

```
DIGIT(バイト位置, 位置)
```

パラメーター

名前	種類	説明
バイト位置	数値	レコード内のバイト位置。
位置	数値	返させる数字： ◦ バイトの上位 4 ビットを返させる場合は 1 を指定します。 ◦ バイトの下位 4 ビットを返させる場合は 2 を指定します。

出力

数値。

例

基本的な例

小数点以下 2 桁で、10 バイト目から始まる、値 123.45(00 12 34 5C) のパックフィールドは、データレコードでは次の形式で表されます。

	バイト 10	バイト 11	バイト 12	バイト 13
上位 4 ビット(1)	0	1	3	5
下位 4 ビット(2)	0	2	4	C

次の例では 3 が返されます(12 バイト目の上位 4 ビットに格納されている数字を取得します) 。

```
DIGIT(12, 1)
```

次の例では4が返されます(12バイト目の上位4ビットに格納されている数字を取得します)。

```
DIGIT(12, 2)
```

備考

機能の仕組み

DIGIT( ) は、1バイトの半分をそれぞれ区分し、位置パラメーターで指定されたバイトの値を0から15までの数字で返します。

DIGIT( ) の使用に適する場面

DIGIT( ) は個々のハーフバイト(上位バイトまたは下位バイト)にアクセスする場合に使用できます。Unisysアプリケーションなどの、ハーフバイト(4ビット)単位で配列されたパックフィールドを使用するアプリケーションで作業を行う場合は、これが必要です。

DOW( ) 関数

指定された日付または日付時刻の曜日を表す数値 (1 ～ 7) を返します。"Day of Week" の省略形です。

構文

```
DOW(日付/日付時刻)
```

パラメーター

名前	種類	説明
日付/日付時刻	日付時刻	曜日の数値を返す対象となるフィールド、式、またはリテラルの値。

出力

数値。

例

基本的な例

2014 年 12 月 31 日は 4 番目の曜日である水曜日に当たるため、4 が返されます。

```
DOW(`20141231`)
```

```
DOW(`20141231 235959`)
```

Invoice_date フィールドの各値の曜日を表す数値が返されます。

```
DOW(Invoice_date)
```

高度な例

週末に発生する取引の特定

DOW( ) 関数を使用して、週末に発生する取引を特定することができます。次のフィルターは、**Trans_Date** フィールドの日付のうち、土曜日または日曜日となる日付を抽出します。

```
SET FILTER TO DOW(Trans_Date) = 7 OR DOW(Trans_Date) = 1
```

備考

パラメーターの詳細

*日付/日付時刻*に指定されたフィールドは、フィールド定義で正しく書式を定義してさえいれば、どのような日付書式または日付時刻書式でも使用することができます。

リテラル日付または日付時刻値の指定

*日付/日付時刻*にリテラルの日付値または日付時刻値を指定する場合は、次の表内の書式に制限されます。また、`20141231` のように、値を逆引用符で囲む必要があります。

日付または時刻の個々の要素の間に、スラッシュ(/) やコロン(:) のような区切り文字をいっさい使用しないでください。

- **日付時刻値** - 以下の表に示す日付、区切り文字、および時刻の書式を任意に組み合わせて使用することができます。日付を時刻の前に置く必要があり、かつ、2 つの間に区切り文字を使用する必要があります。有効な区切り文字は空白スペース 1 つ、あるいは文字 't' または 'T' です。
- **時刻値** - 24 時間形式で時刻を指定する必要があります。UTC(Coordinated Universal Time: 協定世界時) からのオフセットは、プラス記号 (+) またはマイナス記号 (-) で始める必要があります。

形式の例	リテラル値の例
YYYYMMDD	`20141231`
YYMMDD	`141231`
YYYYMMDD hhmmss	`20141231 235959`
YYMMDDthhmm	`141231t2359`
YYYYMMDDThh	`20141231T23`
YYYYMMDD hhmmss+/-hhmm (UTC オフセット)	`20141231 235959-0500`
YYMMDD hhmm+/-hh (UTC オフセット)	`141231 2359+01`

形式の例	リテラル値の例
メモ UTC オフセットが設定されているデータのメインの時刻書式で hh を単独で使用しないでください。たとえば、「hh+hhmm」という使い方は避けてください。信頼できない結果になる可能性があります。	

関連する関数

以下の各値を返させたい場合は、それぞれ該当する関数を使用します。

- 曜日の名前を返させる場合：DOW( ) でなくCDOW( ) を使用
- 日付を数字(1 ～ 7) として返させる場合：DOW( ) でなくDAY( ) を使用

DTOU( ) 関数

Analytics 日付値を指定された言語およびロケール書式の Unicode 文字列に変換します。"Date to Unicode" の省略形です。

> **メモ**
>
> この関数は Analytics の Unicode 版に固有です。非 Unicode 版ではサポートされない関数です。

構文

```
DTOU(<日付> <,ロケールコード> <,スタイル>)
```

パラメーター

名前	種類	説明
日付 省略可能	日付時刻	Unicode 文字列に変換するフィールド、式、またはリテラル値。これを省略した場合は、現在のオペレーティングシステム日付が使用されます。 *日付*には日付時刻値を指定できますが、値の時刻部分は無視されます。単独の時刻値はサポートされていません。 フィールドまたはリテラルの日付値を指定できます。 ◦ **フィールド** -にはどのような日付書式でも使用することができます(ただし、フィールド定義で正しく書式を定義している場合) 。 ◦ **リテラル** -には YYYYMMDD 書式または YYMMDD 書式のいずれかを使用する必要があります。例:`20141231` サポートされている最も古い*日付*値は 1969 年 12 月 31 日です。
ロケール 省略可能	文字	出力文字列の言語(と、オプションで、特定の国や地域に関連付けられている言語バージョン) を指定するロケールコード。 たとえば、"zh" では中国語が指定され、"pt_BR" ではブラジルのポルトガル語が指定されます。 これを省略した場合は、コンピューターに設定されているデフォルトのロケールが使用されます。言語コードを指定したが、国コードを指定しなかった場合は、言語コードにデフォルトで設定されている国コードが適用されます。 *日付*を指定しなかった場合に、*ロケール*を指定することはできません。 ロケールコードに関する情報については、www.unicode.org を参照してください。
スタイル 省略可能	数値	Unicode 文字列に使用する日付書式スタイル。この書式スタイルは、指定したロケールの標準書式スタイルと一致します。

名前	種類	説明
		◦ 0 - "Sunday, September 18, 2016" などの完全仕様の書式 ◦ 1 - "September 18, 2016" などの長い書式 ◦ 2 - "Sep 18, 2016" などの中ぐらいの長さの書式 ◦ 3 - "9/18/16" などの短い数値型書式 これを省略した場合は、デフォルト値の 2 が使用されます。*日付とロケール*を指定しなかった場合に、*スタイル*を指定することはできません。

出力

文字。

例

基本的な例

リテラルの入力値

"31 de dezembro de 2014" が返されます。

```
DTOU(`20141231`, "pt_BR", 1)
```

"31 grudnia 2014" が返されます。

```
DTOU(`20141231`, "pl", 1)
```

フィールドの入力値

Invoice_date フィールドの数値の各日付に対する Unicode 文字列が返されます。

```
DTOU(Invoice_date, "zh", 1)
```

出力では正式の日付スタイルが使用されます。

"星期三, 2014 十二月 31" が返されます(地域識別子の指定なし) 。

```
DTOU(`20141231`, "zh", 0)
```

"2014年12月31日 星期三" が返されます(地域識別子の指定あり) 。

```
DTOU(`20141231`, "ja_JP", 0)
```

出力では長い形式の日付スタイルが使用されます。

"2014 十二月 31" が返されます(地域識別子の指定なし) 。

```
DTOU(`20141231`, "zh", 1)
```

"2014年12月31日" が返されます(地域識別子の指定あり) 。

```
DTOU(`20141231`, "ja_JP", 1)
```

備考

関連する関数

DTOU( ) は、Unicode 文字列を日付に変換するUTOD( ) 関数の逆関数です。

EBCDIC( ) 関数

EBCDIC 文字エンコードに変換された文字列を返します。

構文

```
EBCDIC(文字列)
```

パラメーター

名前	種類	説明
文字列	文字	EBCDIC に変換する値。

出力

文字。

例

基本的な例

Returns "ñòó@Æ '...@â£K":

```
EBCDIC("123 Fake St.")
```

高度な例

エクスポート対象となる、EBCDIC でエンコードされるフィールドの作成

EBCDIC エンコードを必要とするアプリケーションへエクスポートするために、**ADDRESS** フィールドの値の EBCDIC でエンコードされた値を格納するフィールドを作成するには、次のように指定します。

```
DEFINE FIELD Name_Exp COMPUTED EBCDIC(Name)
```

備考

EBCDIC( ) の使用に適する場面

この関数は、データを EBCDIC(Extended Binary Coded Decimal Interchange Code) 文字エンコードに変換する場合に使用できます。EBCDIC 文字エンコードは、主として z/OS のような IBM メインフレーム オペレーティング システム上で使用されます。

EFFECTIVE( ) 関数

貸付金に適用される実効年利率を返します。

構文

```
EFFECTIVE(名目利率, 期間)
```

パラメーター

名前	種類	説明
名目利率	数値	名目年利率。
期間	数値	年間の複利計算回数。 **メモ** 整数を指定してください。小数部分を指定しても切り捨てられるためです。

出力

数値。利率は小数点以下 8 桁で計算されます。

例

基本的な例

次の例では、クレジット カードの年利が 18% で、未払い分に対して毎月複利計算される場合の実効年利率、0.19561817(19.56%) が返されます。

```
EFFECTIVE(0.18, 12)
```


備考

実効年利率とは

貸付金の実効年利率とは、未払い額に対する利息に対して毎月または毎日複利計算が行われた場合の、実際に支払われた年利率です。

関連する関数

NOMINAL( ) 関数は EFFECTIVE( ) 関数の逆関数です。

EOMONTH( ) 関数

指定された日付から起算して、指定された月数だけ前または後の月の末日の日付を返します。

構文

EOMONTH(<*日付/日付時刻*> <,*月数*>)

パラメーター

名前	種類	説明
日付/日付時刻 省略可能	日付時刻	月末の日付を計算するための起算日となるフィールド、式、またはリテラル値。これを省略した場合は、現在のオペレーティング システム日付から月末の日付が求められます。 **メモ** *日付/日付時刻*には日付時刻値を指定できますが、値の時刻部分は無視されます。
月数 省略可能	数値	*日付/日付時刻*の前または後にある、月を示す数字。これを省略した場合は、デフォルトの 0(ゼロ) が使用されます。 *日付/日付時刻*を省略した場合に、*月数*を指定することはできません。

出力

日付時刻。日付値は、現在 Analytics に設定されている日付の表示書式を使用して出力されます。

例

基本的な例

入力値なし

現在のオペレーティング システム日付の月末の日付が返されます。

```
EOMONTH( )
```

リテラルの入力値

`20140131` に対して現在の ACL 日付表示書式である DD MMM YYYY を適用した、31 Jan 2014 が返されます。

```
EOMONTH(`20140115`)
```

`20140430` に対して現在の Analytics 日付表示書式である DD MMM YYYY を適用した、30 Apr 2014 が返されます。

```
EOMONTH(`20140115`, 3)
```

`20131031` に対して現在の Analytics 日付表示書式である DD MMM YYYY を適用した、31 Oct 2013 が返されます。

```
EOMONTH(`20140115`, -3)
```

フィールドの入力値

Invoice_date フィールドの各日付の 3 か月後の月の末日が返されます。

```
EOMONTH(Invoice_date, 3)
```

Invoice_date フィールドの各日付に 15 日の猶予期間を足した日付から 3 か月後の月の末日が返されます。

```
EOMONTH(Invoice_date + 15, 3)
```

Invoice_date フィールドの各日付の月の初日が返されます。

```
EOMONTH(Invoice_date, -1) + 1
```

備考

日付時刻書式

*日付/日付時刻*に指定されたフィールドは、フィールド定義で正しく書式を定義してさえいれば、どのような日付書式または日付時刻書式でも使用することができます。

リテラル日付値には、次のいずれかの書式を使用する必要があります。

- YYYYMMDD
- YYMMDD

リテラル日付値はバッククォートで囲む必要があります。例：`20141231`

月数の値の動作

- **値が正の場合** - 出力日付は指定された*日付/日付時刻*より新しい日付になります。
- **値が負の場合** - 出力日付は指定した*日付/日付時刻*より前の日付になります。
- **値を省略した場合、または '0'(ゼロ) を使用した場合** - 出力日付は*日付/日付時刻*が存在する月の末日になります。

月の初日の日付を返させる

月の初日の日付を返させるには、EOMONTH( ) 関数の結果に1を加算します。

次の例では `20140201` に対して現在の Analytics 日付表示書式である DD MMM YYYY を適用した、01 Feb 2014 が返されます。

```
EOMONTH(`20140115`) + 1
```

関連する関数

指定された日付から起算して、指定された月数だけ前または後の月の末日の日付ではなく正確な日付を返したい場合は、GOMONTH( ) 関数を使用してください。

EXCLUDE( ) 関数

指定した文字を除外する文字列を返します。

構文

```
EXCLUDE(文字列, 取り除く文字)
```

パラメーター

名前	種類	説明
文字列	文字	文字を除外する値。
取り除く文字	文字	取り除く文字のリスト。 *取り除く文字*の中に二重引用符を指定する場合は、文字のリスト全体を一重引用符で囲む必要があります。 例：`'"-/'`

出力

文字。

例

基本的な例

入力文字列からすべての数字を取り除いた" Alberni Street" が返されます。

```
EXCLUDE("1550 Alberni Street", "0123456789")
```

Product_Number フィールドにあるすべての値からスラッシュとシャープ記号を取り除いたものが返されます。

```
EXCLUDE(Product_Number, "/#")
```

備考

機能の仕組み

EXCLUDE( ) 関数は、*文字列*内の各文字を、*取り除く文字*に列挙された文字と比較します。一致した文字は出力文字列から取り除かれます。

たとえば、EXCLUDE("123-45-4536", "-") の出力は "123454536" になります。

一致する文字がない場合

*文字列*と*取り除く文字*の間に一致する文字がない場合は、関数の出力は*文字列*と同じになります。

たとえば、EXCLUDE("ABC", "D") の出力は "ABC" です。

大文字と小文字の区別

EXCLUDE( ) 関数では大文字と小文字が区別されます。*取り除く文字*に "ID" を指定した場合、"id#94022" からこれらの文字は取り除かれません。大文字と小文字の両方が混在している可能性がある場合は、UPPER( ) 関数を使用して*文字列*を大文字に変換します。

例：

```
EXCLUDE(UPPER("id#94022"), "ID#")
```

使用上のヒント

含める文字のセットが大きく、取り除く文字のセットが小さい場合は、EXCLUDE( ) を使用します。

一重引用符と二重引用符を取り除く

引用符は文字列の前後の区切り文字として使用されるため、一重引用符と二重引用符を取り除くには、引用符のタイプごとにEXCLUDE( ) が使用されるように、EXCLUDE( ) を入れ子にする必要があります。

```
EXCLUDE(EXCLUDE(field_to_process, """"), "")
```

関連する関数

EXCLUDE( ) 関数は INCLUDE( ) 関数の逆関数です。

EXP( ) 関数

数式の指数値(底 10) を指定された小数点以下の桁数で返します。

構文

```
EXP(数値, 小数位)
```

パラメーター

名前	種類	説明
数値	数値	指数値を返す対象となる数値フィールド、式、値。
小数位	数値	戻り値に含める小数点以下桁数。

出力

数値。

例

基本的な例

1000.00 が返されます。

```
EXP(3, 2)
```

72443.596007 が返されます。

```
EXP(4.86, 6)
```

高度な例

立方根を求める

フィールド X の立方根を小数点以下 2 桁で表すフィールドを作成するには、次のようにします。

```
DEFINE FIELD cube_root COMPUTED EXP(LOG(X, 6) / 3, 2)
```

> ヒント
> ある値の指数 *n* の累乗根は、値の対数を *n* で割ってから、その結果の指数を取ることによって求められます。

備考

機能の仕組み

この関数は数式の指数値 (底は 10) を返します。指数値は 10 の *n* 乗と定義されています。たとえば、3 の指数値は 10^3 、つまり 1000 です。

EXP( ) の使用に適する場面

EXP( ) 関数は、複雑な数学的計算を必要とする財務アプリケーションで使用します。EXP( ) 関数は指数演算子 (^) と同様の演算を行いますが、LOG( ) 関数も実行するアプリケーションで使用すると有用です。

関連する関数

指数の逆は対数です。したがって、EXP( ) は LOG( ) 関数の逆関数です。

FILESIZE( ) 関数

指定されたファイルのサイズをバイト数で返します。ファイルが存在しない場合は -1 を返します。

構文

```
FILESIZE(ファイル名)
```

パラメーター

名前	種類	説明
ファイル名	文字	ファイルの名前。 ファイルが Analytics プロジェクトと同じフォルダーに存在する場合は、ファイルのパスを指定する必要はありません。 他のフォルダーに存在するファイルの場合は、相対パスまたは絶対パスを指定します。例： ◦ "results\test_output.fil" ◦ "c:\results\test_output.fil" **メモ** テーブル名ではなく、Analytics テーブルの物理データファイル名 (.fil) を指定する必要があります。

出力

数値。

例

基本的な例

14744 が返されます。

```
FILESIZE("Inventory.fil")
```

サイズを調べるファイルが Analytics プロジェクトと同じフォルダーにない場合は、ファイルの相対パスか絶対パスを指定します。

6018 が返されます。

```
FILESIZE("C:\ACL Data\Sample Data Files\Backup\Ap_Trans.fil")
```

高度な例

ファイルが存在しない場合にスクリプトを実行する

`Metaphor_Inventory_2002.fil` ファイルが存在しない場合にのみ **import_data** というスクリプトを実行するには、次のコマンドを入力します。

```
DO SCRIPT import_data IF FILESIZE("Metaphor_Inventory_2002.fil") = -1
```

Analytics コマンド ログにファイルのサイズを記録する

Analytics コマンド ログに`Metaphor_Inventory_2002.fil` のサイズを記録するには、CALCULATE コマンドを使用します。

```
CALCULATE FILESIZE("Metaphor_Inventory_2002.fil")
```

FIND( ) 関数

指定された文字列が、特定のフィールド中またはレコード全体のどこかに存在するかどうか示す論理値を返します。

> **メモ**
>
> FIND( ) 関数と"FIND コマンド" ページ 200 は Analytics の 2 つの異なる機能であり、大きな違いがあります。

構文

```
FIND(文字列 <,検索フィールド>)
```

パラメーター

名前	種類	説明
文字列	文字	検索する文字列。この検索では大文字と小文字が区別されません。
検索フィールド 省略可能	文字	検索場所のフィールドまたは変数。省略した場合は、未定義部分もすべて含めたレコードの全体が検索されます。

出力

論理。指定された*文字列*の値が見つかった場合は T(true) 、そうでない場合は F(false) を返します。

例

基本的な例

レコード全体を検索する

次の例では、単一のフィールド、複数のフィールドのいずれか、または未定義部分の値が文字列 "New York" であるすべてのレコードに対し、'T' が返されます。それ以外のレコードに対しては 'F' が返されます。

```
FIND("New York")
```

単一のフィールド内の検索

次の例では、**City** フィールドの値が文字列 "New York" であるすべてのレコードに対し、'T' が返されます。それ以外のレコードに対しては 'F' が返されます。

```
FIND("New York", City)
```

City フィールドの値が文字列 "Ne" であるすべてのレコードに対し、'T' が返されます。それ以外のレコードに対しては 'F' が返されます。

```
FIND("Ne", City)
```

次の例では、**City** フィールドの値が 1 つ以上のスペースと文字列 "New York" であるすべてのレコードに対し、'T' が返されます。それ以外のレコードに対しては 'F' が返されます。

```
FIND(" New York", City)
```

次の例では、*v_search_term* 変数の値と完全一致または部分一致する値を **Description** フィールドに持つすべてのレコードに対し、'T' が返されます。それ以外のレコードに対しては 'F' が返されます。

```
FIND(v_search_term, Description)
```

複数のフィールド内の検索

次の例では、**City** フィールドか **City_2** フィールドのいずれかの値が文字列 "New York" であるすべてのレコードに対し、'T' が返されます。それ以外のレコードに対しては 'F' が返されます。

```
FIND("New York", City+City_2)
```

次の例では、**City** フィールドか **City_2** フィールドのいずれかの値が文字列 "New York" であるすべてのレコードに対し、'T' が返されます。それ以外のレコードに対しては 'F' が返されます。

```
FIND("New York", City) OR FIND("New York", City_2)
```

他の関数との組み合わせ

次の例では、**Last_Name_2** フィールドの値が **Last_Name** フィールドの値と完全一致または部分一致するすべてのレコードに対し、'T' が返されます。それ以外のレコードに対しては 'F' が返されます。

```
FIND(ALLTRIM(Last_Name), Last_Name_2)
```

備考

FIND( ) の使用に適する場面

FIND( ) 関数は、指定された*文字列*が特定のフィールド、2 つ以上のフィールドのいずれか、またはレコード内のいずれかのフィールドに存在するかどうかを調べる場合に使用できます。

文字列が一致する仕組み

*文字列*の値は、正確に一致することもあれば、長い文字列の中に含まれることもあります。フィールド内の先頭スペースは、*文字列*値の先頭に 1 つ以上のスペースを入れない限り、検索には影響しません。

レコード全体を検索する

任意指定の*検索フィールド*が指定されていない場合は、未定義部分もすべて含めたレコードの全体が検索されます。レコード全体が検索される場合には、フィールドの境界は無視され、フィールドの末尾のスペースは文字として扱われます。

> **メモ**
>
> レコード全体を検索する場合は、物理レコードが検索されます。演算フィールドおよび関連フィールドは検索されません。

フィールドのサブセットを検索する

テーブル内のフィールドのサブセット内を検索したい場合は、2 つ以上のフィールドを連結して*検索フィールド*に指定することができます。たとえば、**City** フィールドと **City_2** フィールドの両方で文字列 "New York" を検索するには、次の関数を使用します。

```
FIND("New York", City+City_2)
```

連結したフィールドは 1 つのフィールドのように扱われます。ただし、ALLTRIM( ) 関数を使用して個々のフィールドからスペースを除去しなければ、各フィールドの先頭と末尾のスペースを含むフィールドとなります。

また、各フィールド内を別々に検索する式を作成することもできます。

```
FIND("New York", City) OR FIND("New York", City_2)
```

*文字列*の先頭にスペースが含まれている場合には、この 2 つの手法の検索結果は異なってくる可能性があります。

大文字と小文字の区別と正確な文字比較

FIND( ) 関数は大文字と小文字を区別せず、ASCII 文字と EBCDIC 文字の両方を検索します。この関数は、**[正確な文字比較を行う]** オプション(SET EXACT ON/OFF) の影響を受けません。

演算フィールド内を検索する

演算フィールド内を検索するには、*検索フィールド*にその演算フィールドの名前を指定する必要があります。たとえば、**Vendor_City** が住所から市区町村を抽出する演算フィールドの場合は、次のように指定します。

```
FIND("New York", Vendor_City)
```

関連フィールド内を検索する

関連付けられたフィールド内を検索するには、*検索フィールド*値にフィールドの完全修飾名 (つまり、<*テーブル*>.<*フィールド名*>) を指定する必要があります。

```
FIND("New York", Vendor.Vendor_City)
```

日付時刻データまたは数値データを検索する

レコード レベルでの日付時刻データまたは数値データの検索には、FIND( ) 関数を使用できます。日付時刻または数値の検索では、*検索フィールド*の指定はサポートされていません。

数値または日付時刻 *文字列*は引用符で囲む必要があるとともに、ビューにおける書式ではなくソース データの書式と完全に一致する必要があります。

FIND( ) 関数を使用して演算フィールドや関連フィールド内で日付時刻データや数値データを検索することはできません。

> **メモ**
>
> FIND( ) 関数は、日付時刻データや数値データをあまりうまく検索できないため、お勧めしません。

FINDMULTI( ) 関数

指定された 1 つまたは複数の文字列が、特定のフィールド中、またはレコード全体のどこかに存在するかどうかを示す論理値を返します。

構文

FINDMULTI({*検索対象* | RECORD}, *文字列 1 <,...n>*)

パラメーター

名前	種類	説明
検索対象 | RECORD	文字	検索場所となるフィールドまたは変数。 未定義部分もすべて含めたレコードの全体を検索するには、キーワード、RECORD を指定します。 また、複数のフィールド名を連結したフィールド リストを指定することもできます。 `Field_1+Field_2+Field_3`
文字列 1 <,...n>	文字	検索する 1 つまたは複数の文字列。複数の検索文字列を指定する場合は、カンマで区切ります。 `FINDMULTI(RECORD, "Joa", "Jim", "Joh")` 検索は大文字と小文字を区別しません。

出力

論理。指定された*文字列*の値のいずれかが見つかった場合は **T**(True) 、そうでない場合は **F**(False) を返します。

例

基本的な例

レコード全体を検索する

次の例では、特定のフィールド、複数のフィールド、または未定義部分に"New York" または "Chicago" を含んでいるすべてのレコードに対し、'T' が返されます。それ以外のレコードに対しては 'F' が返されます。

```
FINDMULTI(RECORD, "New York", "Chicago")
```

単一のフィールド内の検索

次の例では、**City** フィールドの値が "New York" または "Chicago" であるすべてのレコードに対し、'T' が返されます。それ以外のレコードに対しては 'F' が返されます。

```
FINDMULTI(City, "New York", "Chicago")
```

次の例では、**City** フィールドの値が文字列 "Ne" または "Chi" であるすべてのレコードに対し、'T' が返されます。それ以外のレコードに対しては 'F' が返されます。

```
FINDMULTI(City, "Ne", "Chi")
```

次の例では、**City** フィールドの値が、"New York" または "Chicago" の前に 1 つ以上のスペースが付加された文字列であるすべてのレコードに対し、'T' が返されます。それ以外のレコードに対しては 'F' が返されます。

```
FINDMULTI(City, " New York", " Chicago")
```

次の例では、*v_search_term* 変数のいずれかの値と完全一致または部分一致する値を **Description** フィールドに持つすべてのレコードに対し、'T' が返されます。それ以外のレコードに対しては 'F' が返されます。

```
FINDMULTI(Description, v_search_term_1, v_search_term_2, v_search_term_3)
```

複数のフィールド内の検索

次の例では、**City** フィールドか **City_2** フィールドのどちらかの値が文字列 "New York" または "Chicago" であるすべてのレコードに対し、'T' が返されます。それ以外のレコードに対しては 'F' が返されます。

```
FINDMULTI(City+City_2, "New York", "Chicago")
```

次の例では、**City** フィールドか **City_2** フィールドのどちらかの値が文字列 "New York" または "Chicago" であるすべてのレコードに対し、'T' が返されます。それ以外のレコードに対しては 'F' が返されます。

```
FINDMULTI(City, "New York", "Chicago") または FINDMULTI(City_2, "New York", "Chicago")
```

他の関数との組み合わせ

次の例では、**Last_Name_1** フィールドの値が **Last_Name_2** または **Last_Name_3** フィールドの値と完全一致または部分一致するすべてのレコードに対し、'T' が返されます。それ以外のレコードに対しては 'F' が返されます。

```
FINDMULTI(Last_Name_1, ALLTRIM(Last_Name_2), ALLTRIM(Last_Name_3))
```

備考

FINDMULTI( ) の使用に適する場面

FINDMULTI( ) 関数は、指定された文字列のいずれかが特定のフィールド、2 つ以上のフィールドのいずれか、またはレコード内のいずれかのフィールドに存在するかどうかを調べる場合に使用できます。

文字列が一致する仕組み

*文字列*の値は、正確に一致することもあれば、長い文字列の中に含まれることもあります。フィールド内の先頭スペースは、*文字列*値の先頭に 1 つ以上のスペースを入れない限り、検索には影響しません。

レコード全体を検索する

*検索対象*フィールド パラメーターの代わりに RECORD を指定した場合には、レコードは、未定義部分もすべて含め、全体が検索されます。レコード全体が検索される場合には、フィールドの境界は無視され、フィールドの末尾のスペースは文字として扱われます。

> **メモ**
>
> レコード全体を検索する場合は、物理レコードが検索されます。演算フィールドおよび関連フィールドは検索されません。

フィールドのサブセットを検索する

テーブル内のフィールドのサブセット内を検索したい場合は、*検索対象*パラメーターの中で 2 つ以上のフィールドを連結することができます。たとえば、**City** フィールドと **City_2** フィールドの両方で文字列 "New York" または "Chicago" を検索するには、次の関数を使用します。

```
FINDMULTI(City+City_2, "New York", "Chicago")
```

連結したフィールドは 1 つのフィールドのように扱われます。ただし、ALLTRIM( ) 関数を使用して個々のフィールドからスペースを除去しなければ、各フィールドの先頭と末尾のスペースを含むフィールドとなります。

また、各フィールド内を別々に検索する式を作成することもできます。

```
FINDMULTI(City, "New York", "Chicago") または FINDMULTI(City_2, "New York", "Chicago")
```

*文字列*の値に先頭スペースが含まれている場合、この2つの手法の検索結果は異なる場合があります。

大文字と小文字の区別と正確な文字比較

FINDMULTI( ) 関数は大文字と小文字を区別せず、ASCII 文字とEBCDIC 文字の両方を検索します。この関数は、 **[正確な文字比較を行う]**オプション(SET EXACT ON/OFF) の影響を受けません。

演算フィールド内を検索する

演算フィールド内を検索するには、*検索対象*にその演算フィールドの名前を指定する必要があります。たとえば、**Vendor_City** が住所から市区町村を抽出する演算フィールドの場合は、次のように指定します。

```
FINDMULTI(Vendor_City, "New York", "Chicago")
```

関連フィールド内を検索する

関連付けられたフィールド内を検索するには、*検索対象*の値としてフィールドの完全修飾名(つまり、*<テーブル>.<フィールド名>*) を指定する必要があります。

```
FINDMULTI(Vendor.Vendor_City, "New York", "Chicago")
```

日付時刻データまたは数値データを検索する

レコードレベルで日付時刻データまたは数値データを検索するには、FINDMULTI( ) 関数を RECORD とともに使用します。日付時刻または数値の検索では、*検索対象*フィールドの指定はサポートされていません。

数値または日付時刻の*文字列*値は引用符で囲む必要があり、ビューにおける書式ではなくソースデータの書式と完全に一致する必要があります。

FINDMULTI( ) 関数を使用して演算フィールドや関連フィールド内で日付時刻データや数値データを検索することはできません。

> **メモ**
>
> FINDMULTI( ) 関数は、日付時刻データや数値データをあまりうまく検索できないため、お勧めしません。

FREQUENCY( ) 関数

連続する先頭の正の数値桁について、ベンフォードの法則で期待される頻度を小数点以下 8 桁で返します。

構文

```
FREQUENCY(数字列)
```

パラメーター

名前	種類	説明
数字列	文字	頻度を識別する数字(0 ～ 9) を含んでいる文字列。*数字列*は正数である必要があります。先頭のゼロは無視されます。

出力

数値。

例

基本的な例

0.00998422 が返されます。

```
FREQUENCY("43")
```

0.00000000 が返されます。

```
FREQUENCY("87654321")
```

> **メモ**
>
> 結果は 0.00000000495 になりますが、Analytics は計算結果を小数点以下 8 桁の精度にするため、ゼロ値が返されます。

備考

機能の仕組み

連続する先頭の正の数値桁について、ベンフォードの法則で予測される頻度を小数点以下 8 桁で返します。この結果を使用して、特定の状況に限定されたベンフォード検査を実施できます。

特定の数値の組み合わせにこの関数を使用する

特定の数字の組み合わせに着目したい場合は、BENFORD コマンドの代わりにこの関数を使用できます。たとえば、承認する請求金額が制限されている保険請求を監査する場合に、限度額をほんの少し下回るだけの請求金額について調べるためなどに FREQUENCY( ) 関数を使用することができます。

承認される限度額の $5,000 に近い請求金額を調べるために、$4,900 から $4,999 の範囲を選択することができます。まず、レコードの総数を求めます。次に、フィルターを使用して LEADING( ) 関数で 49 が返されるレコードの数を求めます。最後に、2 つの件数の比率と、FREQUENCY("49") を実行して得た値とを比較します。

この方法は、百万件のレコードを持つテーブルで完全な分析を実行するよりも短時間で実行でき、テーブルやコマンド ログのサイズも抑えられます。

6 桁を超える文字列を指定する

6 桁を超える文字列を指定すると結果がゼロになる場合があります。6 桁を超える文字列の計算では、Analytics の制限である小数点以下 8 桁よりも高い精度が必要になる場合があるためです。

FTYPE( ) 関数

フィールドまたは変数のデータカテゴリや、Analyticsプロジェクト項目の種類を識別する文字を返します。

構文

```
FTYPE(フィールド名文字列)
```

パラメーター

名前	種類	説明
フィールド名文字列	文字	フィールド名、変数名、またはAnalyticsプロジェクト項目の名前。 *フィールド名文字列*は引用符で囲みます。 `FTYPE("Amount")`

出力

文字。この関数はフィールド、変数、またはAnalyticsプロジェクト項目の種類を示す以下の文字の1つを返します。

- **"C"** - 文字フィールド
- **"N"** - 数値フィールド
- **"D"** - 日付時刻フィールド
- **"L"** - 論理フィールド
- **"c"** - 文字変数
- **"n"** - 数値変数
- **"d"** - 日付時刻変数
- **"l"** - 論理変数
- **"b"** - Analyticsスクリプト
- **"y"** - Analyticsテーブルレイアウト
- **"w"** - Analyticsワークスペース
- **"i"** - Analyticsインデックス
- **"r"** - Analyticsレポート
- **"a"** - Analyticsログファイル
- **"U"** - 未定義

例

基本的な例

次の例では、*num* 変数に値 4 を代入した後、その種類を検査しています。

"n" が返されます。

```
ASSIGN num = 4
FTYPE("num")
```

高度な例

フィールドのデータ型のテスト

数値型の **Amount** フィールドを必要とするスクリプト、つまりアナリティクスがあるとします。そのフィールドが適切な型であるかを調べてから、スクリプトを実行する必要があります。

次のコマンドは、**Amount** が数値フィールドの場合のみ Script_1 を実行します。

```
OPEN Invoices
DO Script_1 IF FTYPE("Amount") = "N"
```

テーブルまたは Analytics プロジェクト項目が存在するかどうかをテストする

次のコマンドは、プロジェクトに Invoices というテーブルがある場合に限り、Script_1 を実行します。

```
DO Script_1 IF FTYPE("Invoices") <> "U"
```

ランタイム環境をテストする

アナリティクスを Analytics、Analytics Exchange、分析アプリウィンドウのいずれで実行できるかを特定するには、FTYPE を使用します。

アナリティクスが Analytics Exchange または分析アプリウィンドウで実行できる場合には、'ax_main' に対する戻り値が 'b' になります。

```
IF FTYPE('ax_main') = 'b' v_running_in_ax_or_analysis_app = T
```

アナリティクスが Analytics で実行できる場合には、'ax_main' に対する戻り値が 'b' 以外の値になります。

```
IF FTYPE('ax_main') = 'b' v_running_in_ax_or_analysis_app = T
```

ランタイム環境を特定するこの機能により、実行環境のアプリケーションに応じて異なるコード ブロックを実行する単一のスクリプトを設計することができます。

FVANNUITY( ) 関数

一定の利率を使って計算した一連の預金額の将来価値を返します。将来価値は、毎月の預金額と複利利息の合計です。

構文

FVANNUITY(*利率*, *期間*, *支払金額* <,*種類*>)

パラメーター

名前	種類	説明
利率	数値	1 期あたりの利率。
期間	数値	支払期間の総数。
支払金額	数値	1 期間あたりの支払金額。 年金期間中、支払金額は毎回同じ金額でなければなりません。
種類 省略可能	数値	支払いのタイミング: ◦ 0 - 期末払い ◦ 1 - 期首払い 支払いのタイミングが省略された場合は、デフォルト値の 0 が使用されます。

> **メモ**
>
> *利率*、*期間*、*支払金額*を指定する際には、**1 期あたりの利率**を確実に指定するため、整合性のある期間を使用する必要があります。
>
> 例:
>
> - 年利 5% の 2 年間の貸付金または投資に対して月払いする場合は、*利率*に 0.05/12、*期間*に 2 * 12 を指定します。
> - 同じ貸付金または投資に対して年払いする場合は、*利率*に 0.05、*期間*に 2 を指定します。

出力

数値。結果は小数点以下 2 桁まで計算されます。

例

基本的な例

月次支払額

2 年間、毎月月初に支払う$1,000 の将来価値(月利 1% の複利) として、($) 27243.20 が返されます。

```
FVANNUITY(0.01, 12, 1000, 1)
```

同じ年金の 1 年後の将来価値として、($) 12809.33 が返されます。

```
FVANNUITY(0.01, 2*12, 1000, 1)
```

年次支払額

2 年間、毎年年末に支払う$12,000 の将来価値(年利 12% の複利) として、($) 25440.00 が返されます。

```
FVANNUITY(0.12, 2, 12000, 0)
```

高度な例

年金の計算

年金の計算では、次の 4 つの変数が使用されます。

- **現在価値または将来価値** - 下の例では $21,243.39 と$ 26,973.46
- **1 期間あたりの支払金額** - 下の例では $1,000.00
- **1 期間あたりの利率** - 下の例では月あたり 1%
- **期間の数** - 下の例では 24 か月

これらの変数のうち 3 つの値がわかっている場合は、Analytics 関数を使って残りの変数の値を計算できます。

求めたい値:	使用する Analytics 関数
現在価値	PVANNUITY( ) 21243.39 を返す: `FVANNUITY(0.01, 12, 1000)`
将来価値	FVANNUITY( ) 26973.46 を返す:

求めたい値：	使用する Analytics 関数
	FVANNUITY(0.01, 24, 1000)
1 期間あたりの支払金額	PMT( ) 1000 を返す： PMT(0.01, 24, 21243.39)
1 期間あたりの利率	RATE( ) 0.00999999(1%) を返す： RATE(24, 1000, 21243.39)
期間の数	NPER( ) 24.00 を返す： NPER(0.01, 1000, 21243.39)

年金の式

期末年金(期末払い) の**現在価値**を計算する式：

期末年金(期末払い) の**将来価値**を計算する式：

備考

関連する関数

PVANNUITY( ) 関数は FVANNUITY( ) 関数の逆関数です。

FVLUMPSUM( ) 関数

一定の利率を使って計算した現在の総額の将来価値を返します。

構文

FVLUMPSUM(*利率*, *期間*, *金額*)

パラメーター

名前	種類	説明
利率	数値	1 期あたりの利率。
期間	数値	期間の総数。
金額	数値	最初の期間の冒頭に支払う投資(出資)額。

> **メモ**
>
> *利率*、*期間*を指定する際には、**1 期あたりの**利率を確実に指定するため、整合性のある期間を使用する必要があります。
>
> 例:
>
> - 年利 5% の 2 年間の貸付金または投資に対して月*払い*する場合は、*利率*に 0.05/12、期間に 2 * 12 を指定します。
> - 同じ貸付金または投資に対して年払いする場合は、*利率*に 0.05、*期間*に 2 を指定します。

出力

数値。結果は小数点以下 2 桁まで計算されます。

例

基本的な例

月ごとに複利が付く場合

総額 $1,000を月利 1% の複利で 2 年間運用した場合の将来価値として、($) 1269.73 が返されます。

```
FVLUMPSUM(0.01, 2*12, 1000)
```

同じ投資の1年後の将来価値として、($) 1126.83 が返されます。

```
FVLUMPSUM(0.01, 12, 1000)
```

月利 1% の複利で2年間運用した場合の $21,455.82 の将来価値として、($) 27243.20 が返されます。

```
FVLUMPSUM(0.01, 2*12, 1000)
```

半年ごとに複利が付く場合

総額 $1,000 を年利 12% の年複利で2年間運用した場合の将来価値として、($) 1262.48 が返されます。

```
FVLUMPSUM(0.12/2, 2*2, 1000)
```

1年ごとに複利が付く場合

総額 $1,000 を年利 12% の年複利で2年間運用した場合の将来価値として、($) 1254.408 が返されます。

```
FVLUMPSUM(0.12, 2, 1000)
```

備考

将来価値とは

運用する総額の将来価値は、初期投資元本と複利の合計です。

関連する関数

PVLUMPSUM( ) 関数は FVLUMPSUM( ) 関数の逆関数です。

FVSCHEDULE( ) 関数

一連の利率を使って計算した現在の総額の将来価値を返します。

構文

FVSCHEDULE(*元金*, *利率1*<,*利率2*...>)

パラメーター

名前	種類	説明
元金	数値	初期投資額。
利率1, *利率2*...	数値	同じ長さを持つ複数の期間における一連の利率。 **メモ** タイプが同じ複数の期間により、複数の月、年、またはその他の期間を表すことができます。 **1 期間あたりの**利率を指定する必要があります。したがって、利率の 1 つが年 5% で、期間が月の場合は、0.05/12 を指定します。

出力

数値。結果は小数点以下 2 桁まで計算されます。

例

基本的な例

$1000 の総額を 1 年目に 10%、2 年目に 9%、3 年目に 7% の年複利で運用した場合の将来価値として、($) 1282.93 が返されます。

```
FVSCHEDULE(1000, 0.1, 0.09, 0.07)
```

備考

運用する総額の将来価値は、初期投資元本と複利の合計です。

GETOPTIONS( ) 関数

指定されたAnalytics オプションの現在の設定([**オプション**]ダイアログ ボックスの設定) を返します。

構文

```
GETOPTIONS(オプション)
```

パラメーター

名前	種類	説明
オプション	文字	設定を返す対象となる Analytics オプション。 オプションの名前は、以下の一覧に示されるとおり正確に指定する必要があり、また引用符で囲む必要があります。 ◦ **separators** - Analytics の 3 つの区切り文字の現在の設定を次の順序で返します。 • 小数点の記号 • 桁区切り記号 • リストの区切り文字 **メモ** 現時点では、"separators" が、GETOPTIONS( ) 関数で指定できる唯一の*オプション*です。

出力

文字。

例

基本的な例

3 つの Analytics 区切り文字の現在の設定が返されます。例 : ".,,"

```
GETOPTIONS("separators")
```

高度な例

スクリプト内での GETOPTIONS( ) の使用

スクリプトで Analytics の区切り文字のうち 1 つ以上を変更する必要が生じた場合、GETOPTIONS( ) 関数により現在の設定を検出することができます。現在の設定を変数に格納しておき、スクリプトの最後で元に戻すことができます。

```
ASSIGN v_SeparatorsSetting = GETOPTIONS("separators")
SET SEPARATORS ",.;"
<スクリプト内容>
SET SEPARATORS "%v_SeparatorsSetting%"
```

備考

3 つの Analytics 区切り文字が、 [**オプション**]ダイアログボックスにある以下のオプションとして指定されています。

- **小数点の記号**
- **桁区切り記号**
- **リストの区切り文字**

GOMONTH( ) 関数

指定された日付から、指定された月数前または月数後の日付を返します。

構文

```
GOMONTH(日付/日付時刻, 月数)
```

パラメーター

名前	種類	説明
日付/日付時刻	日付時刻	出力日付を求めるフィールド、式、またはリテラル値。
月数	数値	*日付/日付時刻*の前または後にある、月を示す数字。 **メモ** *日付/日付時刻*には日付時刻値を指定できますが、値の時刻部分は無視されます。

出力

日付時刻。日付値は、現在 Analytics に設定されている日付の表示書式を使用して出力されます。

例

基本的な例

リテラルの入力値

現在の Analytics 日付表示書式 DD MMM YYYY の形式を `20140415` に適用した値、15 Apr 2014 が返されます。

```
GOMONTH(`20140115`, 3)
```

`20131015` に対して現在の Analytics 日付表示書式である DD MMM YYYY を適用した、15 Oct 2013 が返されます。

```
GOMONTH(`20140115`, -3)
```

`20140430` に対して現在の Analytics 日付表示書式である DD MMM YYYY を適用した、30 Apr 2014 が返されます(無効な日付である 2014 年 4 月 31 日が返されないように日付が丸められます) 。

```
GOMONTH(`20140330`, 1)
```

```
GOMONTH(`20140331`, 1)
```

`20140501` に対して現在の Analytics 日付表示書式である DD MMM YYYY を適用した、01 May 2014 が返されます。

```
GOMONTH(`20140331`, 1)
```

フィールドの入力値

Invoice_date フィールドの各日付の 3 か月後の日付が返されます。

```
GOMONTH(Invoice_date, 3)
```

Invoice_date フィールドの各日付に猶予期間 15 日を加えた日付から 3 か月後の日付が返されます。

```
GOMONTH(Invoice_date + 15, 3)
```

備考

日付時刻書式

*日付/日付時刻*に指定されたフィールドは、フィールド定義で正しく書式を定義してさえいれば、どのような日付書式または日付時刻書式でも使用することができます。

リテラル日付値には、次のいずれかの書式を使用する必要があります。

- YYYYMMDD
- YYMMDD

リテラル日付値はバッククォートで囲む必要があります。例 : `20141231`

月数の値の動作

- **値が正の場合** - 出力日付は指定された*日付/日付時刻*より新しい日付になります。
- **値が負の場合** - 出力日付は指定した*日付/日付時刻*より前の日付になります。

- **値を省略した場合、または '0'(ゼロ) を使用した場合** - 出力日付は*日付/日付時刻*と同じになります。

実在しない日付にしないための日付の丸め処理

*日付/日付時刻*と*月数*の組み合わせが実在しない日付を作り出す場合は、GOMONTH( ) 関数は「日付の丸め」を使用して、同月内の最も近い有効な日付を返します。

次の例の場合、31 Apr 2014 は無効な日付であるため、`20140430`(30 Apr 2014) が返されます。

```
GOMONTH(`20140331`,1)
```

関連する関数

指定された日付から、指定された月数だけ前または後に当たる月の、正確な日付ではなく末日の日付を返したい場合は、EOMONTH( )関数を使用してください。

HASH( ) 関数

入力値に基づいてソルト付き暗号化ハッシュ値を返します。

構文

HASH(*フィールド* <,*ソルト値*>)

パラメーター

名前	種類	説明
フィールド	文字 数値 日付時刻 論理	ハッシュする値。
ソルト値 省略可能	文字 数値	使用するソルト値。1 から 10 までの PASSWORD 識別番号か、または文字列を指定できます。 これを省略した場合は、Analytics のデフォルトのソルト値が使用されます。 ソルト値は 128 文字に制限されています。それより長いソルト値を指定した場合は、自動的に 128 文字に切り詰められます。 詳細については、"ソルト値" ページ 585を参照してください。

出力

文字。

例

基本的な例

Analytics のデフォルトのソルト値を使用する場合

"819A974BB91215D58E7753FD5A42226150100A0763087CA7DECD93F3C3090405" が返されます。

```
HASH("555-44-3322")
```

Credit_card_num フィールド内の各番号のハッシュ値が返されます。

```
HASH(Credit_card_num)
```

ユーザー指定のソルト値を使用する場合

"AD1E7D9B97B6F6B5345AB13471A74C31EBE6630CA2622BB7E8C280E9FBEE1F17" が返されます。

```
HASH("555443322", "マイ ソルト値 123")
```

高度な例

同一のハッシュ値が確実に生成されるようにする

クリア テキスト値を統一して同一のハッシュ値を生成するために、HASH( ) とともにその他の関数を使用します。

次の一連の例について考えてください。最初の 2 つの例では、クリア テキスト値の大文字と小文字の違いによって出力のハッシュ値がまったく変更されるかどうかに注目してください。

"DF6789E1EC65055CD9CA17DD5B0BEA5892504DFE7661D258737AF7CB9DC46462" が返されます。

```
HASH("John Smith")
```

"3E12EABB5940B7A2AD90A6B0710237B935FAB68E629907927A65B3AA7BE6781D" が返されます。

```
HASH("John Smith")
```

次の例では、UPPER( ) 関数を使って大文字と小文字の違いを統一することで、同一のハッシュ値が生成されます。

"3E12EABB5940B7A2AD90A6B0710237B935FAB68E629907927A65B3AA7BE6781D" が返されます。

```
HASH(UPPER("John Smith"))
```

HASH( ) を使用した大きなテキストのブロック同士の比較

HASH( ) 関数を使用して、2 つのコメント フィールド内のテキストのブロックが同一であるかどうかをテストすることができます。

このテストを実行するには、以下と同様の 2 つの演算フィールドを作成してから、同一でないテキスト ブロックを見つけるためのフィルターを作成します。

```
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
```

コメント フィールドが別々のテーブルにある場合は、各テーブルにHASH( ) 演算フィールドを作成し、その演算フィールドを共通のキー フィールドとして使用してそれら2つのテーブルを不一致結合します。結合された出力テーブル内のレコードは、同一でないテキスト ブロックを表します。

備考

HASH( ) の使用に適する場面

HASH( ) 関数は、クレジット カード番号、給与情報、および社会保障番号などの機密データを保護する場合に使用できます。

機能の仕組み

HASH( ) では一方向のエンコードが行われます。クリア テキストのデータを使用してハッシュ値を生成することはできますが、その後、そのハッシュ値のエンコードを解除したり、ハッシュ値を復号したりすることはできません。

特定のクリア テキスト値は常に同じハッシュ値を生成するため、ハッシュされたクレジット カード番号のフィールドで重複を検索したり、ハッシュされたクレジット カード番号の2つのフィールドを結合したりすることができます。この操作を同等のクリア テキスト フィールドで実行した場合、結果は同じになります。

機密データの保護

機密データをサーバー上に保存することを避けるために、ローカルでHASH( ) 関数を使用して演算フィールドを作成することができます。ハッシュされたフィールドとその他必要なフィールドを抽出すると同時に、クリア テキスト フィールドを除外して、新しいテーブルを作成します。分析でサーバー上の新しいテーブルを使用して、一度その結果を得たら、ハッシュされたデータのクリア テキスト バージョンを確認する必要がある場合には、元のテーブルを参照することができます。

初期使用以外で、機密データをローカルに保存することが禁止されている場合は、ハッシュ値を使って新しいテーブルを作成した後、元のテーブルを削除してもかまいません。削除しても、クリア テキスト値について元のソース システムを参照することができます。

クリア テキスト値は同一である必要あり

同一のハッシュ値を生成するためには、2つのクリア テキスト値はまったく同じでなければなりません。たとえば、同じクレジット カード番号でもハイフンがあるかどうかによって、また、同じ名前でも先頭文字が大文字かすべて大文字かどうかによって、異なるハッシュ値になります。

クリア テキスト値を統一するために、INCLUDE( )、EXCLUDE( )、またはUPPER( ) などの関数をHASH( ) 関数に組み込む必要があるかもしれません。

先頭と末尾のスペースはHASH( ) 関数によって自動的に除去されるため、TRIM( ) 関数やALLTRIM( ) 関数を使用する必要はありません。

先頭や末尾にあるスペースに意味がある場合の処理

値の先頭や末尾にあるスペースで、値間の違いを表しているデータがある場合は、スペースを別の文字に置き換えてから、値のハッシュ値を求める必要があります。

ハッシュ処理を行う前に、フィールド値内のスペースをアンダースコア文字(_) に置き換えます。

```
HASH(REPLACE(フィールド名, " ", "_"))
```

HASH( ) で使用される暗号化アルゴリズム

HASH( ) は、SHA-2 暗号化ハッシュアルゴリズムを使用します。このアルゴリズムでは、入力値の長さに関係なく、64 バイトの固定長のハッシュされた出力が生成されます。クリアテキストの入力値は 64 バイトより長くてもかまいません。

ソルト値

機能の仕組み

HASH( ) 関数によって提供される保護は、ハッシュする前にソルト値を自動的に追加することによって強化されます。ソルト値は、ソースデータ値と連結される英数字の文字列です。連結された文字列全体を使用して、ソルト付きのハッシュされた値が生成されます。この手法は、ハッシュされた値の復号技術に対する耐性を強めます。

任意で独自のソルト値を指定する

ソルト値を指定しない限り、固定のデフォルトのソルト値が使用されます。次のいずれかの方法を使用して、ソルト値を指定できます。

- **クリアテキスト文字列としてのソルト値**

  英数字の文字列を指定します。例：

  ```
  HASH(Credit_card_num, "my salt value")
  ```

- **パスワードとしてのソルト値**

  HASH( ) 関数と併せてPASSWORDコマンドを使用し、1 から 10 までのPASSWORD 識別番号を指定します。例：

  ```
  PASSWORD 3 "ソルト値を入力"
  EXTRACT FIELDS HASH(Credit_card_num, 3) TO "Protected_table"
  ```

> **メモ**
>
> HASH( ) 関数のフィールドを抽出するには、その前にPASSWORD のソルト値を入力しておく必要があります。

HASH() とともにPASSWORD 識別番号を使用することの利点は、クリアテキストのソルト値をあらわにしないで済むことです。

詳細については、"PASSWORD コマンド" ページ 345を参照してください。

パスワード方式のガイドライン

このパスワード方式は、スクリプトの先頭、あるいはスクリプト内で HASH( ) 関数が現れる前に、パスワードの入力を求めるスクリプトで使用することを目的としています。

PASSWORD の割り当ては Analytics を閉じるときに削除されるため、パスワード方式は演算フィールドでの使用には適していません。

さらに、パスワード ベースのソルト値を使用している演算フィールドは、Analytics を再度開くときにビューから自動的に削除されます。この削除は、デフォルトのソルト値を使用したハッシュ値の再計算を避けるために必要です。再計算される値は、ユーザー指定したソルト値を使って計算された元のハッシュ値とは異なることになります。

HEX( ) 関数

ASCII 文字列を 16 進数の文字列に変換します。

構文

```
HEX(フィールド)
```

パラメーター

名前	種類	説明
フィールド	文字	16 進数の文字列に変換する ASCII 文字列。

出力

文字。

例

基本的な例

"3132333435" が返されます。

```
HEX("12345")
```

Count フィールドの値に対する 16 進数の文字列が返されます。

```
HEX(Count)
```

備考

機能の仕組み

この関数は、指定されたフィールド値または式と等価な 16 進数の文字列を返します。CR(キャリッジ リターン) 、LF(ライン フィード) 、NUL(ヌル) など画面に表示できない文字を含んでいるフィールドの正確な内容を確認する必要がある場合に、この関数を使用できます。

戻り値の長さ

戻り値は、*フィールド*値の 2 倍の長さを持つ文字列になります。0 から 9 までの数字と A から F までの文字 (10 から 15 までの数字に相当) によって 16 進数の値を表します。

式でなくフィールドを入力値として使用する

一般に、この関数は式ではなくフィールドに適用します。式に適用すると、式の内部記憶形式が表示され、ほとんどの場合意味がないためです。

HOUR( ) 関数

指定された時刻または日付時刻から時間を抽出し、それを 24 時間制の数値として返します。

構文

```
HOUR(時刻/日付時刻)
```

パラメーター

名前	種類	説明
時刻/日付時刻	日付時刻	時間を抽出するフィールド、式、またはリテラル値。

出力

数値。

例

基本的な例

23 が返されます。

```
HOUR(`t235959`)
```

```
HOUR(`20141231 235959`)
```

Call_start_time フィールドの各値の時間数が返されます。

```
HOUR(Call_start_time)
```


備考

パラメーターの詳細

*時刻/日付時刻*に指定されたフィールドは、フィールド定義で正しく書式を定義してさえいれば、どのような時刻書式または日付時刻書式でも使用することができます。

リテラル時刻または日付時刻値の指定

*日付時刻*にリテラルの時刻値または日付時刻値を指定する場合は、次の表内の書式に制限されます。また、`20141231 235959` のように、値を逆引用符で囲む必要があります。

日付または時刻の個々の要素の間に、スラッシュ(/) やコロン(:) のような区切り文字をいっさい使用しないでください。

- **時刻値** - 以下の表に示す任意の時刻の書式を使用することができます。関数が正しく動作するためには、単独の時刻値の前に区切り文字を使用する必要があります。有効な区切り文字は文字 't' または 'T' です。24 時間形式で時刻を指定する必要があります。UTC(Coordinated Universal Time：協定世界時) からのオフセットは、プラス記号(+) またはマイナス記号(-) で始める必要があります。
- **日付時刻値** - 以下の表に示す日付、区切り文字、および時刻の書式を任意に組み合わせて使用することができます。日付を時刻の前に置く必要があり、かつ、2 つの間に区切り文字を使用する必要があります。有効な区切り文字は空白スペース 1 つ、あるいは文字 't' または 'T' です。

形式の例	リテラル値の例
thhmmss	`t235959`
Thhmm	`T2359`
YYYYMMDD hhmmss	`20141231 235959`
YYMMDDthhmm	`141231t2359`
YYYYMMDDThh	`20141231T23`
YYYYMMDD hhmmss+/-hhmm (UTC オフセット)	`20141231 235959-0500`
YYMMDD hhmm+/-hh (UTC オフセット)	`141231 2359+01`
メモ UTC オフセットが設定されているデータのメインの時刻書式で hh を単独で使用しないでください。たとえば、「hh+hhmm」という使い方は避けてください。信頼できない結果になる可能性があります。	

HTOU( ) 関数

16 進数の文字列を Unicode 文字列に変換します。"Hexadecimal to Unicode" の省略形です。

> **メモ**
>
> この関数は Analytics の Unicode 版に固有です。非 Unicode 版ではサポートされない関数です。

構文

```
HTOU(16進数文字列)
```

パラメーター

名前	種類	説明
16 進数文字列	文字	Unicode 文字列に変換する 16 進数の文字列。この文字列には "20AC" などの 16 進値のみを指定できます。

出力

文字。

例

基本的な例

"ABC123" が返されます。

```
HTOU("004100420043003100320033")
```

高度な例

金額に通貨記号を付加する

金額単位フィールドを新しいテーブルに抽出する必要があるとします。このフィールドには、ユーロ通貨記号 (€)

と、数値型の **Amount** フィールドの値が表示されます。

```
EXTRACT HTOU("20AC") + STRING(Amount, 10) AS "Currency_Amount" TO Display_Table
```

この EXTRACT コマンドを実行すると、HTOU( ) により、ユーロ記号 "€" と、STRING( ) によって文字に変換された **Amount** の値が併せて表示されます。**Amount** の元の値が 2000 の場合は、**Currency_Amount** の値は "€2000" となります。

備考

関連する関数

HTOU( ) は、Unicode 文字列を 16 進数文字列に変換する DHEX( ) 関数の逆関数です。

INCLUDE( ) 関数

指定した文字のみを含む文字列を返します。

構文

```
INCLUDE(文字列, 含める文字)
```

パラメーター

名前	種類	説明
文字列	文字	含める文字に制限する値。
含める文字	文字	含める文字のリスト。 *含める文字*の中に二重引用符を指定する場合は、文字のリスト全体を一重引用符で囲む必要があります。 例：`'"-/'` **メモ** 含める文字のリストに指定した文字が*文字列*にない場合には、その文字は戻り値に含まれません。

出力

文字。

例

基本的な例

入力文字列の中から数字のみを選択した"123" が返されます。

```
INCLUDE("123 Main St.", "0123456789")
```

入力文字列の中から数字のみを選択した"1231234" が返されます。

```
INCLUDE("123-123-4", "1243")
```

入力文字列には "D" が含まれていないため、"" が返されます(つまり、何も返されません) 。

```
INCLUDE("ABC", "D")
```

備考

機能の仕組み

INCLUDE( ) 関数は、*文字列*内の各文字を、*含める文字*に列挙された文字と比較します。一致した文字は出力文字列に追加されます。

一致する文字がない場合

*文字列*と*含める文字*の間に一致する文字がない場合、関数の出力は空になります。

大文字と小文字の区別

INCLUDE( ) 関数では大文字と小文字が区別されます。このため、*含める文字*に "ID" を指定しても、これらの文字は "id#94022" には含まれていないことになります。大文字と小文字の両方が混在している可能性がある場合は、UPPER( ) 関数を使用して*文字列*を大文字に変換します。

例 :

```
INCLUDE(UPPER("id#94022"), "ID0123456789")
```

使用上のヒント

含める文字のセットが小さく、取り除く文字のセットが大きい場合は、INCLUDE( ) を使用します。

関連する関数

INCLUDE( ) 関数は EXCLUDE( ) 関数の逆関数です。

INSERT( ) 関数

元の文字列の指定のバイト位置に、指定した文字列が挿入された内容が返されます。

構文

```
INSERT(文字列, 挿入するテキスト, 位置)
```

パラメーター

名前	種類	説明
文字列	文字	テキストを挿入する値。
挿入するテキスト	文字	挿入するテキスト。
位置	数値	*挿入テキスト*を*文字列*に挿入する文字の位置。

出力

文字。

例

基本的な例

"aXXXbcde" が返されます。

```
INSERT("abcde", "XXX", 2)
```

"XXXabcde" が返されます。

```
INSERT("abcde", "XXX", 0)
```

`"abcde"` の長さが 5 バイトのため、バイト位置 8 でなく 6 に `"XXX"` を挿入した "abcdeXXX" が返されます。

```
INSERT("abcde", "XXX", 8)
```

備考

機能の仕組み

INSERT( ) 関数は、指定された文字またはスペースを文字列の指定された桁に挿入します。

INSERT( ) の使用に適する場面

INSERT( ) は、データを正規化する場合に使用できます。データの正規化は、書式を設定する場合や、重複するデータを突き合わせる場合、また、フィールド同士が一致している必要のある JOIN および DEFINE RELATION コマンドを使用する場合などに行います。

たとえば、パーツ番号フィールドが、あるファイルでは "12345" という形式になっていて、別のファイルでは "12-345" という形式になっているとします。最初のファイルでは、INSERT( ) 関数を使って、3 桁目にハイフン(-) を挿入できます。

位置についてのガイドライン

- *位置*の値が*文字列*の長さより大きい場合、*挿入するテキスト*の値は文字列の最後に挿入されます。
- *位置*が 0 または 1 の場合は、*挿入するテキスト*は文字列の先頭に挿入されます。

二重引用符を挿入する

*含める文字*の中に二重引用符を指定する場合は、二重引用符を一重引用符で囲む必要があります。

例 : `""`

INT( ) 関数

数式またはフィールド値の整数値を返します。

構文

```
INT(数値)
```

パラメーター

名前	種類	説明
数値	数値	整数に変換するフィールドまたは数式。指定された値が小数を含んでいる場合、小数点以下は四捨五入されないで切り捨てられます。

出力

数値。

例

基本的な例

7 が返されます。

```
INT(7.9)
```

-7 が返されます。

```
INT(-7.9)
```

IPMT( ) 関数

単一の期間で貸付金に対して支払われた利息を返します。

構文

IPMT(*利率*, *指定期間*, *期間*, *金額* <,*種類*>)

パラメーター

名前	種類	説明
利率	数値	1 期あたりの利率。
指定期間	数値	支払利息を確認する期間。
期間	数値	支払期間の総数。
金額	数値	貸付金の元金。
種類 省略可能	数値	支払いのタイミング: ◦ 0 - 期末払い ◦ 1 - 期首払い 支払いのタイミングが省略された場合は、デフォルト値の 0 が使用されます。

> **メモ**
>
> *利率*、*期間*を指定する際には、**1 期あたりの**利率を確実に指定するため、整合性のある期間を使用する必要があります。
>
> 例:
>
> - 年利 5% の 2 年間の貸付金または投資に対して月払いする場合は、*利率*に 0.05/12、期間に 2 * 12 を指定します。
> - 同じ貸付金または投資に対して年払いする場合は、*利率*に 0.05、*期間*に 2 を指定します。

出力

数値。

例

基本的な例

$275,000 の貸付金を年利 6.5 パーセントで 25 年間にわたって返済する場合 (支払期日は月末です) の、1 年目に支払う利息、($) 1489.58 が返されます。

```
IPMT(0.065/12, 1, 12*25, 275000, 0)
```

貸付金の最終月において支払う利息、($) 10.00 が返されます。

```
IPMT(0.065/12, 300, 12*25, 275000, 0)
```

備考

関連する関数

PPMT( ) 関数は IPMT( ) 関数に対して補完的役割を果たします。

CUMIPMT( ) 関数は一連の期間にわたって支払われた利息を計算します。

ISBLANK( ) 関数

入力値が空白かどうか示す論理値を返します。

構文

```
ISBLANK(文字列)
```

パラメーター

名前	種類	説明
文字列	文字	空白データを調べる値。

出力

論理。*文字列*パラメーターの値が空白の場合は**T**(true) 、そうでない場合は**F**(false) を返します。

例

基本的な例

F が返されます。

```
ISBLANK(" A")
```

T が返されます。

```
ISBLANK(" ")
```

```
ISBLANK("")
```

Address フィールド内のすべての値が空白の場合は 'T'、そうでない場合は 'F' が返されます。

```
ISBLANK(Address)
```

備考

ISBLANK( ) の使用に適する場面

ISBLANK( ) は、データが欠落したフィールドを識別するために、分析プロジェクトのデータの整合性フェーズで使用することができます。このようなフィールドは、ソースデータの問題を示している可能性があります。

空白の入力値とは

この関数がtrueと評価されるには、入力値が次のいずれかである必要があります。

- 完全に空白(つまり、スペースのみが含まれる)
- 長さがゼロの文字列

ただし、この関数ではソースデータ内にある真の空白が識別されるだけで、ビュー内で空白として表示される無効な文字は識別されません。

NULL 文字

ISBLANK( ) では、NULL 文字を値として持つ文字フィールドを使用した場合、有効な結果が返されない可能性があります。Analytics では、文字列の終端を示すためにNULL 文字が使用されています。このため、ISBLANK( ) 関数は NULL 文字があった場合、それ以降の文字を空白も含めてすべて読み取りません。

ISDEFINED( ) 関数

指定されたフィールドが定義されている場合は T(true) 、そうでない場合は F(false) を返します。

構文

```
ISDEFINED(文字列)
```

パラメーター

名前	種類	説明
文字列	文字	存在を確認するフィールドまたは変数の名前。値は、引用符で囲んだ文字列として入力する必要があります。 `ISDEFINED("v_numeric_limit")`

出力

論理。

例

基本的な例

v_numeric_limit が変数またはフィールドとして定義されている場合は T、それ以外の場合は F が返されます。

```
ISDEFINED("v_numeric_limit")
```

高度な例

ISDEFINED( ) を使ってフィールドをテストする

次の例では、Limit フィールドの値に基づいてレコードを抽出する前に、このフィールドがテーブルに定義されて

いるかどうかを ISDEFINED( ) 関数を使ってテストしています。

```
OPEN Metaphor_Employees_US
IF ISDEFINED("Limit") EXTRACT RECORD IF Limit > 50000 TO "HighLimit.fil"
```

ISFUZZYDUP( ) 関数

文字列が、比較文字列のあいまい重複であるかどうかを示す論理値を返します。

構文

ISFUZZYDUP(*文字列1*, *文字列2*, *レーベンシュタイン距離* <,*相違のパーセント*>)

パラメーター

名前	種類	説明
文字列1	文字	比較の最初の文字列。
文字列2	文字	比較の2番目の文字列。
レーベンシュタイン距離	数値	2つの文字列があいまい重複と認定されるために、それらの文字列間で許容される最大のレーベンシュタイン距離。 *レーベンシュタイン距離*の値は1未満にすることや、10を超えることはできません。 *レーベンシュタイン距離*の値を大きくすると、あいまい度が高い値(相互の関連性が低い値)が含まれるため、結果の件数が多くなります。
相違のパーセント 省略可能	数値	'相違のパーセント'(相違度)の上限。 相違のパーセントについては、"機能の仕組み" ページ 606を参照してください。 *相違のパーセント*の値は1未満にすることや、99を超えることはできません。 *相違のパーセント*の値を大きくすると、長さに比べて相違の割合が高い値が含まれるようになるため、結果の件数が多くなります。 このパラメーターを省略した場合は、ISFUZZYDUP( ) 関数の処理時に相違のパーセントが考慮されません。

出力

論理。*文字列*の値があいまい重複である場合はT(true)、そうでない場合はF(false)が返されます。

例

基本的な例

"Smith" を "Smythe" に変えるには 2 回の編集が必要ですが、*レーベンシュタイン距離*値が 1 にすぎないため、F が返されます。

```
ISFUZZYDUP("Smith","Smythe", 1, 99)
```

"Smith" を "Smythe" に変えるには 2 回の編集が必要ですが、*レーベンシュタイン距離*値が 2 のため、T が返されます。

```
ISFUZZYDUP("Smith","Smythe", 2, 99)
```

"SMITH" を "smith" に変えるには編集が不要 (ISFUZZYDUP( ) では大文字と小文字が区別されない) であり、*レーベンシュタイン距離*値が 1 のため、T が返されます。

```
ISFUZZYDUP("SMITH","smith", 1, 99)
```

Last_name フィールド内の個々の値が、文字列 "Smith" のあいまい重複であるかどうかを示す論理値 (T または F) が返されます。

```
ISFUZZYDUP(Last_Name,"Smith", 3, 99)
```

高度な例

相違のパーセントの使用方法

相違のパーセントは、ISFUZZYDUP( ) で返される誤検出の数を減らすための手段になります。

*相違のパーセント*を指定しなかった場合

"abc" を "Smith" に変えるには 5 回の編集が必要ですが、*レーベンシュタイン距離*値が 5 のため、T が返されます。

```
ISFUZZYDUP("abc", "Smith", 5)
```

*相違のパーセント*を指定した場合

"abc" が "Smith" から指定したレーベンシュタイン距離内にあっても、*長さ 3 の文字列に対して 5 回の編集*を行うと相違のパーセントが 167% になり、指定した*相違のパーセント*である 99% を超えるため、F が返されます。

```
ISFUZZYDUP("abc", "Smith", 5, 99)
```

相違のパーセントの詳細については、"機能の仕組み" 下を参照してください。

"Smith" に対するあいまい重複の抽出

Last_name フィールド内から "Smith" に対するあいまい重複であるすべての値を抽出するフィルターを作成するには、次のように指定します。

```
SET FILTER TO ISFUZZYDUP(Last_Name, "Smith", 3, 99)
```

*レーベンシュタイン距離*または*相違のパーセント*を変更すると、フィルターされる値の相違の量を調整することができます。

備考

ISFUZZYDUP( ) の使用に適する場面

ISFUZZYDUP( ) 関数は、ほぼ同一の値 (あいまい重複) の検出や、手作業で入力されたデータで一貫性のないつづりを見つける場合に使用できます。

機能の仕組み

ISFUZZYDUP( ) 関数は、2 つの文字列間のレーベンシュタイン距離を計算し、相違のパーセントを求めます。

ISFUZZYDUP( ) は次の場合に T(true) と評価されます。

- 実際のレーベンシュタイン距離が、指定した*レーベンシュタイン距離*の値以下の場合
- 実際の相違のパーセントが、*相違のパーセント*の値 (指定されている場合) 以下の場合

レーベンシュタイン距離

レーベンシュタイン距離は、ある文字列を別の文字列にするために必要な、1 文字の編集の最小回数を示す値です。

詳細については、"LEVDIST( ) 関数" ページ 614を参照してください。

相違のパーセント

相違のパーセントは、異なる 2 つの評価対象文字列のうちの短い方の文字列に対する長い方の文字列のパーセントです。

相違のパーセントは、2 つの文字列間のレーベンシュタイン距離を使用する、次のような Analytics での内部計算の結果です。

レーベンシュタイン距離 / 短い文字列内の文字数 × 100 = 相違のパーセント

任意である相違のパーセントを使用することで、ISFUZZYDUP( ) で返される誤検出の数を減らすことができます。

- *相違のパーセント*の上限は 99% です。99% では、文字列の置き換えがすべて禁止されるため、文字列が同一になります。
- 長さに比べて多数回の編集が必要な文字列は除外されます。

使用上のヒント

- **大文字と小文字の区別** - この関数では大文字と小文字が区別されないため、"SMITH" は "smith" と同じであると判断されます。
- **末尾にあるスペース** - フィールド内の末尾にあるスペースはこの関数により自動的に除去されるため、パラメーターとしてフィールドを指定するときに TRIM( ) 関数を使う必要はありません。
- **一般要素の削除** -OMIT( ) 関数は、フィールド値から "Corporation" や "Inc." などの総称要素を除去することによって、ISFUZZYDUP( ) 関数の有効性を高めることができます 。

  一般的要素の除去により、ISFUZZYDUP( ) の文字列比較は、意味のある違いが発生する可能性のある文字列の部分だけに集中されます。

FUZZYDUP コマンドと ISFUZZYDUP( ) 関数の相違点

FUZZYDUP コマンドでは、フィールド内のすべてのあいまい重複を特定し、それらをグループにまとめて、完全重複を除外した非網羅的な結果セットを出力します。

これに対し、ISFUZZYDUP( ) 関数は、単一の文字値に対するあいまい重複と完全重複を含む網羅的な一覧を特定します。

FUZZYDUP コマンドと ISFUZZYDUP( ) 関数はともに完全重複を特定します。しかし、ISFUZZYDUP( ) 関数を使用する場合には、FUZZYDUP コマンドとは違って完全重複を除外することができません。

「網羅的」とは

「網羅的」とは、検査値から指定された相違の度合いの範囲内であれば、検査値に関連する検査フィールド内の位置に関係なく、すべての値が返されることを意味します。

ISFUZZYDUP( ) 関数は、FUZZYDUP コマンドによって生成された非網羅的な結果では分析目的として十分でない場合や、特定の文字値のあらゆるあいまい重複について直接綿密に調べる必要がある場合に役に立ちます。

関連する関数

- **LEVDIST( )** - は、レーベンシュタイン距離に基づいて文字列を比較するためのもう 1 つの手段です。

  ISFUZZYDUP( ) とは異なり、LEVDIST( ) では既定で大文字と小文字が区別されます。
- **DICECOEFFICIENT( )** - は、文字列を比較するときに、文字または文字ブロックの相対位置を重視しないか、または完全に無視します。
- **SOUNDSLIKE( ) および SOUNDEX( )** - は、正字法の比較 (綴り) ではなく、発音記号の比較 (発音) に基づいて文字列を比較します。

LAST( ) 関数

文字列の末尾から指定された数の文字を返します。

構文

```
LAST(文字列, 長さ)
```

パラメーター

名前	種類	説明
文字列	文字	文字列を返す値。
長さ	数値	返させる文字数。

出力

文字。

例

基本的な例

"Savings" が返されます。

```
LAST("Account Type: Savings", 7)
```

"efghi" が返されます。

```
LAST("abcdefghi", 5)
```

"fghi" が返されます。

```
LAST("abcdefghi ", 5)
```

*文字列*が指定された長さである 6 より短いので、短い分のスペースが出力の先頭に追加されるため、" abc" が返されます。

```
LAST("abc", 6)
```

備考

末尾のスペースによって結果が空白になる場合

*文字列*の末尾にスペースがあるために、LAST( ) 関数によって生成される結果が空白になる場合があります。

たとえば、LAST("6483-30384 ", 3) の出力は " " になります。

ALLTRIM( ) 関数とLAST( ) を組み合わせて使用して*文字列*内の末尾のスペースをすべて取り除くことができます。

たとえば、LAST(ALLTRIM("6483-30384 "), 3) では "384" が返されます。

文字列の先頭にある文字を返す

文字列の先頭にある指定数の文字を返すには、SUBSTR( ) 関数を使用します。詳細については、"SUBSTR( ) 関数" ページ 790を参照してください。

LEADING( ) 関数

指定された数の先頭桁を含んでいる文字列を返します。

構文

```
LEADING(数値, 先頭桁数)
```

パラメーター

名前	種類	説明
数値	数値	先頭桁を返す値。
先頭桁数	数値	返される先頭桁数。

出力

文字。

例

基本的な例

リテラル数値の入力値

623 が返されます。

```
LEADING(6234.56, 3)
```

62345 が返されます。

```
LEADING(6234.56, 3)
```

末尾にゼロを追加する

000 が返されます。

```
LEADING(0.00, 3)
```

00000 が返されます。

```
LEADING(0.00, 5)
```

35500 が返されます。

```
LEADING(3.55, 5)
```

備考

LEADING( ) を使用すると、数値フィールドから数字を文字列として抽出し、小数点やドル記号などの非数値の要素を取り除くことができます。

LENGTH( ) 関数

文字列に含まれている文字数を返します。

構文

```
LENGTH(文字列)
```

パラメーター

名前	種類	説明
文字列	文字	長さを確認する値。

出力

数値。

例

基本的な例

15 が返されます。

```
LENGTH("ABC Corporation")
```

テーブルレイアウト内の **Description** フィールドの、文字数で表した長さを返します。

```
LENGTH(Description)
```

高度な例

住所関連フィールドの各住所の長さを表示する

Vendor_Street フィールド内の各住所の、文字数で表した長さを表示する演算フィールドを作成します。住

所の値から最初に削除される先頭と末尾のスペースは、長さにカウントされません。

```
DEFINE FIELD Address_Length COMPUTED LENGTH(ALLTRIM(Vendor_Street))
```

備考

機能の仕組み

LENGTH( ) 関数はスペースも含めて文字*列*中の文字数を計算して、その数を返します。

末尾のスペース

末尾のスペースは文字としてカウントされます。末尾のスペースをカウントしたくない場合は、TRIM( ) またはALLTRIM( ) 関数を使用して末尾のスペースを削除します。例：

```
LENGTH(TRIM(Vendor_Street))
```

フィールドの値の長さを表示する演算フィールドを作成する場合に、末尾のスペースを削除しないときは、値ごとにフィールドの最大長が表示されます。

LEVDIST( ) 関数

指定された2つの文字列間のレーベンシュタイン距離を返します。これは、2つの文字列がどの程度異なっているかを測る数値です。

構文

LEVDIST(*文字列1*, *文字列2*<,*大文字と小文字の区別*>)

パラメーター

名前	種類	説明
文字列1	文字	比較の最初の文字列。
文字列2	文字	比較の2番目の文字列。
大文字と小文字の区別 省略可能	論理	文字列同士を大文字と小文字を区別して比較する場合はT、大文字と小文字を区別しないで比較する場合はFを指定します。 これを省略した場合は、デフォルト値のTが使用されます。

出力

数値。この値は、2つの文字列間のレーベンシュタイン距離です。

例

基本的な例

"smith" を "Smythe" に変えるには、2回の置換と1回の挿入が必要なため、3が返されます。

```
LEVDIST("smith","Smythe")
```

大文字と小文字が無視されるので、"smith's" を "Smythe" に変えるために必要なのは2回の置換だけのため、2が返されます。

```
LEVDIST("smith's","Smythes",F)
```

Last_name フィールドの各値と文字列 "Smith" の間のレーベンシュタイン距離が返されます。

```
LEVDIST(TRIM(Last_Name),"Smith")
```

高度な例

"Smith" に対する値をランク付けする

"Smith" と **Last_Name** フィールドの各値とのレーベンシュタイン距離を表示する演算フィールド、**Lev_Dist** を作成します。

```
DEFINE FIELD Lev_Dist COMPUTED LEVDIST(TRIM(Last_Name),"Smith", F)
```

ビューに演算フィールド **Lev_Dist** を追加して、この演算フィールドの昇順にクイックソートを行うことで、**Last_name** フィールドのすべての値を "Smith" との相違の量によってランク付けすることができます。

"Smith" に対するあいまい重複の抽出

Last_name フィールドから、"Smith" とのレーベンシュタイン距離が指定された範囲内であるすべての値を抽出するフィルターを作成するには、次のように指定します。

```
SET FILTER TO LEVDIST(TRIM(Last_Name),"Smith", F) < 3
```

式の数字を変更すると、フィルターされる値のレーベンシュタイン距離の量を調整することができます。

備考

LEVDIST( ) の使用に適する場面

LEVDIST( ) 関数は、ほぼ同一の値 (あいまい重複) の検出や、手作業で入力されたデータで一貫性のないつづりを見つける場合に使用できます。また、LEVDIST( ) は完全な重複も識別します。

機能の仕組み

LEVDIST( ) 関数は、2 つの評価される文字列間のレーベンシュタイン距離を返します。レーベンシュタイン距離とは、ある文字列を別の文字列にするために必要な、1 文字の編集の最小回数を示す値です。

必要な編集を行うたびに、レーベンシュタイン距離の値に 1 が加算されます。レーベンシュタイン距離が大きいほど、2 つの文字列間の相違が大きいことになります。距離がゼロ(0) ということは、文字列がまったく同じということです。

編集の種類

編集には次の 3 つの種類があります。

- 挿入
- 削除
- 置換

転置(隣接する2つの文字の位置を入れ替える)は、レーベンシュタインアルゴリズムでは認識されないため、2回の編集、具体的には2回の置換と見なされます。

英数字以外の文字

句読点、特殊文字、および空白は、普通の文字や数字と同様、単一の文字として扱われます。

大文字と小文字

*大文字と小文字の区別*を使用して大小文字の区別をオフにしない限り、文字に使用されている大文字または小文字の変更が1回の置換として数えられます。

文字の位置

レーベンシュタイン距離は文字の位置を考慮に入れます。同じ文字の組み合わせでも文字の順序が異なると、レーベンシュタイン距離が変化する可能性があります。

2が返されます。

```
LEVDIST("abc", "dec")
```

3が返されます。

```
LEVDIST("abc", "dec")
```

LEVDIST( ) とTRIM( ) の併用

LEVDIST( ) を使って"Smith"などのリテラル文字列を文字フィールドと比較する場合、正確な結果を確保するためには、TRIM( ) 関数を使用してフィールドから末尾の空白を除去しておく必要があります。

2つのフィールドを比較する場合は、フィールドごとにTRIM( ) 関数を使用してください。

レーベンシュタインアルゴリズムは空白を文字として数えるため、末尾に空白があると、それらは2つの文字列を同一にするために必要な編集回数の計算にすべて含まれます。

LEVDIST( ) とOMIT( ) の併用

OMIT( ) 関数は、フィールド値から"Corporation"や"Inc."などの総称要素を除去することによって、LEVDIST( ) 関数の有効性を高めることができます。一般的要素の除去により、LEVDIST( ) の文字列比較では、意味のある違いが発生する可能性のある文字列の部分だけが対象になります。

関連する関数

- **ISFUZZYDUP( )** - は、レーベンシュタイン距離に基づいて文字列を比較するためのもう1つの手段です。

  LEVDIST( ) のデフォルトの動作とは異なり、ISFUZZYDUP( ) では大文字と小文字が区別されません。
- **DICECOEFFICIENT( )** - は、文字列を比較するときに、文字または文字ブロックの相対位置を重視しないか、または完全に無視します。
- **SOUNDSLIKE( ) および SOUNDEX( )** - は、正字法の比較 (綴り) ではなく、発音記号の比較 (発音) に基づいて文字列を比較します。

LOG( ) 関数

数式またはフィールド値の対数(底 10) を、指定された小数点以下の桁数で返します。

構文

```
LOG(数値, 小数位)
```

パラメーター

名前	種類	説明
数値	数値	対数を確認する値。
小数位	数値	戻り値の小数点以下の桁数。

出力

数値。

例

基本的な例

3.0000 が返されます。

```
LOG(1000, 4)
```

4.86 が返されます。

```
LOG(72443, 2)
```

高度な例

立方根を求める

フィールド X の立方根を小数点以下 2 桁で表すフィールドを作成するには、次のようにします。

```
DEFINE FIELD Cube_root COMPUTED EXP(LOG(X, 6) / 3, 2)
```

> **メモ**
>
> ある値の指数 *n* の累乗根は、値の対数を *n* で割ってから、その結果の指数を取ることによって求められます。

備考

機能の仕組み

数値の対数とは、その数値を生成するために必要な、10 を底とする指数 (すなわち累乗) です。たとえば、1000 の対数は 3 です。

関連する関数

LOG( ) は EXP( ) 関数の逆関数です。

LOWER( ) 関数

アルファベット文字を小文字に変換した文字列を返します。

構文

```
LOWER(文字列)
```

パラメーター

名前	種類	説明
文字列	文字	小文字に変換する値。

出力

文字。

例

基本的な例

"abc" が返されます。

```
LOWER("ABC")
```

"abc 123 def" が返されます。

```
LOWER("abc 123 DEF")
```

"abcd 12" が返されます。

```
LOWER("AbCd 12")
```

Last_Name フィールドのすべての値を小文字に変換した値が返されます。

```
LOWER(Last_Name)
```

備考

機能の仕組み

LOWER( ) 関数は*文字列*内のすべてのアルファベットを小文字に変換します。アルファベット以外の文字は変換されません。

LOWER( ) の使用に適する場面

LOWER( ) 関数は、大文字と小文字が混在するデータや、どちらの表記が使用されているか不明なデータを検索する場合、または小文字形式のデータが必要な場合に使用できます。

LTRIM( ) 関数

入力文字列から先頭のスペースを除去した文字列を返します。

構文

```
LTRIM(文字列)
```

パラメーター

名前	種類	説明
文字列	文字	先頭スペースを除去する値。

出力

文字。

例

基本的な例

どちらの例も、LTRIM( ) 関数では末尾のスペースが除去されないことに注目してください。

"Vancouver " が返されます。

```
LTRIM("  Vancouver  ")
```

"New York " が返されます。

```
LTRIM("  New York  ")
```

高度な例

改行なしスペースの削除

改行なしスペースはLTRIM( ) 関数で削除されません。

先頭の改行なしスペースを削除する必要がある場合は、次の式を使用して演算フィールドを作成します。

```
DEFINE FIELD Description_cleaned COMPUTED LTRIM(REPLACE(Description, CHR(160), CHR(32)))
```

REPLACE( ) 関数がすべての改行なしスペースを標準のスペースに置換してから、LTRIM( ) がすべての先頭の標準スペースを削除します。

備考

機能の仕組み

LTRIM( ) 関数は先頭スペースだけを除去します。文字列の内部スペースおよび後続スペースは除去されません。

関連する関数

LTRIM( ) の関連関数として、文字列の末尾のスペースを除去する TRIM( ) 関数と、先頭と末尾の両方のスペースを除去する ALLTRIM( ) 関数があります。

MAP( ) 関数

文字列が、ワイルドカード文字やリテラル文字を含んでいる指定された書式文字列と一致するかどうかを示す論理値を返します。

構文

```
MAP(文字列, 書式)
```

パラメーター

名前	種類	説明
文字列	文字	一致をテストするフィールド、式、またはリテラル値。
書式	文字	*文字列*と比較するデータパターンまたは文字列。 *書式*には、ワイルドカード文字、リテラル文字、これら2つの組み合わせを指定できます。 `"\9\9\9-999-9999"` 次のワイルドカード文字がサポートされています。 ◦ `"X"` - すべてのアルファベットに一致します(a ～ z、A ～ Z、欧州文字)。このワイルドカード文字には大文字と小文字の区別はありません。つまり、"X" と "x" のいずれをも使用できます。 ◦ `"9"` - すべての数字に一致します(0 ～ 9)。 ◦ `"!"` - 空白以外のすべての文字に一致します。 ◦ `"?"` - 空白を含むすべての文字に一致します。 ◦ `"\"` - 直後の文字をリテラルとして指定するエスケープ文字。ワイルドカード文字(X、x、9、!、?)のどれかと文字どおり一致させたい場合は、エスケープ文字を使用します。 ◦ `"\\"` - リテラルの円記号(¥)を指定しています。

出力

論理。一致するものが見つかった場合は **T**(true) 、そうでない場合は **F**(false) を返します。

例

基本的な例

シンプルな検索パターン

T が返されます。

```
MAP("ABC Plumbing", "xxx")
```

*文字列*には 4 桁の数字が要求されているのに、3 桁の数字しかないため、F が返されます。

```
MAP("045", "9999")
```

ワイルドカードをエスケープする

リテラル文字 "X" で始まってその後に任意の文字が続く値に対してのみ 'T' を返すことが目的である場合は、*書式*パラメーター "\XX" を使用することで、最初の "X" をワイルドカードとしてでなくリテラルとして解釈されるようにします。

T が返されます。

```
MAP("XA-123", "XX")
```

```
MAP("GC-123", "XX")
```

```
MAP("XA-123", "\XX")
```

F が返されます。

```
MAP("GC-123", "\XX")
```

フィールドとパターン

次の例では、2 つの文字の後に 5 つの数字が続く構成であるか、またはこの構成で始まっている請求書番号が含まれるすべてのレコードに対し、'T' が返されます。それ以外のレコードに対しては 'F' が返されます。

```
MAP(Invoice_Number, "XX99999")
```

次の例では、請求書番号が "AB12345" であるか、または "AB12345" で始まるすべてのレコードに対して、'T' が返されます。それ以外のレコードに対しては 'F' が返されます。

```
MAP(Invoice_Number, "AB12345")
```

次の例では、"AB" の後に5つの数字が続く構成であるか、またはこの構成で始まっている請求書番号が含まれるすべてのレコードに対し、'T' が返されます。それ以外のレコードに対しては 'F' が返されます。

```
MAP(Invoice_Number, "AB99999")
```

次の例では、SSN フィールドの値が社会保障番号の標準形式と一致しないすべてのレコードに対し、'T' が返されます。それ以外のレコードに対しては 'F' が返されます。

```
NOT MAP(SSN, "999-99-9999")
```

高度な例

「859-」で始まる 10 文字の製品コードを持つレコードを抽出する

「859-」で始まる 10 文字以上の製品コードを持つレコードのみを抽出するには、次の IF ステートメントと MAP( ) 関数を使用します。

```
EXTRACT RECORD IF MAP(Product_Code, "85\9-999999") TO "Long_Codes_859"
```

備考

MAP( ) の使用に適する場面

MAP( ) 関数は、英数字データからパターンや特定の書式を検索する場合に使用できます。このパターンや書式にはワイルドカード文字、リテラル文字、またはその双方の組み合わせを指定できます。

大文字と小文字の区別

MAP( ) 関数で 2 つのリテラル値を比較する場合、大文字と小文字は区別されません。たとえば、"a" は "A" と同等ではありません。

*文字列*のデータが大文字と小文字のどちらかに統一されていない場合は、MAP( ) 関数を使用する前に UPPER( ) 関数を使用して、大文字と小文字のどちらかのみになるように値を変換することができます。

例：

```
MAP(UPPER(Invoice_Number), "AB99999")
```

部分一致

MAP( ) では、部分一致がサポートされる場合とサポートされない場合があります。

MAP( ) における部分一致は、**[正確な文字比較を行う]**オプション(SET EXACT ON/OFF) の影響を受けるわけではありません。

部分一致がサポートされる場合とは

部分一致がサポートされるのは、*書式*の値が*文字列*の値より短い場合です。

*書式*が7文字で、*文字列*が9文字のため、T が返されます。

```
MAP("AB1234567", "AB99999")
```

> **メモ**
> T(True) が返されるには、*書式*の値が*文字列*の値の先頭に出現している必要があります。

部分一致がサポートされない場合とは

部分一致がサポートされないのは、*書式*の値が*文字列*の値より長い場合です。

*書式*が7文字で、*文字列*が6文字のため、F が返されます。

```
MAP("AB1234567", "AB99999")
```

*書式*が*文字列*より長い場合には、結果は常にFalse になります。

空白の処理

空白は文字として扱われ、次のいずれかの方法で処理できます。

- 文字どおり空白に一致させる。そのように処理するには、*書式*の値の該当する位置に空白を挿入します。
- ワイルドカード "?" を使用する。これは、空白を含むすべての文字に一致します。

必要に応じて、*文字列*パラメーターに対し、TRIM( )、LTRIM( ) または ALLTRIM( ) 関数を使用して先頭または末尾の空白を除去することにより、テキストの文字とテキスト中の空白のみが比較されるようにすることができます。

フィールドの連結

テーブル内の複数のフィールド内の値を検索したい場合は、2つ以上のフィールドを連結して*文字列*に指定することができます。連結したフィールドは1つのフィールドのように扱われます。ただし、ALLTRIM( ) 関数を使用して個々のフィールドから空白を除去しなければ、各フィールドの先頭と末尾の空白を含むフィールドとなります。

MASK( ) 関数

2 つの文字列の最初の各バイトに対してビットごとの AND 演算を実行します。

構文

```
MASK(文字の値, 文字のマスク)
```

パラメーター

名前	種類	説明
文字の値	文字	バイトを調べる文字列。
文字のマスク	文字	バイトを調べる相手となる文字列 (マスク値) 。

出力

文字。出力は、ビットごとの AND 演算のバイナリ結果に対する文字表記です。

例

基本的な例

3(00110011) と 6(00110110) のビットごとの AND 演算の結果である "2"(00110010) が返されます。

```
MASK("3", "6")
```

備考

MASK( ) の使用に適する場面

MASK( ) 関数は、1 バイト データにおける特定のビット パターン (特定のビットが 1 に設定されているかどうかは問わない) を識別する場合に使用できます。

機能の仕組み

MASK( ) 関数は、*文字の値*と*文字のマスク*の最初の文字のバイナリ表記に対してビットごとのAND論理演算を実行します。比較対象となる2つのバイトが一度に1ビットずつ比較されて、第3のバイナリ値への出力が行われます。

対応する各ビット同士の比較結果は1か0になります。

文字の値のビット	*文字のマスクのビット*	結果
0	0	0
0	1	0
1	0	0
1	1	1

比較対象となる文字列が1バイトより長い

比較対象となる文字列のどちらかが1バイトより長い場合、後続の文字は無視されます。

MATCH( ) 関数

指定された値が比較対象の値のうちのどれかと一致するかどうかを示す論理値を返します。

構文

```
MATCH(比較値, テスト <,...n>)
```

パラメーター

名前	種類	説明
比較値	文字 数値 日付時刻	一致をテストするフィールド、式、またはリテラル値。
テスト <,...n>	文字 数値 日付時刻	*比較値*と比較するフィールド、式、またはリテラル値。 必要な数のテスト値を指定できますが、指定したすべての値は同じデータ型である必要があります。 `MATCH(比較値, `20140930`, `20141030`)`

> **メモ**
>
> MATCH( ) 関数への入力値には文字、数値、日付時刻のデータを指定できます。データ型を混在させることはできません。入力値はすべて同じデータ型に属している必要があります。

出力

論理。1 つでも一致するものが見つかった場合は **T**(true) 、そうでない場合は **F**(false) を返します。

例

基本的な例

> **メモ**
>
> 特に指定のない限り、SET EXACT は OFF(デフォルト設定) であることを前提として、文字比較の値を返します。

リテラル値をテストする

T が返されます。

```
MATCH("ABC", "BCD", "CDE", "AB")
```

F が返されます。

```
MATCH(98, 99, 100, 101)
```

フィールドをテストする

次の例では、**Vendor_City** フィールドの値が "Phoenix"、"Austin"、または "Los Angeles" であるすべてのレコードに対し、T が返されます。それ以外のレコードに対しては 'F' が返されます。

```
MATCH(Vendor_City, "Phoenix", "Austin", "Los Angeles")
```

次の例では、**Vendor_City** フィールドの値が "Phoenix"、"Austin"、または "Los Angeles" でないすべてのレコードに対し、'T' が返されます。それ以外のレコードに対しては 'F' が返されます。

```
NOT MATCH(Vendor_City, "Phoenix", "Austin", "Los Angeles")
```

次の例では、**Vendor_City** フィールドの値が "PHOENIX"、"AUSTIN"、または "LOS ANGELES"(大文字か小文字かを問わない) であるすべてのレコードに対し、'T' が返されます。それ以外のレコードに対しては 'F' が返されます。

Vendor_City フィールドの値が、大文字に変換されてから、大文字の都市名と比較されます。

```
MATCH(UPPER(Vendor_City), "PHOENIX", "AUSTIN", "LOS ANGELES")
```

複数のフィールドをテストする

次の例では、**Vendor_City**、**City**、または **City_2** フィールドの値が "Phoenix" であるすべてのレコードに対し、'T' が返されます。それ以外のレコードに対しては 'F' が返されます。

```
MATCH("Phoenix", Vendor_City, City, City_2)
```

SET EXACT の動作

次の例では、**Product_Code** フィールド内の製品コードが "A"、"D"、または "F" であるか、もしくは "A"、"D"、"F" で始まるすべてのレコードに対し、'T' が返されます。それ以外のレコードに対しては 'F' が返されます。

```
MATCH(Product_Code, "A", "D", "F")
```

次の例では、**Product_Code** フィールド内の製品コードが 1 文字の "A"、"D"、または "F" であるすべてのレコードに対し、'T' が返されます。それ以外のレコードに対しては F が返されます(SET EXACT を ON にしておく必要があります) 。

```
MATCH(Product_Code, "A", "D", "F")
```

2 つのフィールドを比較する

次の例では、業者と従業員の住所が同一の値であるすべてのレコードに対し、'T' が返されます。それ以外のレコードに対しては 'F' が返されます。

業者と従業員の住所の形式を統一する必要がある場合は、その他の関数を使用します。

```
MATCH(Vendor_Address, Employee_Address)
```

日付同士を比較する

次の例では、請求日が 2014 年 9 月 30 日か 2014 年 10 月 30 日であるすべてのレコードに対し、'T' が返されます。それ以外のレコードに対しては 'F' が返されます。

```
MATCH(Invoice_Date, `20140930`, `20141030`)
```

高度な例

異常な在庫レコードを抽出する

Inventory_Value_at_Cost フィールドと演算フィールド **Cost_x_Quantity** の間で金額が異なるすべてのレコードに対し、'T' が返されます。

```
EXTRACT RECORD IF NOT MATCH(Inventory_Value_at_Cost, Cost_x_Quantity) TO "Non_
matching_amounts"
```

部門 101、103、および 107 のレコードを抽出する

以下のようにIF ステートメントとMATCH( ) 関数を使用することで、部門 101、103、または 107 に関連するレコードのみを抽出できます。

```
EXTRACT RECORD IF MATCH(Dept, "101", "103", "107") TO "Three_Departments"
```

備考

OR 演算子の代わりにMATCH( ) を使用する

AND 演算子を使用する式の代わりにMATCH( ) 関数を使用することができます。

例：

```
MATCH(City, "Phoenix", "Austin", "Los Angeles")
```

上記の関数は、次の式と同等です。

```
City="Phoenix" OR City="Austin" OR City="Los Angeles"
```

数値型入力値の小数点以下の精度

比較する数値型入力値の小数点以下の精度が異なる場合、比較は高い方の精度に合わせて行われます。

1.23 は 1.23 と等しいため、T が返されます。

```
MATCH(1.23, 1.23, 1.25)
```

小数点第 3 位まで考慮されるため、1.23 は 1.234 と等しくないと判定されるので、F が返されます。

```
MATCH(1.23, 1.23, 1.25)
```

文字パラメーター

大文字と小文字の区別

文字データが使用される場合、MATCH( ) 関数は大文字と小文字を区別します。文字を比較する場合、"a" は "A" と同等ではありません。

F が返されます。

```
MATCH("a","A","B","C")
```

大文字と小文字のどちらかに統一されていないデータを操作する場合は、MATCH( ) 関数を使用する前に UPPER( ) 関数を使用して、値をすべて大文字か小文字に変換することができます。

T が返されます。

```
MATCH(UPPER("a"), UPPER("A"), UPPER("B"),  UPPER("C"))
```

部分一致

文字比較では、部分一致がサポートされます。比較するどちらか一方の値がもう一方の値の一部である場合に、これらの値が一致と見なされます。

次の例ではいずれも T が返されます。

```
MATCH("AB", "ABC")
```

```
MATCH("ABC", "AB")
```

> **メモ**
> 短い値は、一致を成す長い値の先頭に出現している必要があります。

部分一致と SET EXACT

部分一致は、Analytics のデフォルト設定である SET EXACT = OFF の場合に有効になります。SET EXACT = ON にすると、部分一致は無効になり、比較値が一致を成すには、正確に一致しなければなりません。

SET EXACT が ON の場合、上記の例はどちらも False になります。

SET EXACT([**正確な文字比較を行う**]オプション) の詳細については、"SET コマンド" ページ 404を参照してください。

SET EXACT の OFF または ON への設定

MATCH( ) 関数で [**正確な文字比較を行う**]オプションが使用されないようにしたい場合は、 [**オプション**]ダイアログボックスの [**テーブル**]タブ([**ツール > オプション**]) でこのオプションが選択されていないことを確認してください。

スクリプトを使用している場合は、MATCH( ) 関数が現れる前に SET EXACT OFF コマンドを追加します。必要に応じて、SET EXACT ON コマンドによって前の状態に戻すことができます。

日付時刻パラメーター

関数への入力として指定された日付、日付時刻、または時刻フィールドは、フィールド定義で正しく書式を定義してさえいれば、どのような日付書式、日付時刻書式、または時刻書式でも使用することができます。

日付型、日付時刻型、時刻型の入力値を混在させる

MATCH( ) 関数の入力値に日付値、日付時刻値、および時刻値を混在させることは禁止されていませんが、これら日付時刻のサブタイプを混在させると、意味のない結果が返される可能性があります。

日付時刻値の日付部分のみに関心があっても、時刻部分はまだ計算の一部を構成しているため、Analytics は対応するシリアル値を使用して、日付時刻の計算を処理しています。

次の例で考えてみましょう。

2014 年 12 月 31 日は 2 番目の *テスト*値と一致しているため、T が返されます。

```
MATCH(`20141231`,`20141229`,`20141231`)
```

*比較値*と 2 番目の*テスト*値は同一の日付 2014 年 12 月 31 日を持っているにもかかわらず、F が返されます。

```
MATCH(`20141231 120000`,`20141229`,`20141231`)
```

これら 2 つの式の対応するシリアル値を見ると、どうして 2 番目の式が false と評価されるのかがわかります。

*比較値*のシリアル番号が 2 番目の*テスト*値のシリアル番号と等しいため、T が返されます。

```
MATCH(42003.000000, 42001.000000, 42003.000000)
```

*比較値*のシリアル番号がどの*テスト*値とも等しくないため、F が返されます。

```
MATCH(42003.500000, 42001.000000, 42003.000000)
```

シリアル番号 42003.500000 と42003.000000 の間では、日付部分は一致していますが、時刻部分は一致していません。0.500000 は 12:00 PM に相当するシリアル値です。

日付時刻サブタイプ同士を一致させる

混在する日付時刻サブタイプによって発生する可能性のある問題を回避するには、サブタイプ同士を一致させる関数を使用します。

たとえば、次の式では、上記の 2 番目の式と同じ初期値が使用されていますが、F でなくT が返されます。

```
MATCH(CTOD(DATE(`20141231 120000`,"YYYYMMDD"),"YYYYMMDD"),`20141229`,
`20141231`)
```

リテラル日付、日付時刻、または時刻値の指定

関数への入力のいずれかにリテラルの日付値、日付時刻値、または時刻値を指定する場合は、次の表内の書式に制限されます。また、`20141231` のように、値を逆引用符で囲む必要があります。

日付または時刻の個々の要素の間に、スラッシュ(/) やコロン(:) のような区切り文字をいっさい使用しないでください。

- **日付時刻値** - 以下の表に示す日付、区切り文字、および時刻の書式を任意に組み合わせて使用することができます。日付を時刻の前に置く必要があり、かつ、2 つの間に区切り文字を使用する必要があります。有効な区切り文字は空白スペース 1 つ、あるいは文字 't' または 'T' です。
- **時刻値** - 24 時間形式で時刻を指定する必要があります。UTC(Coordinated Universal Time: 協定世界時) からのオフセットは、プラス記号(+) またはマイナス記号(-) で始める必要があります。

形式の例	リテラル値の例
YYYYMMDD	`20141231`
YYMMDD	`141231`
YYYYMMDD hhmmss	`20141231 235959`
YYMMDDthhmm	`141231t2359`
YYYYMMDDThh	`20141231T23`
YYYYMMDD hhmmss+/-hhmm (UTC オフセット)	`20141231 235959-0500`
YYMMDD hhmm+/-hh (UTC オフセット)	`141231 2359+01`
thhmmss	`t235959`
Thhmm	`T2359`
メモ UTC オフセットが設定されているデータのメインの時刻書式で hh を単独で使用しないでください。たとえば、「hh+hhmm」という使い方は避けてください。信頼できない結果になる可能性があります。	

MAXIMUM( ) 関数

一連の数値中の最大値、または一連の日付時刻値中の最も新しい値を返します。

構文

```
MAXIMUM(値1, 値2<,...n>)
```

パラメーター

名前	種類	説明
値1、値2 <, ...n>	数値 日付時刻	カンマによって区切られた、比較し合う値。 すべての値は同じデータ型である必要があります。 また、日付時刻値は同じサブタイプである必要があります。この関数の 1 回の実行において、日付値、日付時刻値、時刻値を混在させることはできません。

出力

数値または日付時刻。

例

基本的な例

リテラル数値の入力値

7 が返されます。

```
MAXIMUM(4, 7)
```

8 が返されます。

```
MAXIMUM(4, 7, 3, 8)
```

8.00 が返されます。

```
MAXIMUM(4, 7.25, 3, 8)
```

リテラル日付時刻の入力値

`20161231` が返されます。

```
MAXIMUM(`20161231`, `20161229`, `20161230`)
```

`20161231 23:59:59` が返されます。

```
MAXIMUM(`20161231 235959`, `20161229 235959`)
```

`23:59:59` が返されます。

```
MAXIMUM(`.235957`, `.235959`, `.235958`)
```

フィールドへの入力値

各レコードについて、次の 3 つのフィールドのうち、最も新しい日付が返されます。

```
MAXIMUM(PO_Date, Invoice_Date, Payment_Date)
```

高度な例

演算フィールドを作成してデフォルトの最小額を設定する

期限経過勘定のテーブルに**支払利息**という演算フィールドを作成してデフォルトの最小額 $1.00 を設定するには、次の式を使用します。

```
DEFINE FIELD Interest_Due COMPUTED MAXIMUM(BALANCE * ANNUAL_RATE, 1)
```

残高に利率を掛けた値が $1.00 より小さい場合、MAXIMUM( ) 関数は 1 を返します。そうでない場合、MAXIMUM( ) は計算した利息を返します。

四半期の終了日より後の日付を検出する

複数のフィールドにまたがる日付が四半期の終了日を過ぎているかどうかを判断するには、下記のような式を使用した演算フィールドを作成します。

```
DEFINE FIELD Past_Qtr COMPUTED MAXIMUM(PO_Date, Invoice_Date, Payment_Date,
`20160331`)
```

- 2016 年 3 月 31 日以前の日付が含まれるすべてのレコードに対し、`20160331` が返されます。
- 2016 年 3 月 31 日より後の日付が含まれるレコードに対しては、上記 3 つのフィールドのうち、最新の日付が返されます。

備考

数値の比較における小数点以下の桁数の動作

比較し合う数値の小数点以下の桁数が異なる場合、比較結果は最も大きい小数点以下の桁数に合わせて調整されます。

次の例では 20.400 が返されます。

```
MAXIMUM(3.682, 10.88, 20.4)
```

DECIMALS( ) 関数を使用して、*値* パラメーターの小数点以下の桁数を調整することができます。

次の例では 20.40 が返されます。

```
MAXIMUM(DECIMALS(3.682, 2), 10.88, 20.4)
```

MINIMUM( ) 関数

一連の数値中の最小値、または一連の日付時刻値中の最も古い値を返します。

構文

```
MINIMUM(値1, 値2<,...n>)
```

パラメーター

名前	種類	説明
値1、値2<, ...n>	数値 日付時刻	カンマによって区切られた、比較し合う値。 すべての値は同じデータ型である必要があります。 また、日付時刻値は同じサブタイプである必要があります。この関数の1回の実行において、日付値、日付時刻値、時刻値を混在させることはできません。

出力

数値または日付時刻。

例

基本的な例

リテラル数値の入力値

4 が返されます。

```
MINIMUM(4, 7)
```

3 が返されます。

```
MINIMUM(4, 7, 3, 8)
```

3.00 が返されます。

```
MINIMUM(4, 7.25, 3, 8)
```

リテラル日付時刻の入力値

`20161229` が返されます。

```
MINIMUM(`20161231`, `20161229`, `20161230`)
```

`20161229 23:59:59` が返されます。

```
MINIMUM(`20161231 235959`, `20161229 235959`)
```

`23:59:57` が返されます。

```
MINIMUM(`.235957`, `.235959`, `.235958`)
```

フィールドへの入力値

各レコードについて、次の 3 つのフィールドのうち、最も古い日付が返されます。

```
MINIMUM(PO_Date, Invoice_Date, Payment_Date)
```

高度な例

複数のフィールドの値のうち、最も低い値を識別する

"Cost"(費用) 、**"Sale_Price**(販売価格) 、**"Discount_Price"**(割引価格) フィールドの値のうち、最も低い値を識別するには、次の式を使用した演算フィールドを作成します。

```
DEFINE FIELD Low_Value COMPUTED MINIMUM(Cost, Sale_Price, Discount_Price)
```

四半期の開始日より前の日付を検出する

複数のフィールドにまたがる日付が四半期の開始日より前であるかどうかを判断するには、下記のような式を使用した演算フィールドを作成します。

```
DEFINE FIELD Pre_Qtr COMPUTED MINIMUM(PO_Date, Invoice_Date, Payment_Date,
`20160101`)
```

- 2016 年 1 月 1 日以後の日付が含まれるレコードに対し、`20160101` が返されます。
- 2016 年 1 月 1 日より前の日付が含まれるレコードに対しては、上記 3 つのフィールドのうち、最も古い日付が返されます。

備考

数値の比較における小数点以下の桁数の動作

比較し合う数値の小数点以下の桁数が異なる場合、比較結果は最も大きい小数点以下の桁数に合わせて調整されます。

次の例では 3.600 が返されます。

```
MINIMUM(3.6,10.88, 20.482)
```

DECIMALS( ) 関数を使用して、*値* パラメーターの小数点以下の桁数を調整することができます。

次の例では 3.60 が返されます。

```
MINIMUM(3.6,10.88, DECIMALS(20.482, 2))
```

省略形 MIN( )

ACLScript では、省略形の MIN は関数を一意に識別しないにもかかわらず、MINIMUM( ) 関数の省略形として使用することができます。これは、関数名を省略するときの規定の要件です。

MIN() は MINUTE( ) の省略形である可能性も考えられますが、Analytics では、省略形 MIN() を MINIMUM ( ) 関数のために取ってあります。

MINUTE( ) 関数

指定された時刻または日付時刻から分数を抽出し、それを数値として返します。

構文

```
MINUTE(時刻/日付時刻)
```

パラメーター

名前	種類	説明
時刻/日付時刻	日付時刻	分数を抽出するフィールド、式、またはリテラル値。

出力

数値。

例

基本的な例

59 が返されます。

```
MINUTE(`t235930`)
```

```
MINUTE(`20141231 235930`)
```

Call_start_time フィールドの各値の分数が返されます。

```
MINUTE(Call_start_time)
```

備考

スクリプトにおける MINUTE( ) の省略形

ACLScript で MINUTE( ) 関数を短縮する場合は、少なくとも最初の 4 文字 (MINU) を使用する必要があります。Analytics では、MINIMUM( ) 関数の省略形 MIN が予約されています。

パラメーターの詳細

*時刻/日付時刻*に指定されたフィールドは、フィールド定義で正しく書式を定義してさえいれば、どのような時刻書式または日付時刻書式でも使用することができます。

リテラル時刻または日付時刻値の指定

*日付時刻*にリテラルの時刻値または日付時刻値を指定する場合は、次の表内の書式に制限されます。また、`20141231 235959` のように、値を逆引用符で囲む必要があります。

日付または時刻の個々の要素の間に、スラッシュ(/) やコロン(:) のような区切り文字をいっさい使用しないでください。

- **時刻値** - 以下の表に示す任意の時刻の書式を使用することができます。関数が正しく動作するためには、単独の時刻値の前に区切り文字を使用する必要があります。有効な区切り文字は文字 't' または 'T' です。24 時間形式で時刻を指定する必要があります。UTC(Coordinated Universal Time：協定世界時) からのオフセットは、プラス記号 (+) またはマイナス記号 (-) で始める必要があります。
- **日付時刻値** - 以下の表に示す日付、区切り文字、および時刻の書式を任意に組み合わせて使用することができます。日付を時刻の前に置く必要があり、かつ、2 つの間に区切り文字を使用する必要があります。有効な区切り文字は空白スペース 1 つ、あるいは文字 't' または 'T' です。

形式の例	リテラル値の例
thhmmss	`t235959`
Thhmm	`T2359`
YYYYMMDD hhmmss	`20141231 235959`
YYMMDDthhmm	`141231t2359`
YYYYMMDDThh	`20141231T23`
YYYYMMDD hhmmss+/-hhmm (UTC オフセット)	`20141231 235959-0500`
YYMMDD hhmm+/-hh (UTC オフセット)	`141231 2359+01`

形式の例	リテラル値の例
メモ UTC オフセットが設定されているデータのメインの時刻書式で hh を単独で使用しないでください。たとえば、「hh+hhmm」という使い方は避けてください。信頼できない結果になる可能性があります。	

MOD( ) 関数

2 つの数値を除算して余りを返します。

構文

```
MOD(数値, 除数)
```

パラメーター

名前	種類	説明
数値	数値	除算する数値。
除数	数値	*数値*を除算する数値。 *数値*または*除数*、あるいはこれら両方が小数を含んでいる場合、出力の小数点以下の精度は、入力値で小数点以下の桁数が多い方と同じになります。たとえば、MOD(45.35, 5.3) の出力は 2.95 です。

出力

数値。

例

基本的な例

3 が返されます。

```
MOD(93, 10)
```

2.0 が返されます。

```
MOD(66, 16.00)
```

3.45 が返されます。

```
MOD(53.45, 10)
```

高度な例

創業記念日を計算する

前回の創業記念日からの経過月数を表示するフィールドを定義するには、次のように指定します。

```
DEFINE FIELD Months_since_last_anniversary COMPUTED MOD(Months_of_service, 12)
```

備考

MOD( ) の使用に適する場面

MOD( ) 関数は、2 つの数値が等分できるかどうかをテストしたり、除算の余りを分離させたりする場合に使用します。この関数は、一方の数値を別の数値で割り、その余りを返します。

MONTH( ) 関数

指定された日付または日付時刻から月を抽出し、それを数値(1 ～ 12)として返します。

構文

```
MONTH(日付/日付時刻)
```

パラメーター

名前	種類	説明
日付/日付時刻	日付時刻	月を抽出するフィールド、式、またはリテラル値。

出力

数値。

例

基本的な例

12 が返されます。

```
MONTH(`20141231`)
```

```
MONTH(`20141231 235959`)
```

Invoice_date フィールドの各値に対して月が返されます。

```
MONTH(Invoice_date)
```

備考

パラメーターの詳細

*日付/日付時刻*に指定されたフィールドは、フィールド定義で正しく書式を定義してさえいれば、どのような日付書式または日付時刻書式でも使用することができます。

リテラル日付または日付時刻値の指定

*日付/日付時刻*にリテラルの日付値または日付時刻値を指定する場合は、次の表内の書式に制限されます。また、`20141231` のように、値を逆引用符で囲む必要があります。

日付または時刻の個々の要素の間に、スラッシュ(/) やコロン(:) のような区切り文字をいっさい使用しないでください。

- **日付時刻値** - 以下の表に示す日付、区切り文字、および時刻の書式を任意に組み合わせて使用することができます。日付を時刻の前に置く必要があり、かつ、2 つの間に区切り文字を使用する必要があります。有効な区切り文字は空白スペース 1 つ、あるいは文字 't' または 'T' です。
- **時刻値** - 24 時間形式で時刻を指定する必要があります。UTC(Coordinated Universal Time: 協定世界時) からのオフセットは、プラス記号 (+) またはマイナス記号 (-) で始める必要があります。

形式の例	リテラル値の例
YYYYMMDD	`20141231`
YYMMDD	`141231`
YYYYMMDD hhmmss	`20141231 235959`
YYMMDDthhmm	`141231t2359`
YYYYMMDDThh	`20141231T23`
YYYYMMDD hhmmss+/-hhmm (UTC オフセット)	`20141231 235959-0500`
YYMMDD hhmm+/-hh (UTC オフセット)	`141231 2359+01`
メモ UTC オフセットが設定されているデータのメインの時刻書式で hh を単独で使用しないでください。たとえば、「hh+hhmm」という使い方は避けてください。信頼できない結果になる可能性があります。	

関連する関数

月の名前を返したい場合は、MONTH( ) でなくCMOY( ) を使用してください。

NOMINAL( ) 関数

貸付金の名目年利率を返します。

構文

```
NOMINAL(実効利率, 期間)
```

パラメーター

名前	種類	説明
実効利率	数値	実効年利率。
期間	数値	年間の複利計算回数。 **メモ** 整数を指定してください。小数部分を指定しても切り捨てられるためです。

出力

数値。利率は小数点以下 8 桁で計算されます。

例

基本的な例

19.56% の実効年利を請求されるクレジット カードの未払い分に対し、名目年利率 0.17998457(18%) が返されます。

```
NOMINAL(0.1956, 12)
```

備考

名目年利率とは

貸付金の名目年利率とは、未払い額に対する利息を考慮せずに毎月または毎日複利計算が行われると仮定した場合の名目利率または公示利率です。

関連する関数

EFFECTIVE( ) 関数は NOMINAL( ) 関数の逆関数です。

NORMDIST( ) 関数

正規分布データセットのランダム変数が指定した値以下になるか、指定した値とまったく同じになる確率を返します。

構文

```
NORMDIST(x, 平均, 標準偏差, 累計)
```

パラメーター

名前	種類	説明
x	数値	確率を計算する値。
平均	数値	データセットの平均値。
標準偏差	数値	データセットの標準偏差。*標準偏差*値は 0 より大きくなければなりません。
累計	論理	T を指定すると、ランダム変数が *x* 以下である確率(累計確率) を計算します。F を指定すると、ランダム変数が *x* に等しい確率(単純な確率) を計算します。

出力

数値。

例

基本的な例

0.908788780274132 が返されます。

```
NORMDIST(42, 40, 1.5, T)
```

0.109340049783996 が返されます。


```
NORMDIST(42, 40, 1.5, F)
```

NORMSINV( ) 関数

標準正規分布で指定された確率に関連付けられたｚスコアを返します。ｚスコアは、標準正規分布の平均を基準にした、値の標準偏差数です。

構文

```
NORMSINV(確率)
```

パラメーター

名前	種類	説明
確率	数値	ｚスコアを計算する確率。

出力

数値。

例

基本的な例

1.333401745213610 が返されます。

```
NORMSINV(0.9088)
```


NOW( ) 関数

現在のオペレーティングシステム時刻を日付時刻データ型で返します。

構文

```
NOW()
```

パラメーター

この関数にはパラメーターはありません。

出力

日付時刻。

例

基本的な例

現在のオペレーティングシステム時刻が、現在の Analytics の時刻表示書式を使用して表示される日付時刻値 (`t235959` など) として返されます。

```
NOW()
```

備考

関連する関数

現在のオペレーティングシステム時刻を文字列として返させたい場合には、NOW( ) でなく TIME( ) を使用してください。

NPER( ) 関数

貸付金を完済するのに必要な期間の数値を返します。

構文

NPER(*利率*, *支払金額*, *金額* <,*種類*>)

パラメーター

名前	種類	説明
利率	数値	1 期あたりの利率。
支払金額	数値	1 期間あたりの支払金額。
金額	数値	貸付金の元金。
種類 省略可能	数値	支払いのタイミング: ◦ 0 - 期末払い ◦ 1 - 期首払い 支払いのタイミングが省略された場合は、デフォルト値の 0 が使用されます。

出力

数値。

例

基本的な例

毎月末までに$1,856.82 を支払うことによって年利 6.5% の貸付金、$275,000 を完済するのに必要な月数、300.00 が返されます。

```
NPER(0.065/12, 1856.82, 275000, 0)
```

毎月末までに$2,000 を支払うことによって同じ貸付金を完済するのに必要な月数、252.81 が返されます。

```
NPER(0.065/12, 1856.82, 275000, 0)
```

毎月月初までに$2,000を支払うことによって同じ貸付金を完済するのに必要な月数、249.92が返されます。

```
NPER(0.065/12, 1856.82, 275000, 0)
```

高度な例

年金の計算

年金の計算では、次の4つの変数が使用されます。

- **現在価値または将来価値** - 下の例では$21,243.39と$ 26,973.46
- **1期間あたりの支払金額** - 下の例では$1,000.00
- **1期間あたりの利率** - 下の例では月あたり1%
- **期間の数** - 下の例では24か月

これらの変数のうち3つの値がわかっている場合は、Analytics関数を使って残りの変数の値を計算できます。

求めたい値：	使用するAnalytics関数
現在価値	PVANNUITY( ) 21243.39を返す： `FVANNUITY(0.01, 12, 1000)`
将来価値	FVANNUITY( ) 26973.46を返す： `FVANNUITY(0.01, 24, 1000)`
1期間あたりの支払金額	PMT( ) 1000を返す： `PMT(0.01, 24, 21243.39)`
1期間あたりの利率	RATE( ) 0.00999999(1%)を返す： `RATE(24, 1000, 21243.39)`

求めたい値：	使用する Analytics 関数
期間の数	NPER( ) 24.00 を返す： NPER(0.01, 1000, 21243.39)

年金の式

期末年金(期末払い)の**現在価値**を計算する式：

期末年金(期末払い)の**将来価値**を計算する式：

OCCURS( ) 関数

部分文字列が指定された文字値内に現れる回数を数えて返します。

構文

```
OCCURS(文字列, 検索語)
```

パラメーター

名前	種類	説明
文字列	文字	検索される値。 テーブル内の複数のフィールド内の値を検索したい場合は、2 つ以上のフィールドを連結することができます。 `OCCURS(First_Name+Last_Name,"John")`
検索語	文字	検索対象となる値。検索は大文字と小文字を区別します。

出力

数値。

例

基本的な例

2 が返されます。

```
OCCURS("abc/abc/a","ab")
```

3 が返されます。

```
OCCURS("abc/abc/a","a")
```

Invoice_Number フィールド内の各値におけるハイフンの出現回数が返されます。

```
OCCURS(Invoice_Number, "-")
```

高度な例

ハイフンが 2 つ以上入っている請求書番号を見つける

テーブル内の Invoice_Number にハイフンが 1 つしか入ってはいけない場合は、OCCURS( ) 関数を使って、2 つ以上のハイフンが含まれる Invoice_Number を抽出するフィルターを作成します。

```
SET FILTER TO OCCURS(Invoice_Number, "-") > 1
```

あるフィールドの値の他フィールドへの出現箇所を見つける

あるフィールドの値の他フィールドへの出現箇所を見つけるには、OCCURS( ) を使用します。たとえば、**Last_Name** の値が **Full_Name** フィールドに出現しているレコードを抽出するフィルターを作成できます。

```
OCCURS(Full_Name, ALLTRIM(Last_Name)) = 1
```

式に ALLTRIM( ) 関数を追加すると、**Last_Name** フィールドの先頭と末尾のスペースがすべて取り除かれ、テキスト値のみが比較されるようになります。

大文字と小文字を区別する検索を実行する

FIND( ) 関数とは異なり、OCCURS( ) 関数は大文字と小文字を区別するため、大小文字を区別した検索を実行できます。

次の式は、**Vendor_Name** フィールドに大文字の "UNITED EQUIPMENT"という名前が入っているすべてのレコードを抽出します("United Equipment" の出現箇所は無視) 。

```
SET FILTER TO OCCURS(UPPER(Vendor_Name), "UNITED EQUIPMENT") > 0
```

大文字と小文字の違いを無視して "United Equipment" のすべての出現箇所を見つけたい場合は、UPPER( ) 関数を使って検索フィールドの値を大文字に変換します。

```
OCCURS(Vendor_Name, "UNITED EQUIPMENT") > 0
```

OFFSET( ) 関数

指定されたバイト数によって補正された開始位置からフィールドの値を返します。

構文

```
OFFSET(フィールド, バイト数)
```

パラメーター

名前	種類	説明
フィールド	文字 数値 日付時刻	フィールド名。
バイト数	数値	任意の正の数式。

出力

戻り値は、入力の*フィールド* パラメーターと同じデータ型になります。

例

基本的な例

"Number" というフィールドに値 "1234567890" が入っており、"Offset_Number" という開始位置 1、長さ 3、小数位なしの重複フィールドを定義するとします。OFFSET( ) 関数を使用して、フィールド内の数字の位置を変えることができます。

123 が返されます。

```
OFFSET(Offset_Number,0)
```

234 が返されます。

```
OFFSET(Offset_Number,1)
```

789 が返されます。

```
OFFSET(Offset_Number,6)
```

備考

フィールドの開始位置の一時的な補正としてこの関数を使用できます。フィールドの開始位置が可変であるデータを処理している場合、これは有用です。

条件演算フィールドで OFFSET( ) 関数を使用する場合、IF テストで参照されるフィールドも補正される点に注意してください。

OMIT( ) 関数

指定した1つ以上の部分文字列が削除された文字列を返します。

構文

OMIT(*文字列1*, *文字列2*<,*大文字と小文字の区別*>)

パラメーター

名前	種類	説明
文字列1	文字	1つ以上の部分文字列を削除する対象となる文字列。
文字列2	文字	削除対象となる1つ以上の部分文字列。 ◦ 複数の部分文字列を区切るにはカンマを使用します。 ◦ その部分文字列が削除したい部分文字列の一部である場合のみ、カンマの後にスペースを使用します。 ◦ 部分文字列のいずれかに二重引用符文字が出現している場合は、*文字列2*パラメーターの前後を二重引用符でなく一重引用符で囲む必要があります(' ') 。 ◦ カンマを削除するには、部分文字列のリストの最後に単一のカンマを置き、その直後に閉じる引用符を続けます(下記の最後の例を参照してください) 。
大文字と小文字の区別 省略可能	論理	部分文字列で大文字と小文字が区別されるようにするにはTを指定し、大文字と小文字の違いが無視されるようにするにはFを指定します。 *大文字と小文字の区別*を省略した場合は、デフォルト値のTが使用されます。

出力

文字。

例

基本的な例

リテラル文字の入力値

"Intercity Couriers" が返されます。

```
OMIT("Intercity Couriers Corporation", " Corporation, Corp.")
```

"Inter-city Couriers" が返されます。

```
OMIT("Inter-city Couriers Corp.", " Corporation, Corp.")
```

> **メモ**
>
> この戻り値と上記の例の戻り値との間のレーベンシュタイン距離は 1 です。一般的要素が削除されていない場合、2 つの例の間の距離は 8 となり、値があいまい重複としての検出を逃れる可能性があります。

フィールドへの入力値

"Corporation" および "Inc." などの一般的な要素を含む**業者名**フィールドのすべての値を返します。削除済み:

```
OMIT(Vendor_Name," Corporation, Corp., Corp, Inc., Inc, Ltd., Ltd")
```

"Corporation" および "Inc." などの一般的な要素を含む**業者名**フィールドのすべての値を返します。削除済み:

```
OMIT(Vendor_Name," ,.,Corporation,Corp,Inc,Ltd")
```

> **メモ**
>
> 上記の 2 つの例は同じ結果を返しますが、2 番目の例の構文の方がより効率的です。

Vendor_Name フィールドのすべての値から "Corporation" や "Corp" などの要素とすべてのカンマを削除した値が返されます。

```
OMIT(Vendor_Name," Corporation, Corp,")
```

備考

OMIT( ) により、部分文字列ごとに削除を行うことができます。

OMIT( ) 関数は、文字列から 1 つ以上の部分文字列を削除します。これは、CLEAN( )、EXCLUDE( )、INCLUDE( )、および REMOVE( ) などの関数とは異なり、文字単位ではなく、部分文字列単位で文字列と一致し、削除を行います。サブ文字列の削除により、文字列から特定の単語、略語、あるいは繰り返し現れる文字の並びを、文字列の残り部分に影響を与えることなく削除することができます。

あいまい比較用のヘルパー関数

OMIT( ) 関数は、"Corporation" や "Inc." などの総称要素を除去することによって、LEVDIST( ) 関数や ISFUZZYDUP( ) 関数、あるいは FUZZYDUP または FUZZYJOIN コマンドの有効性を高めることができます。。総称要素の除去により、これらの関数およびコマンドで実行される文字列比較は、意味のある違いが発生する可能性のある文字列の部分だけに集中されます。

部分文字列間の順序が結果に及ぼす影響

削除する部分文字列を複数指定する場合、*文字列2* にそれらを列挙した順序が出力結果に影響を与えることがあります。

OMIT( ) 関数の処理時、最初の部分文字列がそれを含んでいるすべての値から削除され、次に 2 番目の部分文字列がそれを含んでいるすべての値から削除され、というように続きます。もし、ある部分文字列が別の文字列の一部を構成している場合、たとえば "Corp" と "Corporation" という場合には、先に短い部分文字列を削除すると、長い部分文字列を含んでいる値も変えてしまい("Corporation" が "oration" になります)、長い部分文字列が検出されなくなります。

このような状況を回避するために、長い部分文字列を、それに含まれるあらゆる短い部分文字列よりも先に指定してください。例：

```
OMIT(Vendor_Name," Corporation, Corp., Corp")
```

まず特殊文字を削除するようにしてください。

句読点、特殊文字、およびスペースなどの単一文字の部分文字列を指定して、文字列内の一般的な構成要素をさらに減らすことができます。最初にピリオドや空白などの単一文字を削除する方が効率的かもしれません。これにより、その後指定する必要のある部分文字列の変化形の数が減ります。上記の 3 番目と 4 番目の例を比較してください。どちらも同じ結果を返しますが、4 番目の例の方がより効率的です。

空白やスペースの扱い

部分文字列内の空白やスペースは他の文字と同様に扱われます。サブ文字列の一部として削除する各スペースを明示的に指定する必要があります。たとえば、空白をまったく入れないでアンパサンドを指定した場合("&")、"Ricoh Sales & Service" は "Ricoh Sales Service" になります。空白を入れた場合(" & ")、"Ricoh Sales & Service" は "Ricoh SalesService" になります。

サブ文字列の一部ではない空白を指定する場合、サブ文字列は検索されません。たとえば、アンパサンドと空白(" & ") を指定する場合、"Ricoh Sales&Service" は変更されません。

カンマを使って複数の部分文字列を区切る場合、カンマの後にスペースを入れるのは、そうすることで、削除したい実際の部分文字列と一致するときのみです。

空白を扱う手法の1つは、最初にフィールドからすべての空白を削除することです。これは、ほかのどの部分文字列を指定するより先に、空白を単一文字の部分文字列として指定することで行えます。

OMIT( ) を使用した結果を見直す

OMIT( ) を使って演算フィールドを作成した後、そのフィールドの内容を見直して、文字列の意味のある部分を不注意に削除していないかを確認してください。たとえば、"Co" を削除すると、"Company" に対してよく用いられる略語(Co) が取り除かれますが、それはまた、"Coca-Cola" の2か所から文字 "Co" を除くことにもなります。

PACKED( ) 関数

パック データ型に変換された数値データを返します。

構文

```
PACKED(数値, 結果の長さ)
```

パラメーター

名前	種類	説明
数値	数値	変換する数値またはフィールド。
結果の長さ	数値	出力文字列で使用するバイトの数。

出力

数値。

例

基本的な例

整数と小数の入力値

00075C が返されます。

```
PACKED(75, 3)
```

```
PACKED(7.5, 3)
```

桁を切り捨てて出力する

00000012456D が返されます。


```
PACKED(-12.456, 6)
```

456D が返されます。

```
PACKED(-12.456, 2)
```

高度な例

メインフレームを更新する 8 バイトのフィールドを作成する

メインフレームへのアップロード対象として、各従業員の給与をパック型の数値にした 8 バイトのフィールドを作成する場合は、次のように指定します。

```
EXTRACT PACKED(SALARY, 8) AS "Salary_Export" TO "export"
```

備考

パック データとは

パック データ型は、メインフレーム オペレーティング システムで使用されるデータ型で、最小の記憶域を使用する形式で数値を格納します。パック データ型は 1 バイトにつき 2 桁の数字を格納し、最終バイトは、値が正負いずれであるかを示します。

PACKED( ) の使用に適する場面

PACKED( ) 関数は、数値データをメインフレーム システムへエクスポートするために、数値データをパックされた形式に変換する場合に使用できます。

戻り値が切り捨てられる場合とは

*結果の長さ*の値が*数値*の長さより短い場合は、余分な桁が切り詰められます。

PI( ) 関数

円周率 π の値を、小数点以下 15 桁で返します。

構文

```
PI( )
```

パラメーター

この関数にはパラメーターはありません。

出力

数値。

例

基本的な例

3.141592653589793(小数点以下 15 桁での円周率 π の値) が返されます。

```
PI( )
```

60 度のラジアンに相当する 1.047197551196598 が返されます。

```
60 * PI( )/180
```

高度な例

入力値として度を使用する

60 度の正弦に相当する 0.866025403784439 が返されます。

```
SIN(60 * PI( )/180)
```

備考

PI( ) の使用に適する場面

PI( ) 関数は、角度をラジアンに変換する場合に使用できます。(度 * PI( )/180) = ラジアンとなります。ラジアンは、Analytics の 3 つの数学関数 SIN( )、COS( )、TAN( ) で入力が必須です。

PMT( ) 関数

貸付金を完済するのに必要な定期払込金額(元金 + 利息)を返します。

構文

PMT(*利率*, *期間*, *金額* <,*種類*>)

パラメーター

名前	種類	説明
利率	数値	1 期あたりの利率。
期間	数値	支払期間の総数。
金額	数値	貸付金の元金。
種類 省略可能	数値	支払いのタイミング: ◦ 0 - 期末払い ◦ 1 - 期首払い 支払いのタイミングが省略された場合は、デフォルト値の 0 が使用されます。

> **メモ**
>
> *利率*、*期間*を指定する際には、**1 期あたりの**利率を確実に指定するため、整合性のある期間を使用する必要があります。
>
> 例:
>
> - 年利 5% の 2 年間の貸付金または投資に対して月払いする場合は、*利率*に 0.05/12、期間に 2 * 12 を指定します。
> - 同じ貸付金または投資に対して年払いする場合は、*利率*に 0.05、*期間*に 2 を指定します。

出力

数値。

例

基本的な例

$275,000 の貸付金を年利 6.5 パーセントで 25 年間で完済する場合 (支払期日は月末です) の月次支払額 (元金 + 利息) 、($) 1856.82 が返されます。

```
PMT(0.065/12, 12*25, 275000, 0)
```

同じ貸付金を完済する場合 (支払期日は月初です) の月次支払額 (元金 + 利息) 、($) 1846.82 が返されます。

```
PMT(0.065/12, 12*25, 275000, 1)
```

高度な例

年金の計算

年金の計算では、次の 4 つの変数が使用されます。

- **現在価値または将来価値** - 下の例では $21,243.39 と $ 26,973.46
- **1 期間あたりの支払金額** - 下の例では $1,000.00
- **1 期間あたりの利率** - 下の例では月あたり 1%
- **期間の数** - 下の例では 24 か月

これらの変数のうち 3 つの値がわかっている場合は、Analytics 関数を使って残りの変数の値を計算できます。

求めたい値:	使用する Analytics 関数
現在価値	PVANNUITY( ) 21243.39 を返す: `FVANNUITY(0.01, 12, 1000)`
将来価値	FVANNUITY( ) 26973.46 を返す: `FVANNUITY(0.01, 24, 1000)`
1 期間あたりの支払金額	PMT( ) 1000 を返す:

求めたい値：	使用する Analytics 関数
	PMT(0.01, 24, 21243.39)
1 期間あたりの利率	RATE( ) 0.00999999(1%) を返す： RATE(24, 1000, 21243.39)
期間の数	NPER( ) 24.00 を返す： NPER(0.01, 1000, 21243.39)

年金の式

期末年金(期末払い) の**現在価値**を計算する式：

期末年金(期末払い) の**将来価値**を計算する式：

PPMT( ) 関数

単一の期間で貸付金に対して支払われた元金を返します。

構文

PPMT(*利率*, *指定期間*, *期間*, *金額* <,*種類*>)

パラメーター

名前	種類	説明
利率	数値	1 期あたりの利率。
指定期間	数値	元金の支払いを確認する期間。
期間	数値	支払期間の総数。
金額	数値	貸付金の元金。
種類 省略可能	数値	支払いのタイミング: ◦ 0 - 期末払い ◦ 1 - 期首払い 支払いのタイミングが省略された場合は、デフォルト値の 0 が使用されます。

> **メモ**
>
> *利率*、*期間*を指定する際には、**1 期あたりの**利率を確実に指定するため、整合性のある期間を使用する必要があります。
>
> 例:
>
> - 年利 5% の 2 年間の貸付金または投資に対して月払いする場合は、*利率*に 0.05/12、期間に 2 * 12 を指定します。
> - 同じ貸付金または投資に対して年払いする場合は、*利率*に 0.05、*期間*に 2 を指定します。

出力

数値。

例

基本的な例

$275,000 の貸付金を年利 6.5 パーセントで 25 年間にわたって返済する場合 (支払期日は月末です) の、1 年目に支払う元金、($) 367.24 が返されます。

```
PPMT(0.065/12, 1, 12*25, 275000, 0)
```

上記の貸付金の最終月において支払う元金、($) 1846.82 が返されます。

```
PPMT(0.065/12, 300, 12*25, 275000, 0)
```

備考

関連する関数

IPMT( ) 関数は PPMT( ) 関数に対して補完的役割を果たします。

CUMPRINC( ) 関数は一連の期間にわたって支払われた元金を計算します。

PROPER( ) 関数

各単語の最初の文字を大文字に、残りの文字を小文字に設定した文字列を返します。

構文

```
PROPER(文字列)
```

パラメーター

名前	種類	説明
文字列	文字	適切な大文字と小文字に変換する値。

出力

文字。

例

基本的な例

"John Doe" が返されます。

```
PROPER("JOHN DOE")
```

"John Doe" が返されます。

```
PROPER("john doe")
```

"1550 Alberni St." が返されます。

```
PROPER("1550 ALBERNI st.")
```

"Bill O'Hara" が返されます。

```
PROPER("BILL O'HARA")
```

Company_Name フィールドのすべての値を適切な大文字と小文字に変換した値が返されます。

```
PROPER(Company_Name)
```

備考

機能の仕組み

PROPER( ) 関数は、*文字列*の最初の文字と、空白の次に来る任意の文字を大文字に変換します。

ハイフン、アポストロフィ、アンパサンド(&) 、および他のいくつかの句読点と特殊文字に続く文字も大文字に変換されます。ほかのすべてのアルファベット文字は小文字に変換されます。

PROPER( ) の使用に適する場面

PROPER( ) の最も一般的な使い方は、すべて大文字かすべて小文字でデータソースに保存されている名前を適切な大文字と小文字書式に変換して、定型書簡やレポートに名前が正しく表示されるようにすることです。

PROPERTIES( ) 関数

指定されたAnalytics プロジェクト項目のプロパティ情報を返します。

構文

PROPERTIES(*名前*, *オブジェクトの種類*, *情報の種類*)

パラメーター

名前	種類	説明
名前	文字	情報を必要とするAnalytics プロジェクト項目の名前。*名前*は大文字と小文字を区別しません。 プロジェクト項目がAnalytics テーブルである場合は、データファイル名ではなくテーブルレイアウト名を指定します。たとえば、「january_invoices.fil」ではなく「Invoices」と指定します。 PROPERTIES( ) 関数を使用してアクティブなテーブルの名前を返させる場合は、「*名前* activetable」を指定します。
オブジェクトの種類	文字	*名前*で参照するAnalytics プロジェクト項目の種類。 **メモ** 現在のところ、サポートされているプロジェクト項目の種類は"table" のみです。
情報の種類	文字	Analytics プロジェクト項目について必要な情報の種類。 詳細については、"プロパティ情報の種類" ページ 681を参照してください。

出力

文字。出力文字列の最大長は260 文字です。プロパティ情報が見つからない場合は、空の文字列が返されます。

例

基本的な例

Analytics データ ファイル(.fil) に関する情報

"Ap_Trans.fil" が返されます。

```
PROPERTIES("Ap_Trans", "table", "filename")
```

"C:\ACL DATA\Sample Data Files" が返されます。

```
PROPERTIES("Ap_Trans", "table", "filepath")
```

開かれている Analytics テーブルに関する情報

"Ap_Trans" が返されます。

```
PROPERTIES("activetable", "table", "open")
```

外部データ ソースに関する情報

"Trans_May.xls" が返されます。

```
PROPERTIES("Trans_May", "table", "sourcename")
```

"C:\Project Data\Monthly Invoices_Excel" が返されます。

```
PROPERTIES("Trans_May", "table", "sourcepath")
```

"EXCEL" が返されます。

```
PROPERTIES("Trans_May", "table", "sourcetype")
```

備考

ファイル情報

"file" で始まる種類の情報は、Analytics テーブルと関連付けられている Analytics データ ファイル(.fil) に関する情報を提供します。

ソース情報

"source" で始まる種類の情報は、Analytics テーブルと関連付けることができる外部データソースに関する情報を提供します。PROPERTIES( ) 関数を使用して報告できる外部データソースは、Analytics テーブルの更新をサポートしている次の外部データソースだけです。

- Microsoft Excel
- Microsoft Access
- 区切り文字付きテキスト
- Adobe Acrobat(PDF)
- 印刷イメージ(レポート)
- SAP プライベート ファイル形式/DART
- XML
- XBRL
- ODBC データソース

プロパティ情報の種類

次の表は、PROPERTIES( ) 関数が返すことのできるプロパティ情報の種類の一覧です。Analytics テーブルが、PROPERTIES( ) 関数で現在使用できる唯一の Analytics プロジェクト項目です。

オブジェクトの種類	情報の種類	戻り値
"table"	"filename"	Analytics テーブルに関連付けられているデータ ファイルの名前。
	"filepath"	Analytics テーブルに関連付けられているデータ ファイルのパス。
	"filesize"	Analytics テーブルに関連付けられているデータ ファイルのサイズ(KB) 。
	"filemodifiedat"	Analytics テーブルに関連付けられているデータ ファイルが最後に変更された日付と時刻。
	"sourcename"	Analytics テーブルに関連付けられているデータ ソースの名前。 データ ソースは、Excel、Access、PDF、XML、または区切り文字付きテキスト ファイルなどの外部ファイルか、ODBC データ ソースである可能性があります。
	"sourcepath"	Analytics テーブルに関連付けられているデータ ソースのパス。 ODBC データ ソースではサポートされません。
	"sourcetype"	Analytics テーブルに関連付けられているデータ ソースの種類。
	"sourcesize"	Analytics テーブルに関連付けられているデータ ソースのサイズ(KB) 。 ODBC データ ソースではサポートされません。
	"sourcemodifiedat"	Analytics テーブルに関連付けられているデータ ソースが最後に変更された日付と時刻。 ODBC データ ソースではサポートされません。
	"open"	現在アクティブな Analytics テーブルの名前。 **メモ** 複数の Analytics テーブルを同時に開くことは可能ですが、ユーザー インターフェイスでは一度に 1 つのテーブルしかアクティブにすることができません。

PVANNUITY( ) 関数

一定の利率を使って計算した一連の将来価値の現在価値を返します。現在価値は現在の総額です。

構文

PVANNUITY(*利率*, *期間*, *支払金額* <,*種類*>)

パラメーター

名前	種類	説明
利率	数値	1 期あたりの利率。
期間	数値	支払期間の総数。
支払金額	数値	1 期間あたりの支払金額。 年金期間中、支払金額は毎回同じ金額でなければなりません。
種類 省略可能	数値	支払いのタイミング: ◦ 0 - 期末払い ◦ 1 - 期首払い 支払いのタイミングが省略された場合は、デフォルト値の 0 が使用されます。

> **メモ**
>
> *利率*、*期間*、*支払金額*を指定する際には、**1 期あたりの利率**を確実に指定するため、整合性のある期間を使用する必要があります。
>
> 例:
>
> - 年利 5% の 2 年間の貸付金または投資に対して月払いする場合は、*利率*に 0.05/12、*期間*に 2 * 12 を指定します。
> - 同じ貸付金または投資に対して年払いする場合は、*利率*に 0.05、*期間*に 2 を指定します。

出力

数値。結果は小数点以下 2 桁まで計算されます。

例

基本的な例

月次支払額

2 年間、月利 1% の複利で毎月月初に$1,000 を支払う場合の現在価値として、($) 21455.82が返されます。

```
PVANNUITY(0.01, 2*12, 1000, 1)
```

年次支払額

2 年間、年利 12% の複利で毎年年末に$12,000 を支払う場合の現在価値として、($) 20280.61 が返されます。

```
PVANNUITY(0.12, 2, 12000, 0)
```

高度な例

年金の計算

年金の計算では、次の 4 つの変数が使用されます。

- **現在価値または将来価値** - 下の例では $21,243.39 と$ 26,973.46
- **1 期間あたりの支払金額** - 下の例では $1,000.00
- **1 期間あたりの利率** - 下の例では月あたり 1%
- **期間の数** - 下の例では 24 か月

これらの変数のうち 3 つの値がわかっている場合は、Analytics 関数を使って残りの変数の値を計算できます。

求めたい値：	使用する Analytics 関数
現在価値	PVANNUITY( ) 21243.39 を返す： `FVANNUITY(0.01, 12, 1000)`
将来価値	FVANNUITY( ) 26973.46 を返す：

求めたい値：	使用する Analytics 関数
	FVANNUITY(0.01, 24, 1000)
1 期間あたりの支払金額	PMT( ) 1000 を返す： PMT(0.01, 24, 21243.39)
1 期間あたりの利率	RATE( ) 0.00999999(1%) を返す： RATE(24, 1000, 21243.39)
期間の数	NPER( ) 24.00 を返す： NPER(0.01, 1000, 21243.39)

年金の式

期末年金(期末払い) の**現在価値**を計算する式：

期末年金(期末払い) の**将来価値**を計算する式：

備考

関連する関数

FVANNUITY( ) 関数は PVANNUITY( ) 関数の逆関数です。

PVLUMPSUM( ) 関数

一定の利率を使って計算する特定の将来総額を生成するのに必要な現在価値を返します。現在価値は現在の総額です。

構文

PVLUMPSUM(*利率*, *期間*, *金額*)

パラメーター

名前	種類	説明
利率	数値	1 期あたりの利率。
期間	数値	期間の総数。
金額	数値	将来総額(最終期間の期末時の価値)

> **メモ**
>
> *利率*、*期間*を指定する際には、**1 期あたりの**利率を確実に指定するため、整合性のある期間を使用する必要があります。
>
> 例:
>
> - 年利 5% の 2 年間の貸付金または投資に対して月払いする場合は、*利率*に 0.05/12、期間に 2 * 12 を指定します。
> - 同じ貸付金または投資に対して年払いする場合は、*利率*に 0.05、*期間*に 2 を指定します。

出力

数値。結果は小数点以下 2 桁まで計算されます。

例

基本的な例

月ごとに複利が付く場合

月利 1% の複利で 2 年間運用した場合に将来総額 $1,269.73 を生み出せる初期投資元本として、($) 1000.00 が返されます。

```
PVLUMPSUM(0.01, 2*12, 1000)
```

月利 1% の複利で 2 年間運用した場合に将来総額 $1,000 を生み出せる初期投資元本として、($) 787.57 が返されます。

```
FVLUMPSUM(0.01, 2*12, 1000)
```

月利 1% の複利で 2 年間運用した場合に将来総額 $27,243.20 を生み出せる初期投資元本として、($) 21455.82 が返されます。

```
FVLUMPSUM(0.01, 2*12, 1000)
```

半年ごとに複利が付く場合

年利 12% の半年複利で 2 年間運用した場合に将来総額 $1,000 を生み出せる初期投資元本として、($) 792.09 が返されます。

```
FVLUMPSUM(0.12/2, 2*2, 1000)
```

1 年ごとに複利が付く場合

年利 12% の複利で 2 年間運用した場合に将来総額 $1,000 を生み出せる初期投資元本として、($) 797.19 が返されます。

```
FVLUMPSUM(0.12/2, 2*2, 1000)
```

備考

現在価値とは

運用する総額の現在価値とは、特定の時間枠内に特定の将来総額を生み出せる初期元本のことです。将来価値は、元本と複利の合計です。

関連する関数

FVLUMPSUM( ) 関数は PVLUMPSUM( ) 関数の逆関数です。

PYDATE( ) 関数

外部の Python スクリプトの関数によって計算された日付値を返します。Python によるデータ処理は Analytics の外部で行われます。

構文

PYDATE("*Py ファイル,Py 関数*" <, *フィールド|値* <,...*n*>>)

パラメーター

名前	種類	説明
Py ファイル、*Py 関数*	文字	実行する Python スクリプトの名前、カンマ、値を返す関数の名前。 たとえば、「myScript,myFunction」と指定します。 Python スクリプトを指定する場合、ファイル拡張子は省略してください。呼び出す関数から、同じ Python スクリプトやその他のスクリプトにある他の関数を呼び出すことができますが、実行するスクリプトはすべて、実行前に PYTHONPATH システム環境変数のフォルダーに入れておく必要があります。 詳細については、"Python の Analytics 連携用設定" ページ 899を参照してください。 **メモ** 使用する *Py関数*は、Python の datetime.date オブジェクトを返す必要があります。
フィールド|値 <,...n> 省略可能	文字 数値 日付時刻 論理	Python 関数の引数として使用するフィールド、式、リテラル値から成るこのリスト。値は呼び出す関数に指定順に渡されます。 Python スクリプトの関数定義を満たすのに必要な数の引数を指定できます。 **メモ** 文字入力から先頭と末尾の空白を除去するには、ALLTRIM (str) のように ALLTRIM() 関数を使用します。詳細については、"ALLTRIM( ) 関数" ページ 457を参照してください。

出力

日付時刻。

例

基本的な例

`20160630` が返されます。

```
PYDATE('hello,due_date', `20160531`, 30)
```

次の例は、日付と、日数としての猶予期間を入力として取って、請求書の支払期日を計算する外部 Python スクリプトです。請求日が**2016-05-31** で、猶予期間が30日の場合、支払期日は"2016-06-30"となります。

```
#! python
日付時刻インポート時間差分から

def due_date(inv_date, period):
   return(inv_date + timedelta(period))
```

高度な例

演算フィールドを定義する

Python スクリプトを使って支払期日を計算する演算フィールドを Ap_Trans テーブルに定義します。

```
OPEN AP_Trans
DEFINE FIELD due_date COMPUTED
WIDTH 27
   PYDATE( "hello,due_date" ,Invoice_Date, Pay_Period)
```

PYDATETIME( ) 関数

外部の Python スクリプトの関数によって計算された日付時刻値を返します。Python によるデータ処理は Analytics の外部で行われます。

構文

PYDATETIME("*Pyファイル,Py関数*" <, *フィールド|値* <,...*n*>>)

パラメーター

名前	種類	説明
Py ファイル、Py 関数	文字	実行する Python スクリプトの名前、カンマ、値を返す関数の名前。 たとえば、「myScript,myFunction」と指定します。 Python スクリプトを指定する場合、ファイル拡張子は省略してください。呼び出す関数から、同じ Python スクリプトやその他のスクリプトにある他の関数を呼び出すことができますが、実行するスクリプトはすべて、実行前に PYTHONPATH システム環境変数のフォルダーに入れておく必要があります。 詳細については、"Python の Analytics 連携用設定" ページ 899を参照してください。 **メモ** 使用する *Py関数*は、Python の datetime オブジェクトを返す必要があります。
フィールド|値 <,...n> 省略可能	文字 数値 日付時刻 論理	Python 関数の引数として使用するフィールド、式、リテラル値から成るこのリスト。値は呼び出す関数に指定順に渡されます。 Python スクリプトの関数定義を満たすのに必要な数の引数を指定できます。 **メモ** 文字入力から先頭と末尾の空白を除去するには、ALLTRIM (str) のように ALLTRIM() 関数を使用します。詳細については、"ALLTRIM( ) 関数" ページ 457を参照してください。

出力

日付時刻。

例

基本的な例

`20170101t0500` が返されます。

```
PYDATETIME("hello, combine_date_time", `20170101`, `t0500`)
```

次の例は、日付の引数と時間の引数を入力として取り、これらを合成したdatetimeオブジェクトを返す外部Pythonスクリプトです。

```
# hello.py content
日付時刻インポート日付時刻から

def combine_date_time(d,t):
   return datetime.combine(d,t)
```

高度な例

日付時刻に時刻を付加する

`20160101t2230` が返されます。

```
PYDATETIME("hello,add_time", `20160101 150000`, `t073000`)
```

次の例は、日付時刻と時間を入力として取り、日付時刻に時刻を付加する外部 Python スクリプトです。
データの例：2016-01-01 15:00:00 + 7 時間 30 分 00 秒 = 2016-01-01 22:30:00

```
# hello.py content
日付時刻インポート時間差分から
日付時刻インポート日付時刻から
日付時刻インポート時刻から
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( ) 関数

外部の Python スクリプトの関数によって計算された論理値を返します。Python によるデータ処理は Analytics の外部で行われます。

構文

PYDATE("*Py ファイル,Py 関数*" <, *フィールド|値* <,...*n*>>)

パラメーター

名前	種類	説明
Py ファイル、*Py 関数*	文字	実行する Python スクリプトの名前、カンマ、値を返す関数の名前。 たとえば、「myScript,myFunction」と指定します。 Python スクリプトを指定する場合、ファイル拡張子は省略してください。呼び出す関数から、同じ Python スクリプトやその他のスクリプトにある他の関数を呼び出すことができますが、実行するスクリプトはすべて、実行前に PYTHONPATH システム環境変数のフォルダーに入れておく必要があります。 詳細については、"Python の Analytics 連携用設定" ページ 899を参照してください。 **メモ** 使用する *Py関数*は、Python の真理値を返す必要があります。
フィールド|値 <,...n> 省略可能	文字 数値 日付時刻 論理	Python 関数の引数として使用するフィールド、式、リテラル値から成るこのリスト。値は呼び出す関数に指定順に渡されます。 Python スクリプトの関数定義を満たすのに必要な数の引数を指定できます。 **メモ** 文字入力から先頭と末尾の空白を除去するには、ALLTRIM(str) のように ALLTRIM() 関数を使用します。詳細については、"ALLTRIM( ) 関数" ページ 457を参照してください。

出力

論理。

例

基本的な例

F が返されます。

```
PYLOGICAL( "hello,str_compare", "basketball", "baseball", "b" )
```

次の例は、*char* として渡される文字カウントを使って、*str1* と *str2* を比較する外部 Python スクリプトです。

```
# hello.py content
def str_compare(str1, str2, char):
  return str1.count(char) > str2.count(char)
```

高度な例

フィールドを使用する

Vendor_Name と Vendor_City を比較したときの真理値が返されます。

```
PYLOGICAL( "hello,str_compare" Vendor_Name, Vendor_City, 'b' )
```

次の例は、*char* として渡される文字カウントを使って、*str1* と *str2* を比較する外部 Python スクリプトです。

```
# hello.py content
def str_compare(str1, str2, char):
  return str1.count(char) > str2.count(char)
```

PYNUMERIC( ) 関数

外部の Python スクリプトの関数によって計算された数値を返します。Python によるデータ処理は Analytics の外部で行われます。

構文

PYNUMERIC(*Py ファイル,Py 関数*, *小数点以下の桁数* <, *フィールド|値* <,...n>>)

パラメーター

名前	種類	説明
Py ファイル、*Py 関数*	文字	実行する Python スクリプトの名前、カンマ、値を返す関数の名前。 たとえば、「myScript,myFunction」と指定します。 Python スクリプトを指定する場合、ファイル拡張子は省略してください。呼び出す関数から、同じ Python スクリプトやその他のスクリプトにある他の関数を呼び出すことができますが、実行するスクリプトはすべて、実行前に PYTHONPATH システム環境変数のフォルダーに入れておく必要があります。 詳細については、"Python の Analytics 連携用設定" ページ 899を参照してください。 **メモ** 使用する *Py関数*は、Python の数値型を返す必要があります。
小数点以下の桁数	数値	戻り値に含める小数点以下の桁数。正の整数である必要があります。
フィールド|値 <,...n> 省略可能	文字 数値 日付時刻 論理	Python 関数の引数として使用するフィールド、式、リテラル値から成るこのリスト。値は呼び出す関数に指定順に渡されます。 Python スクリプトの関数定義を満たすのに必要な数の引数を指定できます。 **メモ** 文字入力から先頭と末尾の空白を除去するには、ALLTRIM (str) のように ALLTRIM() 関数を使用します。詳細については、"ALLTRIM( ) 関数" ページ 457を参照してください。

出力

数値。

例

基本的な例

35.00 が返されます。

```
PYNUMERIC("hello,get_nth_percent", 2, 80, 120, 30, 45, 30, 100, 35, 45)
```

次の例は、複数の値を含む、動的にサイズ設定したリストにおける、要求したパーセンタイルの値を返す外部 Python スクリプトです。

```
# hello.py content
演算インポート上限から
def get_nth_percent(percentage, *values):
    input_length = len(values)
    position = ceil((percentage/100.00) * input_length)
    return values[position-1]
```

PYSTRING( ) 関数

外部の Python スクリプトの関数によって計算された文字値を返します。Python によるデータ処理は Analytics の外部で行われます。

構文

PYSTRING("*Py ファイル,Py 関数*", *長さ*<,*フィールド|値* <,...*n*>>)

名前	種類	説明
Py ファイル、*Py 関数*	文字	実行する Python スクリプトの名前、カンマ、値を返す関数の名前。 たとえば、「myScript,myFunction」と指定します。 Python スクリプトを指定する場合、ファイル拡張子は省略してください。呼び出す関数から、同じ Python スクリプトやその他のスクリプトにある他の関数を呼び出すことができますが、実行するスクリプトはすべて、実行前に PYTHONPATH システム環境変数のフォルダーに入れておく必要があります。 詳細については、"Python の Analytics 連携用設定" ページ 899を参照してください。 **メモ** 使用する *Py関数*は、Python の string オブジェクトを返す必要があります。
長さ	数値	返される文字列に割り当てられる長さ。
フィールド|値 <,...n> 省略可能	文字 数値 日付時刻 論理	Python 関数の引数として使用するフィールド、式、リテラル値から成るこのリスト。値は呼び出す関数に指定順に渡されます。 Python スクリプトの関数定義を満たすのに必要な数の引数を指定できます。 **メモ** 文字入力から先頭と末尾の空白を除去するには、ALLTRIM (str) のように ALLTRIM() 関数を使用します。詳細については、"ALLTRIM( ) 関数" ページ 457を参照してください。

出力

文字。

例

基本的な例

"my test" が返されます。

```
PYSTRING('hello,main', 20, "my")
```

次の例は、文字列を入力として取り、その文字列に"test" を連結する外部 Python スクリプトです。

```
#! python
# hello.py content
def main(str):
   str2 = str + ' test'
   return(str2)
```

高度な例

部分文字列が返される

次の例では、Vendor_Name フィールドから最後の 2 文字が削除されて、削除後の部分文字列が返されます。

```
PYSTRING( "hello,sub_set", LENGTH(Vendor_Name), ALLTRIM(Vendor_Name), LENGTH
(ALLTRIM(Vendor_Name)), 0, LENGTH(ALLTRIM(Vendor_Name)) - 2
```

次の例は、文字列(str) 、文字列長(length) 、2 つの桁位置(p1 とp2) を入力として取る外部 Python スクリプトです。この関数により、p1(位置 1) とp2(位置 2) の間にある部分文字列が返されます。

```
#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( ) 関数

外部の Python スクリプトの関数によって計算された時刻値を返します。Python によるデータ処理は Analytics の外部で行われます。

構文

PYTIME("*Py ファイル,Py 関数*" <, *フィールド|値* <,...*n*>>)

パラメーター

名前	種類	説明
Py ファイル、Py 関数	文字	実行する Python スクリプトの名前、カンマ、値を返す関数の名前。 たとえば、「myScript,myFunction」と指定します。 Python スクリプトを指定する場合、ファイル拡張子は省略してください。呼び出す関数から、同じ Python スクリプトやその他のスクリプトにある他の関数を呼び出すことができますが、実行するスクリプトはすべて、実行前に PYTHONPATH システム環境変数のフォルダーに入れておく必要があります。 詳細については、"Python の Analytics 連携用設定" ページ 899を参照してください。 **メモ** 使用する *Py関数*は、Python の datetime.time オブジェクトを返す必要があります。
フィールド|値 <,...n> 省略可能	文字 数値 日付時刻 論理	Python 関数の引数として使用するフィールド、式、リテラル値から成るこのリスト。値は呼び出す関数に指定順に渡されます。 Python スクリプトの関数定義を満たすのに必要な数の引数を指定できます。 **メモ** 文字入力から先頭と末尾の空白を除去するには、ALLTRIM (str) のように ALLTRIM() 関数を使用します。詳細については、"ALLTRIM( ) 関数" ページ 457を参照してください。

出力

日付時刻。

例

基本的な例

`t2122` が返されます。

```
ASSIGN v_time_part = PYTIME("hello,get_time", `20160101 212223`)
```

外部 Python スクリプト：

```
# hello.py content
日付時刻インポート時刻から
日付時刻インポート日付から

def get_time(timestamp):
  return timestamp.time();
```

RAND( ) 関数

指定した境界内にあるランダム数を返します。

構文

```
RAND(数値)
```

パラメーター

名前	種類	説明
数値	数値	乱数の数値境界。 小数点以下の桁数を持つ数値を指定した場合には、生成される乱数は、数値と同じ小数点以下の桁数を持つことになります。 ◦ **正の数値** -を入力した場合には、返される乱数はゼロ以上、指定した数値未満になります。 0 ～ 99 の数値が返されます。 `RAND(100)` ◦ **負の数値** -を入力した場合には、返されるランダム数は指定した数値以上、ゼロ未満になります。 -1 ～ 100 の数値が返されます。 `RAND(−100)`

出力

数値。

例

基本的な例

278.61 が返されます。

```
RAND(1000.00)
```

3781 が返されます。

```
RAND(10000)
```

> **メモ**
> 戻り値は関数の実行ごとに異なります。

備考

RAND( ) では結果を複製できない

RAND( ) 関数を同じ*数値*に対して 2 度続けて使用した場合、異なる結果が返されます。RAND( ) 関数には、RANDOM コマンドと違ってシード値がありません。

重複する乱数は生成される可能性あり

RAND( ) を使用して、乱数をテーブルのすべてのレコードに割り当てる演算フィールドを作成する場合は、重複する乱数が生成されることがあります。乱数が一意である保証はありません。

テーブルのレコード数に対する*数値*の比率が大きいほど、生成される数が一意である確率が高くなります。

乱数は動的に更新される

簡易並べ替えを実行したり、フィルターを適用したり、列の配置を変えたり、ビューをスクロールしたりするような操作を実行するたびに、RAND( ) 関数の演算フィールドにより新しい乱数セットが生成されます。

乱数セットを固定するには、 **抽出**]ダイアログボックスで **ビュー**]または **フィールド**]オプションを使用して、新しいテーブルにそのデータを抽出します。

RATE( ) 関数

期間ごとの利率を返します。

構文

```
RATE(期間, 支払金額, 金額)
```

パラメーター

名前	種類	説明
期間	数値	支払期間の総数。
支払金額	数値	1 期間あたりの支払金額。
金額	数値	貸付金の元金。

> **メモ**
> RATE( ) 関数では支払は期末に行われることが前提となっています。

出力

数値。利率は小数点以下 8 桁で計算されます。

例

基本的な例

$275,000 の 25 年ローンに対して毎月 $1856.82 を支払う場合の月利率として、0.00541667(0.54%) が返されます。

```
RATE(12*25, 1856.82, 275000)
```

同じローンの年利率として 0.06500004(6.5%) が返されます。

```
RATE(12*25, 1856.82, 275000)*12
```

高度な例

名目利率から実効利率を求める

RATE( ) 関数では名目利率が計算されます。RATE( ) の結果から実効利率を求めるには、EFFECTIVE( ) 関数を使用します。

前記例のローンの実効利率として 0.06715155(6.7%) が返されます。

```
EFFECTIVE((RATE(12*25, 1856.82, 275000)*12), 12*25)
```

年金の計算

年金の計算では、次の 4 つの変数が使用されます。

- **現在価値または将来価値** - 下の例では $21,243.39 と $ 26,973.46
- **1 期間あたりの支払金額** - 下の例では $1,000.00
- **1 期間あたりの利率** - 下の例では月あたり 1%
- **期間の数** - 下の例では 24 か月

これらの変数のうち 3 つの値がわかっている場合は、Analytics 関数を使って残りの変数の値を計算できます。

求めたい値：	使用する Analytics 関数
現在価値	PVANNUITY( ) 21243.39 を返す： `FVANNUITY(0.01, 12, 1000)`
将来価値	FVANNUITY( ) 26973.46 を返す： `FVANNUITY(0.01, 24, 1000)`
1 期間あたりの支払金額	PMT( ) 1000 を返す： `PMT(0.01, 24, 21243.39)`
1 期間あたりの利率	RATE( ) 0.00999999(1%) を返す： `RATE(24, 1000, 21243.39)`

求めたい値：	使用する Analytics 関数
期間の数	NPER( ) 24.00 を返す： NPER(0.01, 1000, 21243.39)

年金の式

期末年金(期末払い)の**現在価値**を計算する式：

期末年金(期末払い)の**将来価値**を計算する式：

RDATE( ) 関数

R の関数またはスクリプトによって計算された日付値を返します。R によるデータ処理は Analytics の外部で行われます。

構文

RDATE(*rScript|rCode* <,*フィールド|値* <,...*n*>>)

パラメーター

名前	種類	説明
rScript | *rCode*	文字	実行する R コードのスニペットまたは R スクリプトの絶対または相対パス。 外部ファイルを使用せずに直接 R コードを入力する場合、コード内では、前後の引用符文字はエスケープしても使用することはできません。 ○ **有効** - 'var <- "\"test\"" ' ○ **無効** - 'var <- "\'test\'" '
フィールド|値 <,...*n*> 省略可能	文字 数値 日付時刻 論理	R スクリプトまたは R コード スニペットの引数として使用するフィールド、式、リテラル値から成るリスト。 値は呼び出す関数に指定順に渡されます。また、値を参照するには、R コード内で「value1, value2 ... value*N*」を使用します。 R コードの関数定義を満たすのに必要な数の引数を指定できます。 **メモ** 文字入力から先頭と末尾の空白を除去するには、ALLTRIM(str) のように ALLTRIM() 関数を使用します。詳細については、"ALLTRIM( ) 関数" ページ 457を参照してください。

出力

日付時刻。

例

基本的な例

`20160530` が返されます。

```
RDATE("as.Date(value1,'%m-%d-%Y')", "05-30-16")
```

高度な例

外部 R スクリプトを使用する

文字列から変換された日付が返されます。

```
RDATE("a<-source('c:\\scripts\\r_scripts\\sample.r');a[[1]]", dateText)
```

外部 R スクリプト (sample.r) :

```
dateForm <- function(dateText) {
  return(as.Date(dateText,format='%y%m%d'))
}
dateForm(value1)
```

備考

R からデータを返す

R スクリプトを呼び出す場合、source 関数を使用して、戻りオブジェクトを変数に割り当てます。次に、R 関数から返されて戻りオブジェクトに格納された値に、次のようにアクセスできます。

```
# 'a' はレスポンスオブジェクトを格納し、a[[1]] はデータ値にアクセスしています
"a<-source('c:\\scripts\\r_scripts\\sample.r');a[[1]]"
```

R のログファイル

R 言語のメッセージは、Analytics によりプロジェクト フォルダーの `aclrlang.log` ファイルに記録されます。このファイルを使って、R のエラーをデバッグします。

> **ヒント**
>
> ログファイルは Analytics Exchange のアナリティクス ジョブの結果フォルダーにあります。

AX Server での外部 R スクリプトの実行

AX Server で実行する分析アプリを作成していて、外部の R スクリプトを使用したい場合は、次の手順を実行します。

1. 分析アプリとともに、このファイルを関連ファイルとしてアップロードします。
2. このファイルを指定する際、FILE ANALYTIC タグを使用します。
3. ファイルの参照には、相対パス `./filename.r` を使用します。

> **メモ**
>
> 関連ファイルを使用することで、Analytics Exchange とともに R を実行する際に、TomEE アプリケーション サーバーのアカウントに、このファイルにアクセスするための十分な権限が与えられます。

RDATETIME( ) 関数

R の関数またはスクリプトによって計算された日付時刻値を返します。R によるデータ処理は Analytics の外部で行われます。

構文

RDATETIME(*rScript|rCode* <,*フィールド|値* <,...*n*>>)

パラメーター

名前	種類	説明
*rScript	rCode*	文字	実行する R コードのスニペットまたは R スクリプトの絶対または相対パス。 外部ファイルを使用せずに直接 R コードを入力する場合、コード内では、前後の引用符文字はエスケープしても使用することはできません。 ◦ **有効** - 'var <- "\"test\"" ' ◦ **無効** - 'var <- "\'test\'" '
フィールド|値 <,...n> 省略可能	文字 数値 日付時刻 論理	R スクリプトまたは R コード スニペットの引数として使用するフィールド、式、リテラル値から成るリスト。 値は呼び出す関数に指定順に渡されます。また、値を参照するには、R コード内で「value1, value2 ... value*N*」を使用します。 R コードの関数定義を満たすのに必要な数の引数を指定できます。 **メモ** 文字入力から先頭と末尾の空白を除去するには、ALLTRIM (str) のように ALLTRIM() 関数を使用します。詳細については、"ALLTRIM( ) 関数" ページ 457を参照してください。

出力

日付時刻。

例

基本的な例

現在の日付と時刻に45 分が加算されます。

```
RDATETIME("Sys.time() + value1",2700)
```

高度な例

外部 R スクリプトを使用する

次の例では、外部 R 関数にフィールドとリテラル値を渡すことで、45 分が日付時刻フィールドに加算されます。

```
RDATETIME("a<-'c:\\scripts\\sample.r');a[[1]]", start_date, 2700)
```

外部 R スクリプト (sample.r) :

```
add_time <- function(start, sec) {
  return(start + sec)
}
add_time(value1, value2)
```

備考

R からデータを返す

R スクリプトを呼び出す場合、source 関数を使用して、戻りオブジェクトを変数に割り当てます。次に、R 関数から返されて戻りオブジェクトに格納された値に、次のようにアクセスできます。

```
# 'a' はレスポンスオブジェクトを格納し、a[[1]] はデータ値にアクセスしています
"a<-source('c:\\scripts\\r_scripts\\sample.r');a[[1]]"
```

R のログファイル

R 言語のメッセージは、Analytics によりプロジェクトフォルダーの `aclrlang.log` ファイルに記録されます。このファイルを使って、R のエラーをデバッグします。

> **ヒント**
>
> ログファイルは Analytics Exchange のアナリティクスジョブの結果フォルダーにあります。

AX Server での外部 R スクリプトの実行

AX Server で実行する分析アプリを作成していて、外部の R スクリプトを使用したい場合は、次の手順を実行します。

1. 分析アプリとともに、このファイルを関連ファイルとしてアップロードします。
2. このファイルを指定する際、FILE ANALYTIC タグを使用します。
3. ファイルの参照には、相対パス `./filename.r` を使用します。

> **メモ**
>
> 関連ファイルを使用することで、Analytics Exchange とともに R を実行する際に、TomEE アプリケーション サーバーのアカウントに、このファイルにアクセスするための十分な権限が与えられます。

システム タイムゾーン

グリニッジ標準時 (GMT) は、Analytics で使用される R 環境のデフォルトの現在のタイムゾーンです。

RECLEN( ) 関数

現在のレコードの長さを返します。

構文

```
RECLEN( )
```

パラメーター

この関数にはパラメーターはありません。

出力

数値。

例

基本的な例

次の例では、長さが 110 であるすべてのレコードが抽出されます。

```
EXTRACT RECORD IF RECLEN( ) = 110 TO "Extract.fil"
```

備考

特定の長さのレコードを識別したり、予測より短いレコードを検査したりするには、RECLEN( ) 関数を使用します。この関数はレコード長を簡単に調べられる方法を提供するので、印刷イメージ(レポート) ファイルで作業している場合に有用です。

- 固定長のレコードでは、戻り値は一定(レコード長) です。
- 可変長のレコードでは、レコードによって戻り値は異なります。

RECNO( ) 関数

現在のレコード番号を返します。

構文

```
RECNO( )
```

パラメーター

この関数にはパラメーターはありません。

出力

数値。

例

基本的な例

次の例は、番号 10 から 20 のレコードを新しい Analytics テーブルに抽出します。

```
EXTRACT RECORD IF BETWEEN(RECNO( ),10,20) TO "Subset.fil"
```

備考

RECNO( ) 関数を使用すると、レコード番号をテーブルに出力したり、特定のレコードのテーブル内での相対的な位置を確認できます。

インデックス付きのテーブルとインデックス付きでないテーブル

この関数は現在の論理レコード番号を返します。

- テーブルにインデックスが付いていない場合、RECNO( ) 関数は値 1 から始まり、テーブル内のレコードごとに 1 ずつ増えていきます。この場合、論理レコード番号と物理レコード番号は同じです。
- テーブルにインデックスが付いている場合も RECNO( ) 関数による操作は変わりませんが、物理的ではなく論理的な順番でレコードをカウントします。

SEEK または FIND コマンドを実行する

SEEK または FIND コマンドを実行すると、レコード番号は 1 にリセットされます。

レコードを並べ替える

テーブル内のレコードを並べ替えても、RECNO( ) によって生成されたレコード番号の順序は変えられません。レコードに元々関連付けられていたレコード番号を保持するには、レコードを並べ替える前に、[**フィールド**]オプションを使用してこのデータを新しいテーブルに抽出してください。

RECOFFSET( ) 関数

現在のレコードを基準にして、指定されたn番目にあるレコードのフィールド値を返します。

構文

```
RECOFFSET(フィールド, レコードの数)
```

パラメーター

名前	種類	説明
フィールド	文字 数値 日付時刻	値を取得するフィールドの名前。
レコードの数	数値	現在のレコードから数えたレコード数。現在のレコードより後のレコードを指定する場合は正の数、現在のレコードより前のレコードを指定する場合は負の数にします。

出力

文字、数値、または日付時刻。戻り値は、入力の*フィールド* パラメーターと同じデータカテゴリに属します。

例

基本的な例

次のレコードの *Amount* 値が返されます。

```
RECOFFSET(Amount,1)
```

前のレコードの *Amount* 値が返されます。

```
RECOFFSET(Amount, -1)
```

高度な例

演算フィールド内で RECOFFSET を使用する

演算フィールド *Next_Amount* には、次のレコードが同じ顧客番号である場合に限り、次のレコードの Amount フィールドの値が表示されます。

この演算フィールドをスクリプトで定義するには、次の構文を使用します。

```
DEFINE FIELD Next_Amount COMPUTED
RECOFFSET(Amount,1) IF RECOFFSET(Customer,1) = Customer
0
```

次のレコードの顧客番号が現在のレコードの顧客番号と同じ場合に限り、*Next_Amount* は次のレコードの Amount フィールドの値になります。そうでない場合、*Next_Amount* には値ゼロが割り当てられます。

備考

RECOFFSET( ) 関数は現在のレコードを基準にして、指定された前後 n 番目にあるレコードのフィールド値を返します。

RECOFFSET( ) の使用に適する場面

この関数は高度な比較テストを行う場合によく使用されます。

この関数は、現在のレコードのフィールドと別のレコードのフィールドで値を比較するために使用できます。たとえば、現在のレコードの金額と前のレコードの金額の差額を計算する演算フィールドを追加する場合などに使用します。

テーブルの開始または終了位置

この関数では、フィールドにテーブルの開始または終了位置を指定した場合、数値フィールドに対してはゼロ、文字フィールドに対しては空白の文字列、日付フィールドに対しては 1900/01/01 が返されます。現在のレコードと比較するレコードがそれ以上存在しないため、このような場合には、関数は空白の出力を返します。

REGEXFIND( ) 関数

正規表現で指定されたパターンが文字列内に現れるかどうかを示す論理値を返します。

構文

```
REGEXFIND(文字列, パターン)
```

パラメーター

名前	種類	説明
文字列	文字	パターンとの一致をテストするフィールド、式、またはリテラル値。
パターン	文字	検索するパターン文字列(正規表現)。 パターンには、リテラル文字、メタ文字、またはこれら 2 つの組み合わせを指定できます。リテラル文字には、すべての英数字、一部の句読点、および空白が含まれます。 検索は大文字と小文字を区別します。つまり、英字の大文字と小文字は明確に指定しなければならないということです。

出力

論理。指定されたパターンの値が見つかった場合は **T**(true)、そうでない場合は **F**(false) を返します。

例

基本的な例

英字のパターン

次の例では、**Vendor_City** フィールドの値が "Phoenix"、"Austin"、または "Los Angeles" であるすべてのレコードに対し、T が返されます。それ以外のレコードに対しては 'F' が返されます。

```
REGEXFIND(Vendor_City, "Phoenix|Austin|Los Angeles")
```

次の例では、ラスト ネームが "John" または "Jon" で始まるすべてのレコードに対し、T が返されます。たとえば、Jon、Johnson、Johnston、Jonson、Jonston、Jones などが該当します。それ以外のレコードに対しては 'F' が返されます。

```
REGEXFIND(Last_Name,"^Joh?n")
```

次の例では、ラスト ネームが "John" または "Jon" であるすべてのレコードに対してのみ、T が返されます。それ以外のレコードに対しては 'F' が返されます。

```
REGEXFIND(Last_Name,"^Joh?n\b")
```

数値文字のパターン

次の例では、請求書番号に "98" を含んでいるすべてのレコードに対し、T が返されます。それ以外のレコードに対しては 'F' が返されます。

```
REGEXFIND(Invoice_Number, "98")
```

次の例では、請求書番号が "98" で始まるすべてのレコードに対し、'T' が返されます。それ以外のレコードに対しては 'F' が返されます。

```
REGEXFIND(Invoice_Number, "\b98")
```

次の例では、請求書番号が "98" で終わるすべてのレコードに対し、'T' が返されます。それ以外のレコードに対しては 'F' が返されます。

```
REGEXFIND(Invoice_Number, "98\b")
```

次の例では、請求書番号の左端から 5 桁目と 6 桁目に "98" を含んでいるすべてのレコードに対し、'T' が返されます。それ以外のレコードに対しては 'F' が返されます。

```
REGEXFIND(Invoice_Number, "\b\d\d\d\d98")
```

```
REGEXFIND(Invoice_Number, "\b\d{4}98")
```

文字が混在するパターン

次の例では、製品コードが 3 つの数字、1 つのハイフン、6 つの文字の順になっているすべてのレコードに対し、T が返されます。それ以外のレコードに対しては 'F' が返されます。

```
REGEXFIND(Product_Code, "\b\d{3}-[a-zA-Z]{6}\b")
```

次の例では、製品コードが3つ以上の数字、1つのハイフン、6つ以上の文字の順になっているすべてのレコードに対し、Tが返されます。それ以外のレコードに対しては'F'が返されます。

```
REGEXFIND(Product_Code, "\b\d{3,}-[a-zA-Z]{6}")
```

次の例では、英数字の請求書番号の左端から5桁目と6桁目に"98"を含んでいるすべてのレコードに対し、Tが返されます。それ以外のレコードに対しては'F'が返されます。

```
REGEXFIND(Invoice_Number, "\b\w{4}98")
```

次の例では、請求書番号に次の文字をすべて含んでいるすべてのレコードに対し、Tが返されます。それ以外のレコードに対してはFが返されます。

- 左端から4桁目までに任意の文字
- 左端から5桁目と6桁目に"98"

```
REGEXFIND(Invoice_Number, "\b.{4}98")
```

次の例では、請求書番号に"98"を含んでおり、その前に1 ～ 4文字があるすべてのレコードに対し、Tが返されます。それ以外のレコードに対しては'F'が返されます。

```
REGEXFIND(Invoice_Number, "\b.{1,4}98")
```

次の例では、請求書番号に次の文字をすべて含んでいるすべてのレコードに対し、Tが返されます。それ以外のレコードに対してはFが返されます。

- 左端から3桁目までに任意の文字
- 左端から4桁目に"5"または"6"
- 左端から5桁目と6桁目に"98"

```
REGEXFIND(Invoice_Number, "\b.{3}[56]98")
```

次の例では、請求書番号の次の文字をすべて含んでいるすべてのレコードに対し、Tが返されます。それ以外のレコードに対してはFが返されます。

- 左端から2桁目までに任意の文字
- 左端から3桁目と4桁目に"55"または"56"
- 左端から5桁目と6桁目に"98"

```
REGEXFIND(Invoice_Number, "\b.{2}(55|56)98")
```

備考

機能の仕組み

REGEXFIND( ) 関数は正規表現を使ってAnalyticsでデータを検索します。

正規表現は、さまざまな検索操作を実行する特殊文字であるメタ文字と、リテラル文字を組み合わせた、強力で柔軟性のある検索文字列です。

例：

```
REGEXFIND(Last_Name,"Sm(i|y)the{0,1}")
```

これはグループ( )、交替 |、量指定子 {} のメタ文字を使用して、**Last_Name** フィールド内から "Smith"、"Smyth"、"Smithe"、または "Smythe" を見つけるための正規表現を作成しています。

照合は順次実行される

*文字列*の値と*パターン*の値との照合は、順次実行されます。上記の例：

- "S" が **Last_Name** フィールドの左端から1桁目と照合される
- "m" が2桁目と照合される
- "i" と "y" が3桁目と照合される
- "t" が4桁目と照合される
- "h" が5桁目と照合される
- "e" が6桁目と照合される(照合対象に6桁目が存在する場合)

REGEXFIND( ) の使用に適する場面

REGEXFIND( ) は、シンプルまたは複雑なパターン一致を使ってデータを検索する場合に使用できます。

正規表現の構文を見るのは初めてである場合は特に、正規表現の作成は難しいかもしれません。より単純な FIND( )、MATCH( )、または MAP( ) などのAnalytics検索関数を使用して、検索の目的を達成できる可能性もあります。

検索要件がこれらの単純な関数の能力を超えている場合は、正規表現を使用することで、検索文字列の作成にほぼ無制限の柔軟性を与えられます。

REGEXFIND( ) によるスペースの扱い

スペース(空白) は、*文字列*と*パターン*のどちらに指定した場合でも文字として扱われるので、スペースを扱う際には注意が必要です。

*パターン*に空白を空白そのものとして指定するには、スペースを入力するか、またはメタ文字の \s を使用します。メタ文字を使用すると、特に複雑なパターンを指定する場合に、パターン内のスペースが読みやすくなるため、見落とされにくくなります。

フィールドの連結

同時に複数のフィールドにわたって検索したい場合は、2つ以上のフィールドを連結して*文字列*に指定することができます。

例：

```
REGEXFIND(Vendor_Name+Vendor_Street,"Hardware.*Main")
```

Vendor_Nameと**Vendor_Street**の両方のフィールドで、"Hardware"と"Main"という単語間に0個以上の文字が入っているパターンを検索します。

名前に"Hardware"という単語を含み、"Main"という通り沿いに立地する企業は、この正規表現に一致します。"Hardware on Main"という企業(名)は一致します。

連結したフィールドは1つのフィールドのように扱われます。ただし、ALLTRIM()関数を使用して個々のフィールドからスペースを除去しなければ、各フィールドの先頭と末尾のスペースを含むフィールドとなります。

連結するフィールド間の順序は重要

REGEXFIND()は、パターン内の文字を指定された順序で検索するため、フィールドを連結する順序は検索に影響を与えます。上の式で**Vendor_Name**と**Vendor_Street**を逆にした場合、得られる結果が少なくなる可能性があります。

正規表現のメタ文字

次の表は、REGEXFIND()およびREGEXREPLACE()で使用できるメタ文字の一覧と、それぞれのメタ文字で実行される操作を示しています。

このほかの正規表現構文もあって、Analyticsでサポートされていますが、もっと複雑です。それらの構文の完全な説明はこのガイドには記載されていません。正規表現について説明している多数のリソースがインターネットで入手できます。

Analyticsは、**ECMAScript**の正規表現の実装を使用しています。ほとんどの正規表現の実装は、共通のコア構文を使用しています。

> **メモ**
>
> Analyticsにおける現在の正規表現の実装は、英語以外の言語の検索を完全にサポートしていません。

メタ文字	説明
.	任意の1文字に一致します(改行文字を除く)。
?	直前のリテラル、メタ文字、または要素と0回または1回一致します。

メタ文字	説明
*	直前のリテラル、メタ文字、または要素と0回以上一致します。
+	直前のリテラル、メタ文字、または要素と1回以上一致します。
{}	直前のリテラル、メタ文字、または要素と指定した回数一致します。正確な数、範囲、または終わりが決められていない範囲を指定できます。 例： ◦ **a{3}** は "aaa" と一致します。 ◦ **X{0,2}L** は "L"、"XL"、および "XXL" と一致します。 ◦ **AB-\d{2,}-YZ** は、英数字の識別子で、接頭辞に "AB-"、接尾辞に "-YZ"、および本体部分に2つ以上の数字を持つすべての識別子と一致します。
[]	角かっこで囲まれた文字の中のいずれかに一致します。 例： ◦ **[aeiou]** はa、e、i、o、またはuと一致します。 ◦ **[^aeiou]** はa、e、i、o、またはu以外の文字と一致します。 ◦ **[A-G]** は大文字の英字AからGの範囲にある任意の文字と一致します。 ◦ **[A-Ga-g]** は大文字の英字AからG、または小文字の英字aからgの範囲にある任意の文字と一致します。 ◦ **[5-9]** は5から9の範囲にある任意の数字と一致します。
()	文字の並びまたはブロックを定義するグループを作成します。その後、グループは単一ユニットとして扱うことができます。 例： ◦ **S(ch)?mid?th?** これは、"Smith" または "Schmidt" と一致します。 ◦ **(56A.*){2}** は、"56A" の並びが少なくとも2回出現するすべての英数字の識別子と一致します。 ◦ **(56A).*-.*\1** は、"56A" の並びが少なくとも2回出現し、2つの出現の間にハイフンがあるすべての英数字の識別子と一致します。
\	エスケープ文字は、直後の文字がリテラルであることを示します。メタ文字と文字どおり一致させたい場合は、エスケープ文字を使用します。たとえば、**\(** は左かっこを見つけ、**** はバックスラッシュを見つけます。 次の文字のいずれかと完全に一致させたい場合は、エスケープ文字を使用します。 **^ $. * + ? = ! : | \ ( ) [ ] { }** アンパサンド(&) やアットマーク(@) などの他の句読点文字は、エスケープ文字を必要としません。
\ *int*	前にかっこ **( )** を使って定義したグループを繰り返し使用することを指定します。*int* は、前に定義したグループの、その他のグループからの位置(先頭から数える) を特定する整数です。このメタ文字は、REGEXFIND( ) とREGEXREPLACE( ) いずれのパターンパラメーターでも使用できます。 例： ◦ **(123).*\1** は、数字 "123" のグループが少なくとも2回出現するすべての識別子と一致します。 ◦ **^(\d{3}).*\1** は、先頭の3桁の数字が繰り返し現れるすべての識別子と一致します。

メタ文字	説明
	◦ **^(\d{3}).*\1.*\1** は、先頭の 3 桁の数字が繰り返し少なくとも 2 回現れるすべての識別子と一致します。 ◦ **^(\D)(\d)-.*\2\1** は、先頭の英字と数字が反転して再び現れるすべての識別子と一致します。
$ ***int***	ターゲット文字列で見つかったグループが置換文字列として使用されることを指定します。*int* は、対象文字列における、そのグループの、その他のグループからの位置 (先頭から数える) を特定する整数です。このメタ文字は、REGEXREPLACE ( ) の *新しい文字列* パラメーターで使用できます。 例 : ◦ さまざまな電話番号の表示形式と一致させるために **(\d{3})[-]?(\d{3})[-]?(\d{4})** というパターンを使用している場合、*新しい文字列* として **($1)-$2-$3** を使用すると、それ自体を数字と置き換えて、表示形式を標準化することができます。999 123-4567 と 9991234567 はどちらも (999)-123-4567 になります。
|	パイプ (|) の前後にある文字、文字のブロック、または式と一致します。 例 : ◦ **a|b** は a または b と一致します。 ◦ **abc|def** は "abc" または "def" と一致します。 ◦ **Sm(i|y)th** は "Smith" または "Smyth" と一致します。 ◦ **[a-c]|[Q-S]|[x-z]** は a、b、c、Q、R、S、x、y、z のいずれかの文字と一致します。 ◦ **\s|-** は空白文字またはハイフンと一致します。
\w	単語に使用される任意の文字と一致します (a から z、A から Z、0 から 9、およびアンダースコア _) 。
\W	単語に使用される文字以外の任意の文字と一致します (a から z、A から Z、0 から 9、またはアンダースコア _ 以外) 。
\d	任意の数字 (10 進数字) と一致します。
\D	10 進数字以外の任意の 1 文字と一致します。
\s	スペース (空白文字) と一致します。
\S	空白文字以外の任意の文字と一致します。
\b	単語の境界 (\w 文字と \W 文字の間) と一致します。 単語の境界自体に領域は取りません。例 : ◦ "United Equipment" には単語の境界が 4 つあります。スペースの左右に 1 つずつと、文字列のはじめと終わりに 1 つずつです。"United Equipment" は、正規表現 **\b\w*\b\W\b\w*\b** と一致します。

メタ文字	説明
	ヒント 単語の境界は、単語を区切るスペース以外の、カンマやピリオドなどの文字によっても区切ることができます。 たとえば、次の式は True と評価されます。 `REGEXFIND("jsmith@example.net", "\bexample\b")`
^	文字列の先頭と一致します。 角かっこ [ ] 内では、^ は指定内容の否定になります。
$	文字列の末尾と一致します。

関連する関数

一致パターンを検索、置換したい場合は、"REGEXREPLACE( ) 関数" 見開きページを使用します。

REGEXREPLACE( ) 関数

正規表現と一致する文字列のすべてのインスタンスを新しい文字列で置き換えます。

構文

```
REGEXREPLACE(文字列, パターン, 新しい文字列)
```

パラメーター

名前	種類	説明
文字列	文字	パターンとの一致をテストするフィールド、式、またはリテラル値。
パターン	文字	検索するパターン文字列(正規表現)。 パターンには、リテラル文字、メタ文字、またはこれら 2 つの組み合わせを指定できます。リテラル文字には、すべての英数字、一部の句読点、および空白が含まれます。 検索は大文字と小文字を区別します。つまり、英字の大文字と小文字は明確に指定しなければならないということです。
新しい文字列	文字	パターンと一致するすべての値と置き換える文字列。 置換文字列には、リテラル文字、元の文字列に基づく文字のグループ($*int* 要素を使用)、またはこれら 2 つの組み合わせを含めることができます。

出力

文字。

例

基本的な例

スペースを操作する

次の例では、文字列間の複数のスペースを単一スペースに置換することで、"AB CD EF" が返されます。

```
REGEXREPLACE("AB CD   EF", "\s+", " ")
```


文字フィールドのデータに対し、単語間の間隔を単一スペースに統一したデータが返されます。

```
REGEXREPLACE(文字フィールド, "\s+", " ")
```

文字フィールドのデータに対し、単語間の間隔を単一スペースに統一したデータが返されます。*新しい文字列*としてリテラルスペースの代わりにBLANKS( ) 関数を使用すると、スペースが視認しやすくなり、見落とされる可能性が低くなります。

```
REGEXREPLACE(character_field, "\s+", BLANKS(1))
```

電話番号を標準的な書式にする

"(123) 456-7890" が返されます。つまり、電話番号 '1234567890' が標準的な書式に置き換えられました。

```
REGEXREPLACE(SUBSTR("1234567890",1,14), "(\d{3})[\s-]*(\d{3})[\s-]*(\d{4})","($1) $2-$3")
```

Telephone_Number フィールド内の数字を標準的な書式にした値が返されます。

```
REGEXREPLACE(Telephone_Number, ".*(\d{3})[\s-\.\)]*(\d{3})[\s-\.]*(\d{4})", "($1) $2-$3")
```

テキストから "123-456-7890" が抽出されます。

```
REGEXREPLACE("Tel num: 123-456-7890 (office)", "(.*)(\d{3}[\s-\)\.]*\d{3}[\s-\.]*\d{4})(.*)", "$2")
```

Comment フィールドのテキスト内から電話番号を抽出し、標準的な書式に置き換えます。

```
REGEXREPLACE(Comment, "(.*)(\d{3})[\s-\)\.]*(\d{3})[\s-\.]*(\d{4})(.*)","($2) $3-$4")
```

一般的な書式を抽出する

文字列("1ABC-123aa") によって指定された値の一般的な書式を示す "9XXX-999xx" が返されます。

```
REGEXREPLACE(REGEXREPLACE(REGEXREPLACE("1ABC-123aa","\d","9"),"[a-z]","x"),"[A-Z]", "X")
```

Invoice_Number フィールド内のすべての識別子の一般的な書式が返されます。

```
REGEXREPLACE(REGEXREPLACE(REGEXREPLACE(Invoice_Number,"\d","9"),"[a-z]","x"),"[A-Z]", "X")
```

氏名を標準的な書式に置き換える

"John David Smith" が返されます。

```
REGEXREPLACE("Smith, John David", "^(\w+),(\s\w+)(\s\w+)?(\s\w+)?","$2$3$4 $1")
```

Full_Name フィールド内の氏名の構成要素が正式な順序である「*ファースト* (*ミドル*) (*ミドル*) *ラスト*」として返されます。

```
REGEXREPLACE(Full_Name, "^(\w+),(\s\w+)(\s\w+)?(\s\w+)?","$2$3$4 $1")
```

> **メモ**
>
> 氏名のデータには、アポストロフィが入った名前など、さまざまな要素の複雑な結合が存在していることがあります。氏名のデータにおける変化が多ければ、必要となる正規表現はこの例で提供されているものより通常、複雑になります。

備考

機能の仕組み

REGEXREPLACE( ) 関数では、正規表現を使って、データ内で一致するパターンを見つけ、その一致する値を新しい文字列で置き換えることができます。

例：

```
REGEXREPLACE(文字フィールド, "\s+", " ")
```

これは、テキスト文字の間にある 1 つ以上のスペースを単一のスペースに置き換えることにより、文字データにおける間隔を標準化しています。

REGEXREPLACE( ) 関数の検索部分は、REGEXFIND( ) 関数の検索部分と同じです。両方の関数に共通する検索機能の詳細については、"REGEXFIND( ) 関数" ページ 717を参照してください。

REGEXREPLACE( ) の使用に適する場面

REGEXREPLACE( ) は、Analytics でシンプルまたは複雑なパターン一致を使ってデータを検索、置換したい場合に使用できます。

文字列をそれ自体に置換する

$*int* 要素を使用すると、文字列をそれ自体に置換できます。これにより、意味のあるデータ部分を維持しながら、周囲のデータや混合されたデータを標準化または省略することが可能になります。

上には、電話番号と名前を使用するいくつかの例が示されています。

$*int* 要素を使用するには、まずパターン値をかっこ ( ) で囲むことでグループを作成する必要があります。詳細については、"REGEXFIND( ) 関数" ページ 717を参照してください。

文字の順次の照合を避ける

REGEXREPLACE( ) 関数を入れ子にすることで、文字の順次の照合を避け、他の部分文字列との位置関係にかかわらず、部分文字列を置き換えることができます。

次の2つの例における課題は、数字と文字がランダムな順序で出現している英数字のソースデータから一般的な書式を抽出することです。このように数字と文字の順序が不明な場合、どのようにしてパターン文字列を指定できるでしょうか?

解決策は、以下に例を示すように、まず内側のREGEXREPLACE( ) 関数を使って数字を検索、置換し、次に外側のREGEXREPLACE( ) 関数を使って文字を検索、置換することです。

次の例の場合には、"999XXX" が返されます。

```
REGEXREPLACE(REGEXREPLACE("123ABC","\d","9"),"[A-Z]","X")
```

次の例の場合には、"9X9X9X" が返されます。

```
REGEXREPLACE(REGEXREPLACE("1A2B3C","\d","9"),"[A-Z]","X")
```

置換文字列の長さと切り詰め

REGEXREPLACE( ) を使用して演算フィールドを作成した場合、演算フィールドの長さは元のフィールドの長さと同じになります。

置換文字列の長さがターゲット文字列の長さを超えており、文字列全体の長さが増えた場合、その増えた文字列の長さを演算フィールドの長さに収めることができないときには、文字列が切り詰められます。

まず、ターゲット文字列の末尾の文字から切り詰められ、次に置換文字列の末尾から切り詰められます。以下に切り詰めの例を示します。

文字列	*パターン*	*新しい文字列*	*フィールド長*	*結果*	*切り詰め後の文字*
x123x	123	A	5	xAx	なし
x123x	123	ABC	5	xABCx	なし
x123x	123	ABCD	5	xABCD	x
x123x	123	ABCDE	5	xABCD	x、E
x123x	123	ABCDE	6	xABCDE	x
x123x	123	ABCDE	7	xABCDEx	なし

切り詰めを避ける方法

切り詰められないようにするにはSUBSTR( ) 関数を使用してフィールド長を大きくしておきます。次の2番目の例を参照してください。

置換文字列 "E" と既存の文字 "x" を切り捨てた"xABCD" が返されます。

```
REGEXREPLACE("x123x","123","ABCDE")
```

すべての置換文字列と、置換されなかった既存の文字列で構成される、"xABCDEx" が返されます。

```
REGEXREPLACE(SUBSTR("x123x",1,10),"123","ABCDE")
```

正規表現のメタ文字

次の表は、REGEXFIND( ) およびREGEXREPLACE( ) で使用できるメタ文字の一覧と、それぞれのメタ文字で実行される操作を示しています。

このほかの正規表現構文もあって、Analyticsでサポートされていますが、もっと複雑です。それらの構文の完全な説明はこのガイドには記載されていません。正規表現について説明している多数のリソースがインターネットで入手できます。

Analyticsは、**ECMAScript** の正規表現の実装を使用しています。ほとんどの正規表現の実装は、共通のコア構文を使用しています。

> **メモ**
>
> Analyticsにおける現在の正規表現の実装は、英語以外の言語の検索を完全にサポートしていません。

メタ文字	説明
.	任意の1文字に一致します(改行文字を除く) 。
?	直前のリテラル、メタ文字、または要素と0回または1回一致します。
*	直前のリテラル、メタ文字、または要素と0回以上一致します。
+	直前のリテラル、メタ文字、または要素と1回以上一致します。
{}	直前のリテラル、メタ文字、または要素と指定した回数一致します。正確な数、範囲、または終わりが決められていない範囲を指定できます。 例: ◦ **a{3}** は "aaa" と一致します。 ◦ **X{0,2}L** は "L"、"XL"、および "XXL" と一致します。

メタ文字	説明
	◦ **AB-\d{2,}-YZ** は、英数字の識別子で、接頭辞に"AB-"、接尾辞に"-YZ"、および本体部分に2つ以上の数字を持つすべての識別子と一致します。
[]	角かっこで囲まれた文字の中のいずれかに一致します。 例： ◦ **[aeiou]** はa、e、i、o、またはuと一致します。 ◦ **[^aeiou]** はa、e、i、o、またはu以外の文字と一致します。 ◦ **[A-G]** は大文字の英字AからGの範囲にある任意の文字と一致します。 ◦ **[A-Ga-g]** は大文字の英字AからG、または小文字の英字aからgの範囲にある任意の文字と一致します。 ◦ **[5-9]** は5から9の範囲にある任意の数字と一致します。
()	文字の並びまたはブロックを定義するグループを作成します。その後、グループは単一ユニットとして扱うことができます。 例： ◦ **S(ch)?mid?th?** これは、"Smith" または "Schmidt" と一致します。 ◦ **(56A.*){2}** は、"56A" の並びが少なくとも2回出現するすべての英数字の識別子と一致します。 ◦ **(56A).*-.*\1** は、"56A" の並びが少なくとも2回出現し、2つの出現の間にハイフンがあるすべての英数字の識別子と一致します。
****	エスケープ文字は、直後の文字がリテラルであることを示します。メタ文字と文字どおり一致させたい場合は、エスケープ文字を使用します。たとえば、**\(** は左かっこを見つけ、**** はバックスラッシュを見つけます。 次の文字のいずれかと完全に一致させたい場合は、エスケープ文字を使用します。 **^ $. * + ? = ! : | \ ( ) [ ] { }** アンパサンド(&) やアットマーク(@) などの他の句読点文字は、エスケープ文字を必要としません。
**** *int*	前にかっこ **( )** を使って定義したグループを繰り返し使用することを指定します。*int* は、前に定義したグループの、その他のグループからの位置(先頭から数える) を特定する整数です。このメタ文字は、REGEXFIND( ) とREGEXREPLACE( ) いずれの*パターン* パラメーターでも使用できます。 例： ◦ **(123).*\1** は、数字 "123" のグループが少なくとも2回出現するすべての識別子と一致します。 ◦ **^(\d{3}).*\1** は、先頭の3桁の数字が繰り返し現れるすべての識別子と一致します。 ◦ **^(\d{3}).*\1.*\1** は、先頭の3桁の数字が繰り返し少なくとも2回現れるすべての識別子と一致します。 ◦ **^(\D)(\d)-.*\2\1** は、先頭の英字と数字が反転して再び現れるすべての識別子と一致します。
$ *int*	ターゲット文字列で見つかったグループが置換文字列として使用されることを指定します。*int* は、対象文字列における、そのグループの、その他のグループからの位置(先頭から数える) を特定する整数です。このメタ文字は、REGEXREPLACE( ) の*新しい文字列*パラメーターで使用できます。 例： ◦ さまざまな電話番号の表示形式と一致させるために **(\d{3})[-]?(\d{3})[-]?(\d{4})** というパターンを使用している場合、*新しい文字列*として **($1)-$2-$3** を使用すると、それ自体を数字と置き換えて、表示形式を標準化することができます。

メタ文字	説明
	999 123-4567 と9991234567 はどちらも (999)-123-4567 になります。
|	パイプ(|) の前後にある文字、文字のブロック、または式と一致します。 例： ◦ **a|b** は a または b と一致します。 ◦ **abc|def** は "abc" または "def" と一致します。 ◦ **Sm(i|y)th** は "Smith" または "Smyth" と一致します。 ◦ **[a-c]|[Q-S]|[x-z]** は a、b、c、Q、R、S、x、y、z のいずれかの文字と一致します。 ◦ **\s|-** は空白文字またはハイフンと一致します。
\w	単語に使用される任意の文字と一致します(a から z、A から Z、0 から 9、およびアンダースコア _) 。
\W	単語に使用される文字以外の任意の文字と一致します(a から z、A から Z、0 から 9、またはアンダースコア _ 以外) 。
\d	任意の数字(10 進数字) と一致します。
\D	10 進数字以外の任意の 1 文字と一致します。
\s	スペース(空白文字) と一致します。
\S	空白文字以外の任意の文字と一致します。
\b	単語の境界(\w 文字と\W 文字の間) と一致します。 単語の境界自体に領域は取りません。例： ◦ "United Equipment" には単語の境界が 4 つあります。スペースの左右に 1 つずつと、文字列のはじめと終わりに 1 つずつです。"United Equipment" は、正規表現 **\b\w*\b\W\b\w*\b** と一致します。 **ヒント** 単語の境界は、単語を区切るスペース以外の、カンマやピリオドなどの文字によっても区切ることができます。 たとえば、次の式は True と評価されます。 `REGEXFIND("jsmith@example.net", "\bexample\b")`
^	文字列の先頭と一致します。 角かっこ **[ ]** 内では、**^** は指定内容の否定になります。
$	文字列の末尾と一致します。

関連する関数

一致パターンを置換せずに検索したいだけの場合は、"REGEXFIND( ) 関数" ページ 717 を使用します。

REMOVE( ) 関数

指定した文字のみを含む文字列を返します。

構文

```
REMOVE(文字列, 有効な文字)
```

パラメーター

名前	種類	説明
文字列	文字	文字を取り除く文字列。
有効な文字	文字	*文字列*のうち、保持する文字。 *有効な文字*の中に二重引用符を指定する場合は、文字のリスト全体を一重引用符で囲む必要があります。 例：`'"-/'` **メモ** 指定する文字が*文字列*にない場合、戻り値に含まれません。

出力

文字。

例

基本的な例

"ABC123　" が返されます。

```
REMOVE("ABC 123 XX4","ABC123")
```

"ABC123XX　" が返されます。

```
REMOVE("zABC 123 XX4","ABCX123")
```

"1234 " が返されます。

```
REMOVE("ABC 123 XX4", "1234567890")
```

Product_Number フィールドのすべての値から数字以外のすべての文字を削除した値が返されます。

```
REMOVE(Product_Number,"0123456789")
```

備考

> **メモ**
>
> 現在では、REMOVE 関数の代わりにINCLUDE( ) 関数とEXCLUDE( ) 関数が使われるようになりました。
>
> しかし、旧バージョンとの互換性を保つために、現行バージョンの Analytics でも REMOVE( ) を使用できます。

機能の仕組み

REMOVE( ) 関数は文字データから不要な文字を取り除き、固定長文字列を返します。

REMOVE( ) の使用に適する場面

REMOVE( ) 関数は、住所フィールドなど、書式が統一されていないデータフィールドの書式を統一する場合に使用できます。また、編集が不十分なフィールドから句読点やその他の無効な文字を削除する場合にも使用できます。

さらに、SORT または JOIN コマンドの使用、重複の照合、あるいはレポートの出力を行う前にフィールド内のデータをクリーンにする場合にも、使用できます。

大文字と小文字の区別

REMOVE( ) 関数では大文字と小文字が区別されます。このため、*有効な文字*に "ID" を指定しても、これらの文字は "id#94022" には含まれていないことになります。大文字と小文字の両方が混在している可能性がある場合は、UPPER( ) 関数を使用して*文字列*を大文字に変換します。

例：

```
REMOVE(UPPER("id#94022"), "ID0123456789")
```

関連する関数

REMOVE( ) 関数は INCLUDE( ) 関数と似ていますが、次の点が異なります。

- REMOVE( ) では、削除された文字の代わりとして、空白が出力の末尾に追加されます。*文字列*の元の長さが保持されます。
- INCLUDE( ) では空白が追加されません。

REPEAT( ) 関数

指定された回数だけ部分文字列を繰り返す文字列を返します。

構文

```
REPEAT(文字列, 回数)
```

パラメーター

名前	種類	説明
文字列	文字	繰り返す文字列。
カウント	数値	*文字列*の値を繰り返す回数。

出力

文字。

例

基本的な例

"ABCABCABC" が返されます。

```
REPEAT("ABC",3)
```

"000000000" が返されます。

```
REPEAT("0",9)
```

備考

REPEAT( ) の使用に適する場面

REPEAT( ) 関数は、変数を定数値や空白で初期化したり、演算フィールドにデフォルト値を設定したりする場合に使用することができます。

REPLACE( ) 関数

指定された文字列のすべてのインスタンスを新しい文字列で置き換えます。

構文

```
REPLACE(文字列, 元のテキスト, 新しいテキスト)
```

パラメーター

名前	種類	説明
文字列	文字	文字が置き換えられる値。
元のテキスト	文字	置き換える文字。検索は大文字と小文字を区別します。
新しいテキスト	文字	*元のテキスト*の値を置き換えるテキスト。

出力

文字。

例

基本的な例

"a12345efg" が返されます。

```
REPLACE("abcdefg","bcd","12345")
```

"Rd." が返されます。

```
REPLACE("Road","Road","Rd.")
```

"ac" が返されます。

```
REPLACE("abc","b","")
```

高度な例

指定された文字を削除する

REPLACE( ) 関数では、指定した文字列を空文字列("") で置き換えることによって、指定文字列をソース文字列から削除することもできます。

"1234 Scott" が返されます。

```
REPLACE("1234 Scott rd.", "rd.", "")
```

フィールド長の調整

新しいテキスト("ABC") が*元のテキスト*("X") より長い場合は、置換後の文字列のフィールド長が置換結果を収めることができるように自動的に拡張されます。

フィールド長が3文字から5文字に拡張されることで、"9ABC9" が返されます。

```
REPLACE("9X9", "X", "ABC")
```

フィールド長は、2回目以降の置換では自動拡張されないため、すべての新しい文字を収めることができるほど十分に長くない場合には切り詰められます。

"9ABC9A" が返されます。

```
REPLACE("9X9", "X", "ABC")
```

切り詰められないようにするには、BLANKS( ) 関数またはリテラルスペースを使用して*文字列*の長さを大きくしておきます。

"9ABC9ABC" が返されます。

```
REPLACE("9X9X" + BLANKS(2), "X", "ABC")
```

```
REPLACE("9X9X" + "  ", "X", "ABC")
```

結果の文字列が*文字列*より短くなる場合には、フィールド長を同じにするため、結果の文字列にスペースが追加されます。

"9X9 " が返されます。

```
REPLACE("9X9", "X", "ABC")
```

備考

機能の仕組み

REPLACE( ) 関数は既存の文字列のすべてのインスタンスを新しい文字列で置き換えます。

"1234 Scott Road" が返されます。

```
REPLACE("1234 Scott rd.", "rd.", "")
```

REPLACE( ) の使用に適する場面

REPLACE( ) 関数は、住所フィールドなどの書式が統一されていないデータフィールドを正規化する場合や、編集が不十分なフィールドの無効な文字を置換する場合に使用します。重複の検査や、テーブルの結合または関連付けなどの操作を正確に実行するためには、正規化または標準化された書式のデータを必要とします。

大文字と小文字の区別

REPLACE ( ) 関数では大文字と小文字が区別されます。"RD." を *元のテキスト* に指定する際、*文字列* 内の値が小文字の場合は、一致が検出されないため、値は *新しいテキスト* の値に置き換えられません。

文字列 に大文字と小文字の両方が混在している可能性がある場合は、最初にUPPER( ) 関数を使用してすべての文字を大文字に変換します。

"1234 SCOTT ROAD" が返されます。

```
REPLACE(UPPER("1234 Scott rd."), "RD.", "ROAD")
```

不用意な置換が行われないようにする

REPLACE( ) 式を作成する際は、不用意な置換が行われないように、*文字列* 内に存在し得る *元のテキスト* のすべてのインスタンスを把握しておいてください。

"Richard" の最後の 2 文字が "rd" のため、"645 RichaRoad Road" が返されます。

```
REPLACE("645 Richard rd  ", "rd", "Road")
```

元のテキスト の値の前後にスペースを追加すると(" rd ")、名前に含まれる "rd" のインスタンスのように、前にスペースがないインスタンスは置換されないようになります。

"645 Richard Road" が返されます。

```
REPLACE("645 Richard rd  ", "rd", "Road")
```

REVERSE( ) 関数

文字の順番を逆にした文字列を返します。

構文

```
REVERSE(文字列)
```

パラメーター

名前	種類	説明
文字列	文字	順番を逆にする値。

出力

文字。

例

基本的な例

"E DCBA" が返されます。

```
REVERSE("ABCD E")
```

RJUSTIFY( ) 関数

文字列の末尾の空白をすべて先頭に移動させ、指定された文字列と同じ長さの右寄せした文字列を返します。

構文

```
RJUSTIFY(文字列)
```

パラメーター

名前	種類	説明
文字列	文字	右寄せする値。

出力

文字。

例

基本的な例

" ABC" が返されます。

```
RJUSTIFY("ABC  ")
```

備考

RJUSTIFY( ) の使用に適する場面

RJUSTIFY( ) 関数は文字フィールドを右寄せする場合に使用します。

RLOGICAL( ) 関数

R の関数またはスクリプトによって計算された論理値を返します。R によるデータ処理は Analytics の外部で行われます。

構文

RLOGICAL(*rScript|rCode* <,*フィールド|値* <,...*n*>>)

パラメーター

名前	種類	説明
*rScript	rCode*	文字	実行する R コードのスニペットまたは R スクリプトの絶対または相対パス。 外部ファイルを使用せずに直接 R コードを入力する場合、コード内では、前後の引用符文字はエスケープしても使用することはできません。 ◦ **有効** - 'var <- "\"test\"" ' ◦ **無効** - 'var <- "\'test\'" '
フィールド|値 <,...n> 省略可能	文字 数値 日付時刻 論理	R スクリプトまたは R コード スニペットの引数として使用するフィールド、式、リテラル値から成るリスト。 値は呼び出す関数に指定順に渡されます。また、値を参照するには、R コード内で「value1, value2 ... value*N*」を使用します。 R コードの関数定義を満たすのに必要な数の引数を指定できます。 **メモ** 文字入力から先頭と末尾の空白を除去するには、ALLTRIM (str) のように ALLTRIM() 関数を使用します。詳細については、"ALLTRIM( ) 関数" ページ 457を参照してください。

出力

論理。

例

基本的な例

T が返されます。

```
RLOGICAL("(value1>0.6) & (value2>0.7) & (value3>0.5)", 0.8, 0.9, 0.55)
```

高度な例

外部 R スクリプトを使用する

金額と上下限値を入力として取ります。一連の論理比較に基づいて、真理値が返されます。

```
RLOGICAL("a<-'c:\\scripts\\sample.r');a[[1]]", expense_amt, threshold_low, threshold_hi)
```

外部 R スクリプト (sample.r) :

```
test_truth <- function(amt, low, hi) {
   return(((amt > low) & (amt < hi)) | ((amt==low) | (amt==hi)))
}
test_truth(value1, value2, value3)
```

変数に格納された R コードの使用

次の例は、AND ロジックを使って 3 つのフィールドの論理テストを行っています。

```
v_rcode = "(value1>0.6) & (value2>0.7) & (value3>0.5)"
RLOGICAL(v_rcode, PACKED, MICRO_LONG, ACCPAC)
```

備考

R からデータを返す

R スクリプトを呼び出す場合、source 関数を使用して、戻りオブジェクトを変数に割り当てます。次に、R 関数から返されて戻りオブジェクトに格納された値に、次のようにアクセスできます。

```
# 'a' はレスポンスオブジェクトを格納し、a[[1]] はデータ値にアクセスしています
"a<-source('c:\\scripts\\r_scripts\\sample.r');a[[1]]"
```

R のログファイル

R 言語のメッセージは、Analytics によりプロジェクトフォルダーの `aclrlang.log` ファイルに記録されます。このファイルを使って、R のエラーをデバッグします。

> **ヒント**
>
> ログファイルは Analytics Exchange のアナリティクスジョブの結果フォルダーにあります。

AX Server での外部 R スクリプトの実行

AX Server で実行する分析アプリを作成していて、外部の R スクリプトを使用したい場合は、次の手順を実行します。

1. 分析アプリとともに、このファイルを関連ファイルとしてアップロードします。
2. このファイルを指定する際、FILE ANALYTIC タグを使用します。
3. ファイルの参照には、相対パス `./filename.r` を使用します。

> **メモ**
>
> 関連ファイルを使用することで、Analytics Exchange とともに R を実行する際に、TomEE アプリケーション サーバーのアカウントに、このファイルにアクセスするための十分な権限が与えられます。

RNUMERIC( ) 関数

R の関数またはスクリプトによって計算された数値を返します。R によるデータ処理は Analytics の外部で行われます。

構文

RNUMERIC(*rScript|rCode, 小数点以下の桁数 <,フィールド|値 <,...n>>*)

パラメーター

名前	種類	説明
*rScript	rCode*	文字	実行する R コードのスニペットまたは R スクリプトの絶対または相対パス。 外部ファイルを使用せずに直接 R コードを入力する場合、コード内では、前後の引用符文字はエスケープしても使用することはできません。 ◦ **有効** - 'var <- "\"test\"" ' ◦ **無効** - 'var <- "\'test\'" '
小数位	数値	戻り値に含める小数点以下の桁数。正の整数である必要があります。
フィールド|値 <,...n> 省略可能	文字 数値 日付時刻 論理	R スクリプトまたは R コード スニペットの引数として使用するフィールド、式、リテラル値から成るリスト。 値は呼び出す関数に指定順に渡されます。また、値を参照するには、R コード内で「value1, value2 ... value*N*」を使用します。 R コードの関数定義を満たすのに必要な数の引数を指定できます。 **メモ** 文字入力から先頭と末尾の空白を除去するには、ALLTRIM(str) のように ALLTRIM() 関数を使用します。詳細については、"ALLTRIM( ) 関数" ページ 457を参照してください。

出力

数値。

例

基本的な例

100 に対し、小数点以下の桁数を 10 にした 100.0000000000 が返されます。

```
RNUMERIC("print(value1)", 10, 100)
```

高度な例

変数への R コードの格納

100 に対し、小数点以下の桁数を 10 にした 100.0000000000 が返されます。

```
ASSIGN v_rcode = "print(value1)"
RNUMERIC(v_rcode, 10, 100)
```

外部ファイルに書き込む

簡単な加算を行い、R 内で sink 関数を使って、RNUMERIC( ) 関数に付加したコメントをファイルに書き込みます。

```
RNUMERIC("foo<-function(x,y){x+y};attr(foo, 'comment') <- 'foo は簡単な加算を行います';sink
('c:/temp/result.txt');attributes(foo);sink(NULL);foo(value1,value2)",0, amt, gross)
```

備考

R からデータを返す

R スクリプトを呼び出す場合、source 関数を使用して、戻りオブジェクトを変数に割り当てます。次に、R 関数から返されて戻りオブジェクトに格納された値に、次のようにアクセスできます。

```
# 'a' はレスポンスオブジェクトを格納し、a[[1]] はデータ値にアクセスしています
"a<-source('c:\\scripts\\r_scripts\\sample.r');a[[1]]"
```

R のログファイル

R 言語のメッセージは、Analytics によりプロジェクト フォルダーの `aclrlang.log` ファイルに記録されます。このファイルを使って、R のエラーをデバッグします。

> **ヒント**
>
> ログ ファイルは Analytics Exchange のアナリティクス ジョブの結 果 フォルダーにあります。

AX Server での外 部 R スクリプトの実 行

AX Server で実 行 する分 析 アプリを作 成 していて、外 部 の R スクリプトを使 用 したい場 合 は、次 の手 順 を実 行 します。

1. 分 析 アプリとともに、このファイルを関 連 ファイルとしてアップロードします。
2. このファイルを指 定 する際 、FILE ANALYTIC タグを使 用 します。
3. ファイルの参 照 には、相 対 パス `./filename.r` を使 用 します。

> **メモ**
>
> 関 連 ファイルを使 用 することで、Analytics Exchange とともに R を実 行 する際 に、TomEE アプリケーション サーバーのアカウントに、このファイルにアクセスするための十 分 な権 限 が与 えられます。

ROOT( ) 関数

数式の平方根を返します。

構文

```
ROOT(数値, 小数位)
```

パラメーター

名前	種類	説明
数値	数値	平方根を計算する数式。 *数値*が負の数の場合は、ゼロが返されます。
小数位	数値	出力で使用する小数点以下桁数。

出力

数値。

例

基本的な例

10.00 が返されます。

```
ROOT(100, 2)
```

31.6228 が返されます。

```
ROOT(1000, 4)
```

備考

機能の仕組み

ROOT( ) 関数は、数式やフィールド値の平方根を、指定された小数点以下の桁数で返します。この場合、返す値は適切に丸められます。

ROOT( ) の使用に適する場面

3 乗根などその他のルート関数を実行する場合は、LOG( ) を使用してください。

ROUND( ) 関数

数値を丸めた整数を返します。

構文

```
ROUND(数値)
```

パラメーター

名前	種類	説明
数値	数値	最も近い整数に丸める値。

出力

数値。

例

基本的な例

7 が返されます。

```
ROUND(7.2)
```

8 が返されます。

```
ROUND(7.5)
```

-8 が返されます。

```
ROUND(-7.5)
```

高度な例

通貨額を丸める

残高を最も近いドル金額に丸めたフィールドを作成するには、次のように指定します。

```
DEFINE FIELD Nearest_dollar_value COMPUTED ROUND(Balance)
```

備考

機能の仕組み

ROUND( ) は*数値*を最も近い整数に丸めた値を返します。

	正の値	負の値
次の整数に切り上げます	>= 0.5	< 0.5
次の整数に切り下げます	< 0.5	>= 0.5

小数点以下を特定の桁数で丸める

小数点以下を特定の桁数で丸める必要がある場合は、"DEC( ) 関数" ページ 526 関数を使用してください。ROUND( ) 関数は、0 の小数位を指定したDEC( ) 関数と同じです。

```
ROUND(数値)
```

上記のコマンドは、次のコマンドと同等です。

```
DEC(数値, 0)
```

RSTRING( ) 関数

R の関数またはスクリプトによって計算された文字列の値を返します。R によるデータ処理は Analytics の外部で行われます。

構文

RSTRING(*rScript|rCode*, *長さ*<,*フィールド|値* <,...*n*>>)

パラメーター

名前	種類	説明
*rScript	rCode*	文字	実行する R コードのスニペットまたは R スクリプトの絶対または相対パス。 外部ファイルを使用せずに直接 R コードを入力する場合、コード内では、前後の引用符文字はエスケープしても使用することはできません。 ◦ **有効** - 'var <- "\"test\"" ' ◦ **無効** - 'var <- "\'test\'" '
長さ	数値	返される文字列に割り当てられる長さ。
フィールド|値 <,...n> 省略可能	文字 数値 日付時刻 論理	R スクリプトまたは R コード スニペットの引数として使用するフィールド、式、リテラル値から成るリスト。 値は呼び出す関数に指定順に渡されます。また、値を参照するには、R コード内で「value1, value2 ... value*N*」を使用します。 R コードの関数定義を満たすのに必要な数の引数を指定できます。 **メモ** 文字入力から先頭と末尾の空白を除去するには、ALLTRIM (str) のように ALLTRIM() 関数を使用します。詳細については、"ALLTRIM( ) 関数" ページ 457を参照してください。

出力

文字。

例

基本的な例

"abc123" が返されます。

```
RSTRING("print(paste(value1,value2,sep=""))",6,"abc","123")
```

高度な例

外部 R スクリプトを使用する

x と y を連結し、スペース文字で区切って単一の文字列に格納します。

```
RSTRING("a<-source('./sample.r');a[[1]]",50, FirstName, LastName)
```

外部 R スクリプト (sample.r) :

```
conc <- function(x, y) {
   paste(x, y, sep=" ")
}
print(conc(value1, value2))
```

変数に格納された R コードの使用

x と y を連結し、スペース文字で区切って単一の文字列に格納します。

```
ASSIGN v_script = "conc <- function(x, y){paste(x, y, sep=' ')};conc(value1, value2)"
RSTRING(v_script, 50, FirstName, LastName)
```

テーブルの UUID を生成する R スクリプトを使用する

リザルトにアップロードする、例外から成るテーブルを作成しているとします。レコードごとに、一意であることが保証されている識別子が必要です。このようなフィールドを生成するには、R の **uuid** パッケージを使って、レコードごとに一意の主キー値を作成します。

```
EXTRACT RSTRING("uuid::UUIDgenerate()", 36) AS "id", first_name, last_name, birthdate TO
export_table
```

> **ヒント**
>
> uuid パッケージをインストールするには、R.exe を開いて次のコマンドを実行します。

```
install.packages("uuid")
```

備考

R からデータを返す

R スクリプトを呼び出す場合、source 関数を使用して、戻りオブジェクトを変数に割り当てます。次に、R 関数から返されて戻りオブジェクトに格納された値に、次のようにアクセスできます。

```
# 'a' はレスポンス オブジェクトを格納し、a[[1]] はデータ値にアクセスしています
"a<-source('c:\\scripts\\r_scripts\\sample.r');a[[1]]"
```

R のログ ファイル

R 言語のメッセージは、Analytics によりプロジェクト フォルダーの `aclrlang.log` ファイルに記録されます。このファイルを使って、R のエラーをデバッグします。

> **ヒント**
>
> ログ ファイルは Analytics Exchange のアナリティクス ジョブの結果フォルダーにあります。

AX Server での外部 R スクリプトの実行

AX Server で実行する分析アプリを作成していて、外部の R スクリプトを使用したい場合は、次の手順を実行します。

1. 分析アプリとともに、このファイルを関連ファイルとしてアップロードします。
2. このファイルを指定する際、FILE ANALYTIC タグを使用します。
3. ファイルの参照には、相対パス `./filename.r` を使用します。

> **メモ**
>
> 関連ファイルを使用することで、Analytics Exchange とともに R を実行する際に、TomEE アプリケーション サーバーのアカウントに、このファイルにアクセスするための十分な権限が与えられます。

RTIME( ) 関数

R の関数またはスクリプトによって計算された時刻値を返します。R によるデータ処理は Analytics の外部で行われます。

構文

RTIME(*rScript|rCode* <,*フィールド|値* <,...*n*>>)

パラメーター

名前	種類	説明
rScript | *rCode*	文字	実行する R コードのスニペットまたは R スクリプトの絶対または相対パス。 外部ファイルを使用せずに直接 R コードを入力する場合、コード内では、前後の引用符文字はエスケープしても使用することはできません。 ◦ **有効** - 'var <- "\"test\"" ' ◦ **無効** - 'var <- "\'test\'" '
フィールド|値 <,...*n*> 省略可能	文字 数値 日付時刻 論理	R スクリプトまたは R コード スニペットの引数として使用するフィールド、式、リテラル値から成るリスト。 値は呼び出す関数に指定順に渡されます。また、値を参照するには、R コード内で「value1, value2 ... value*N*」を使用します。 R コードの関数定義を満たすのに必要な数の引数を指定できます。 **メモ** 文字入力から先頭と末尾の空白を除去するには、ALLTRIM(str) のように ALLTRIM() 関数を使用します。詳細については、"ALLTRIM( ) 関数" ページ 457を参照してください。

出力

日付時刻。

例

基本的な例

`t0545` が返されます。

```
RTIME("value1+2700",`t0500`)
```

高度な例

外部 R スクリプトを使用する

次の例では、外部 R 関数にフィールドとリテラル値を渡すことで、45 分が時刻フィールドに加算されます。

```
RTIME("a<-source('c:\\scripts\\sample.r');a[[1]]", end_time, 2700)
```

外部 R スクリプト (sample.r) :

```
add_time <- function(start, sec) {
  return(start + sec)
}
add_time(value1, value2)
```

備考

R からデータを返す

R スクリプトを呼び出す場合、source 関数を使用して、戻りオブジェクトを変数に割り当てます。次に、R 関数から返されて戻りオブジェクトに格納された値に、次のようにアクセスできます。

```
# 'a' はレスポンスオブジェクトを格納し、a[[1]] はデータ値にアクセスしています
"a<-source('c:\\scripts\\r_scripts\\sample.r');a[[1]]"
```

R のログファイル

R 言語のメッセージは、Analytics によりプロジェクト フォルダーの `aclrlang.log` ファイルに記録されます。このファイルを使って、R のエラーをデバッグします。

> **ヒント**
>
> ログファイルは Analytics Exchange のアナリティクス ジョブの結果フォルダーにあります。

AX Server での外部 R スクリプトの実行

AX Server で実行する分析アプリを作成していて、外部の R スクリプトを使用したい場合は、次の手順を実行します。

1. 分析アプリとともに、このファイルを関連ファイルとしてアップロードします。
2. このファイルを指定する際、FILE ANALYTIC タグを使用します。
3. ファイルの参照には、相対パス `./filename.r` を使用します。

> **メモ**
>
> 関連ファイルを使用することで、Analytics Exchange とともに R を実行する際に、TomEE アプリケーション サーバーのアカウントに、このファイルにアクセスするための十分な権限が与えられます。

システム タイムゾーン

グリニッジ標準時 (GMT) は、Analytics で使用される R 環境のデフォルトの現在のタイムゾーンです。

SECOND( ) 関数

指定された時刻または日付時刻から秒数を抽出し、それを数値として返します。

構文

```
SECOND(時刻/日付時刻)
```

パラメーター

名前	種類	説明
時刻/日付時刻	日付時刻	秒数を抽出するフィールド、式、またはリテラル値。

出力

数値。

例

基本的な例

30 が返されます。

```
SECOND(`t235930`)
```

```
SECOND(`20141231 235930`)
```

Call_start_time フィールドの各値の秒数が返されます。

```
SECOND(Call_start_time)
```


備考

パラメーターの詳細

*時刻/日付時刻*に指定されたフィールドは、フィールド定義で正しく書式を定義してさえいれば、どのような時刻書式または日付時刻書式でも使用することができます。

リテラル時刻または日付時刻値の指定

*日付時刻*にリテラルの時刻値または日付時刻値を指定する場合は、次の表内の書式に制限されます。また、`20141231 235959` のように、値を逆引用符で囲む必要があります。

日付または時刻の個々の要素の間に、スラッシュ(/) やコロン(:) のような区切り文字をいっさい使用しないでください。

- **時刻値** - 以下の表に示す任意の時刻の書式を使用することができます。関数が正しく動作するためには、単独の時刻値の前に区切り文字を使用する必要があります。有効な区切り文字は文字 't' または 'T' です。24 時間形式で時刻を指定する必要があります。UTC(Coordinated Universal Time: 協定世界時) からのオフセットは、プラス記号(+) またはマイナス記号(-) で始める必要があります。
- **日付時刻値** - 以下の表に示す日付、区切り文字、および時刻の書式を任意に組み合わせて使用することができます。日付を時刻の前に置く必要があり、かつ、2 つの間に区切り文字を使用する必要があります。有効な区切り文字は空白スペース 1 つ、あるいは文字 't' または 'T' です。

形式の例	リテラル値の例
thhmmss	`t235959`
Thhmm	`T2359`
YYYYMMDD hhmmss	`20141231 235959`
YYMMDDthhmm	`141231t2359`
YYYYMMDDThh	`20141231T23`
YYYYMMDD hhmmss+/-hhmm (UTC オフセット)	`20141231 235959-0500`
YYMMDD hhmm+/-hh (UTC オフセット)	`141231 2359+01`
メモ UTC オフセットが設定されているデータのメインの時刻書式で hh を単独で使用しないでください。たとえば、「hh+hhmm」という使い方は避けてください。信頼できない結果になる可能性があります。	

SHIFT( ) 関数

入力値の最初の文字のビットを左または右に移動させた1文字の文字列を返します。

構文

```
SHIFT(文字, 移動させるビット数)
```

パラメーター

名前	種類	説明
文字	文字	ビットを移動する値。
移動させるビット数	数値	文字値を移動させるビット数を指定します。 ◦ **この値が正の場合** - 文字は左側に移動します。 ◦ **この値が負の場合** - 文字は右側に移動します。 ビット数が 15 より大きいか -15 より小さい場合、結果はバイナリのゼロ、CHR(0) になります。

出力

文字。

例

基本的な例

文字 "X" または CHR(88) が返されます(00010110 が 01011000 になります) 。

```
SHIFT(CHR(22), 2)
```

BS(後退) または CHR(8) が返されます(00010000 が 00001000 になります) 。

```
SHIFT(CHR(16), -1)
```

アクサン グラーブ文字または CHR(96) が返されます(10011011 が 01100000 になります) 。

```
SHIFT(CHR(155), 5)
```

備考

SHIFT( ) の使用に適する場面

SHIFT( ) 関数を BYTE( ) や CHR( )、MASK( ) 関数と組み合わせて使用すると、レコード内の個々のビットを分離させたり移動させることができます。

SIN( ) 関数

ラジアン単位で表された角度のサインを、小数点以下 15 桁の精度で返します。

構文

```
SIN(ラジアン)
```

パラメーター

名前	種類	説明
ラジアン	数値	ラジアン単位の角度。

出力

数値。

例

基本的な例

指定したラジアン数 (30 度に相当) に対し、その正弦である 0.500000000000000 が返されます。

```
SIN(0.523598775598299)
```

30 度に対し、その正弦値である 0.500000000000000 が返されます。

```
SIN(30 * PI( )/180)
```

高度な例

入力値として度を使用する

30 度の正弦を小数点第 3 位に丸めた 0.500 が返されます。

```
DEC(SIN(30 * PI( )/180),3)
```

備考

Mantissa Arc Test の実行

Analytics の 3 つの三角関数 SIN( )、COS( )、および TAN( ) は、ベンフォードの法則に関連する Mantissa Arc Test の実行をサポートします。

度のラジアンへの変換

入力値が度で表されている場合は、PI( ) 関数を使用して角度をラジアンに変換できます。(*度* * PI( )/180) = *ラジアン*となります。必要に応じて、DEC( ) 関数を使用して、戻り値を丸めたり切り捨てたりすることができます。

SOUNDEX( ) 関数

ほかの文字列との発音の比較に使用できる、指定された文字列のsoundexコードを返します。

構文

```
SOUNDEX(名前)
```

パラメーター

名前	種類	説明
名前	文字	評価する文字式。

出力

文字。4文字のsoundexコードを返します。

例

基本的な例

発音は同じだが、綴りが異なる単語

次の2つの例では、綴りが異なっていても発音が同じのため、同じsoundexコードが返されます。
F634 が返されます。

```
SOUNDEX("Fairdale")
```

F634 が返されます。

```
SOUNDEX("Faredale")
```


発音が似ている単語

次の2つの例では異なるsoundexコードが返されますが、これら2つの単語は発音が似ているため、近い値が返されます。

J525が返されます。

```
SOUNDEX("Jonson")
```

J523が返されます。

```
SOUNDEX("Jonston")
```

発音が異なる単語

次の2つの例では、2つの単語がまったく似ていないため、互いから非常に遠いsoundexコードが返されます。

S530が返されます。

```
SOUNDEX("Smith")
```

M235が返されます。

```
SOUNDEX("MacDonald")
```

フィールドへの入力値

Last_nameフィールドの各値のsoundexコードが返されます。

```
SOUNDEX(Last_Name)
```

高度な例

一致するsoundexコードを特定する

Last_Nameフィールドの各値のsoundexコードを表示する演算フィールド、**Soundex_Code**を作成します。

```
DEFINE FIELD Soundex_Code COMPUTED SOUNDEX(Last_Name)
```

ビューに演算フィールド**Soundex_Code**を追加し、その演算フィールドで重複検査を実行することによって、一致するあらゆるsoundexコードを特定することができます。


```
DUPLICATES ON Soundex_Code OTHER Last_Name PRESORT OPEN TO "Possible_Dupes.fil"
```

一致するsoundexコードは、**Last_name**フィールドの関連する文字値が重複している可能性があることを示しています。

備考

SOUNDEX( ) の使用に適する場面

SOUNDEX( ) 関数は発音の似た値を検索する場合に使用できます。発音の類似は、重複している可能性がある値や、手作業で入力されたデータで不統一な綴りを見つける1つの方法です。

機能の仕組み

SOUNDEX( ) は、評価される文字列のAmerican Soundexコードを返します。コードはすべて、1文字の後に3つの数字が続きます。たとえば、"F634" などです。

Soundexコードの意味

- コードの最初の文字は、評価される文字列中の最初の文字を表します。
- コード内の各番号は、6つのAmerican Soundexグループのうちの1つを表します。これらのグループは、発音が似た子音で構成されています。

  これらのグループに基づいて、soundex処理は、評価される文字列のうち、先頭文字の後に来る最初の3つの子音をエンコードします。

Soundex処理で無視される内容

Soundex処理では以下が無視されます。

- 大文字の使用
- 母音
- 子音 "H"、"W"、"Y"
- エンコードされた3つの子音の後に来るすべての子音

返されるコード内の末尾にある1つ以上のゼロ(0) は、評価される文字列には、最初の文字の後に子音が3つ未満しかないことを示しています。

Soundex処理にある制限

SOUNDSLIKE( ) 関数とSOUNDEX( ) 関数のどちらにも、ある制限があります。

- soundexアルゴリズムは、英語で発音された言葉を処理するように設計されており、他の言語で使用した場合には、有効性の度合いが違ってきます。
- soundex処理は発音の照合を実行しますが、一致する単語は同じ文字で始まっていなければなりませ

ん。これはつまり、発音が同じであるいくつかの単語は一致しないということです。

たとえば、"F" で始まる単語と"Ph" で始まる単語は発音が同じですが、これらは決して一致しません。

関連する関数

- **SOUNDSLIKE( )** - 文字列の発音を比較するもう1つの方法。
- **ISFUZZYDUP( ) および LEVDIST** - は、発音記号の比較(発音)ではなく、正字法の比較(綴り)に基づいて文字列を比較します。
- **DICECOEFFICIENT( )** - は、文字列を比較するときに、文字または文字ブロックの相対位置を重視しないか、または完全に無視します。

SOUNDSLIKE( ) 関数

文字列が比較文字列と発音学的に一致しているかどうかを示す論理値を返します。

構文

```
SOUNDSLIKE(名前, 発音の似ている名前)
```

パラメーター

名前	種類	説明
名前	文字	比較の最初の文字列。
発音の似ている名前	文字	比較の 2 番目の文字列。

出力

論理。比較する値同士の発音が一致している場合は **T**(true) 、そうでない場合は **F**(false) が返されます。

例

基本的な例

"Fairdale" と "Faredale" はどちらも soundex コードが F634 のため、T が返されます。

```
SOUNDSLIKE("Fairdale","Faredale")
```

"Jonson" の soundex コードは J525、"Jonston" の soundex コードは J523 のため、F が返されます。

```
SOUNDSLIKE("Jonson","Jonston")
```

Last_name フィールドの各値の soundex コードが文字列 "Smith" の soundex コードと一致するかどうかを示す論理値 (T または F) が返されます。

```
SOUNDSLIKE(Last_Name,"Smith")
```

高度な例

"Smith" と発音が似ている値を抽出する

Last_name フィールドの、"Smith" と発音が似ているすべての値を抽出するフィルターを作成するには、次のように指定します。

```
SET FILTER TO SOUNDSLIKE(Last_Name,"Smith")
```

備考

SOUNDSLIKE( ) の使用に適する場面

SOUNDSLIKE( ) 関数は発音の似た値を検索する場合に使用できます。発音の類似は、重複している可能性がある値や、手作業で入力されたデータで不統一な綴りを見つける 1 つの方法です。

機能の仕組み

SOUNDSLIKE( ) 関数はまず、比較対象となる文字列を 4 文字の American Soundex コードに変換します。このコードは、各文字列の最初の文字と、その後に続く最初の 3 つの子音に基づきます。

次に、各文字列のコードを比較し、それらが一致するかどうかを示す論理値を返します。

soundex コードの詳細については、"SOUNDEX( ) 関数" ページ 765を参照してください。

大文字と小文字の区別

この関数では大文字と小文字が区別されないため、"SMITH" は "smith" と同じであると判断されます。

Soundex 処理にある制限

SOUNDSLIKE( ) 関数と SOUNDEX( ) 関数のどちらにも、ある制限があります。

- soundex アルゴリズムは、英語で発音された言葉を処理するように設計されており、他の言語で使用した場合には、有効性の度合いが違ってきます。
- soundex 処理は発音の照合を実行しますが、一致する単語は同じ文字で始まっていなければなりません。これはつまり、発音が同じであるいくつかの単語は一致しないということです。

  たとえば、"F" で始まる単語と "Ph" で始まる単語は発音が同じですが、これらは決して一致しません。

関連する関数

- **SOUNDEX( )** - 文字列の発音を比較するもう 1 つの方法。
- **ISFUZZYDUP( ) および LEVDIST** - は、発音記号の比較 (発音) ではなく、正字法の比較 (綴り) に基づいて文字列を比較します。

- **DICECOEFFICIENT( )** - は、文字列を比較するときに、文字または文字ブロックの相対位置を重視しないか、または完全に無視します。

SPLIT( ) 関数

文字列のうちの指定された部分を返します。

構文

SPLIT(*文字列*, *区切り文字*, *セグメント* <,*テキスト修飾子*>)

パラメーター

名前	種類	説明
文字列	文字	セグメントを抽出する値。
区切り文字	文字	セグメントを区切る文字または文字列。 詳細については、"区切り文字の動作" ページ 774を参照してください。
セグメント	数値	抽出するセグメント。 抽出するセグメントを指定するには、番号を使用します。たとえば、3 番目のセグメントを抽出するには、「3」を指定します。
テキスト修飾子 省略可能	文字	テキストのセグメントの開始位置と終了位置を示す文字または文字列。 テキスト修飾子のペアの内部に出現する*区切り*文字は、区切り文字でなくテキストとして読み取られます。 *テキスト修飾子*は引用符で囲む必要があります。一重引用符のテキスト修飾子は二重引用符で、二重引用符テキスト修飾子は一重引用符で、それぞれ囲む必要があります。 **ヒント** この任意のパラメーターは、区切り文字とテキスト修飾子が含まれるソース データをインポートして操作するときに役立つ場合があります。

出力

文字。

例

基本的な例

カンマで区切られたセグメント

"seg1" が返されます。

```
SPLIT("seg1,seg2,seg3", ",", 1)
```

"seg3" が返されます。

```
SPLIT("seg1,seg2,seg3", ",", 3)
```

"" が返されます(3 番目のセグメントは空です) 。

```
SPLIT("seg1,seg2,,seg4", ",", 3)
```

文字列およびスペースの区切り文字

"seg3" が返されます。

```
SPLIT("seg1/*seg2/*seg3", "/*", 3)
```

"Doe" が返されます。

```
SPLIT("Jane Doe", " ", 2)
```

区切り文字をテキスト修飾子でエスケープする

区切り文字でなくテキストとして読み取られるカンマが含まれている "Doe, Jane" が返されます。

```
SPLIT('"Doe, Jane","Smith, John"', ",", 1, '"')
```

高度な例

クレジット カード番号から数字を抽出する

SPLIT( ) 関数を使用して、クレジット カード番号からダッシュを取り除くことができます。

変数を使用してクレジット カード番号の各セグメントを取得した後、それらのセグメントを連結して別の変数に格納します。

```
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
```

ccNum の値は "4150222233334444" です。

この例は SPLIT( ) 関数について説明していますが、EXCLUDE( ) 関数を使用すると、より効率的にダッシュを取り除くことができることに留意してください。

備考

機能の仕組み

SPLIT( ) 関数は、文字データをスペースやカンマなどの区切り文字に基づいてセグメントに区切り、指定されたセグメントを返します。

SPLIT( ) の使用に適する場面

SPLIT( ) 関数は、レコードまたはフィールドからデータの特定のセグメントを抽出する場合に使用できます。このセグメントは、個々のレコードまたはフィールドの同じ位置に存在している必要があります。

区切り文字の動作

区切り文字は、ソース文字列においてデータをセグメントとして区切ったり指定したりするものです。

多数のセグメントがある文字列では、セグメントの大部分は 2 つの区切り文字に挟まれています。これに対し、最初のセグメントの前と、最後のセグメントの後には区切り文字がありません。

例：

```
SPLIT("seg1,seg2,seg3", ",", 1)
```

ソース文字列が区切り文字で始まっている場合、その区切り文字の後のセグメントは 2 番目のセグメントとして処理されます。

"seg1" が返されます。

```
SPLIT("seg1,seg2,seg3", ",", 1)
```

大文字と小文字の区別

*区切り文字*または*テキスト修飾子*として、大文字や小文字がある文字を指定する場合、その文字と、データ内の区切り文字やテキスト修飾子の間で、大文字と小文字の違いも一致している必要があります。

関連する関数

SPLIT( ) とSUBSTR( ) はともに、長いソース文字列からデータのセグメントを返します。

- SPLIT( ) は区切り文字を使ってセグメントを識別します。
- SUBSTR( ) は数値文字の位置を使ってセグメントを識別します。

STOD( ) 関数

シリアル日付、つまり、整数で表される日付を日付値に変換します。"Serial to Date" の省略形です。

構文

```
STOD(シリアル日付 <,開始日>)
```

パラメーター

名前	種類	説明
シリアル日付	数値	変換するフィールド、式、またはリテラル値。 *シリアル日付*には、シリアル日付またはシリアル日付時刻を指定できます。シリアル日付時刻の日付部分のみが考慮されます。時刻部分は無視されます。
開始日 省略可能	日付時刻	シリアル日付の計算基準となる開始日。これを省略した場合は、デフォルトの開始日である 1900 年 1 月 1 日が使用されます。

出力

日付時刻。日付値は、現在 Analytics に設定されている日付の表示書式を使用して出力されます。

例

基本的な例

`20141231` に対して現在の Analytics 日付表示書式である DD MMM YYYY を適用した、31 Dec 2014 が返されます。

```
STOD(42003)
```

`20181231` に対して現在の Analytics 日付表示書式である DD MMM YYYY を適用した、31 Dec 2018 が返されます。

```
STOD(42003, `19040101`)
```

Invoice_Date フィールドの各シリアル日付値に対応する日付が返されます。

```
STOD(Invoice_Date)
```

高度な例

開始日を 1900-01-01 より前の日付に変更する

Analytics の最早の開始日である 1900 年 1 月 1 日より前の値に開始日を変更するには、日付の算術を行います。

1. デフォルトの開始日を使ってシリアル日付を変換します。
2. 実際の開始日から 1900-01-01 までの日数を差し引きます。

次の例は、開始日として 1899-01-01 を使用した場合に`20131231`と評価されます。

```
STOD(42003) - 365
```

備考

機能の仕組み

STOD( ) 関数を使用すると、シリアル日付を標準の日付に変換することができます。Analytics のシリアル日付は、1900 年 1 月 1 日からの経過日数を表します。

シリアル日付	相当する標準の日付
1	1900 年 1 月 2 日
365	1900 年 12 月 31 日
42003	2014 年 12 月 31 日
0	有効でない

シリアル日付の詳細については、シリアル日付時刻を参照してください。

Analytics のシリアル日付と Excel シリアルの日付との比較

Analytics のシリアル日付は Microsoft Excel のシリアル日付に似ています。注意すべき類似のキーポイントが 1 つと、相違のキーポイントが 1 つあります。これら 2 つのポイントには関連性がありません。

類似のポイント

Analytics と Excel はどちらも、1900 年を 366 日ある閏年として扱います。1900 年は実際には閏年ではありませんでしたが、Excel は Lotus 1-2-3 との互換性を保つために、この年を閏年として処理しました。

差異の点

Analytics のシリアル日付は、Excel のシリアル日付から 1 日ずれています。Excel では、1900 年 1 月 1 日はシリアル日付 '1' になります。Analytics では、1900 年 1 月 1 日はカウントされないので、1900 年 1 月 2 日がシリアル日付 '1' になります。

開始日

一部のソースデータファイルは 1900 年 1 月 1 日以外の開始日を使用する場合があります。*開始日*は、ソースデータファイルの開始日と一致させることができます。シリアル日付の計算基準となる開始日。

ソースデータファイルの開始日	次を指定します。	詳細
1900 年 1 月 1 日	STOD(*日付フィールド*)	1900 年 1 月 1 日はデフォルト開始日であるため、*開始日*を指定する必要はありません。
1901 年 1 月 1 日	STOD(*日付フィールド*, `19010101`)	*開始日* `19010101` を指定し、ソースデータファイルで使用される 1901 年 1 月 1 日の開始日と合わせます。
1899 年 1 月 1 日	STOD(*日付フィールド*) - 365	1900 年 1 月 1 日より前の*開始日*を指定することはできません。ソースデータファイルが 1900 年 1 月 1 日より前の開始日を使用する場合は、STOD( ) 関数の出力結果から適当な日数を減算する日付時刻式を作成することができます。

他の日付時刻変換関数

シリアルから日付時刻への変換

関数	説明
STODT( )	シリアル日付時刻、つまり、整数部分と24 時間の小数部分で表される日付時刻を日付時刻値に変換します。"Serial to Datetime" の省略形です。
STOT( )	シリアル時刻、つまり、24 時間を 1 として、24 時間が小数部分で表される時刻を時刻値に変換します。"Serial to Time" の省略形です。

文字または数値から日付時刻への変換

関数	説明
CTOD( )	文字または数値の日付値を日付に変換します。また、文字または数値の日付時刻値から日付を抽出し、それを日付として返すこともできます。"Character to Date" の省略形です。
CTODT( )	文字または数値の日付時刻値を日付時刻に変換します。"Character to Datetime" の省略形です。
CTOT( )	文字または数値の時刻値を時刻に変換します。また、文字または数値の日付時刻値から時刻を抽出し、それを時刻として返すこともできます。"Character to Time" の省略形です。

日付時刻から文字への変換

関数	説明
DATE( )	指定された日付または日付時刻から日付を抽出し、それを文字データとして返します。また、現在のオペレーティング システム日付を返すこともできます。
DATETIME( )	日付時刻を文字列に変換します。また、現在のオペレーティング システムの日付時刻を返すこともできます。
TIME( )	指定された時刻または日付時刻から時刻を抽出し、それを文字データとして返します。また、現在のオペレーティング システムの時刻を返すこともできます。

STODT( ) 関数

シリアル日付時刻、つまり、整数部分と24時間の小数部分で表される日付時刻を日付時刻値に変換します。"Serial to Datetime" の省略形です。

構文

STODT(*シリアル日付時刻* <,*開始日*>)

パラメーター

名前	種類	説明
シリアル日付時刻	数値	変換するフィールド、式、またはリテラル値。 日付と時刻の部分が小数点で区切られたシリアル日付時刻値が必要です。たとえば、42003.75000 のように指定します。
開始日 省略可能	日付時刻	シリアル日付の計算基準となる開始日。これを省略した場合は、デフォルトの開始日である 1900 年 1 月 1 日が使用されます。

出力

日付時刻。日付時刻値は、現在 Analytics に設定されている日付と時刻の表示書式を使用して出力されます。

例

基本的な例

開始日を調整しない場合

`20141231t060000` に対して現在の Analytics 日付時刻表示書式である DD MMM YYYY hh:mm:ss PM を適用した、31 Dec 2014 06:00:00 AM が表示されます。

```
STODT(42003.25000)
```

`20141231t191530` が返されますが、表示上は、現在の Analytics 日付時刻表示書式である DD MMM YYYY hh:mm:ss PM を適用した 31 Dec 2014 07:15:30 PM となります。

```
STODT(42003.802431)
```

開始日を調整した場合

`20181231t120000` が返されますが、表示上は、現在の Analytics 日付時刻表示書式である DD MMM YYYY hh:mm:ss PM を適用した 31 Dec 2018 12:00:00 PM として表示されます。

```
STODT(42003.50000, `19040101`)
```

入力用フィールド

Receipt_timestamp フィールドの各シリアル日付時刻値に対応する日付時刻が返されます。

```
STODT(Receipt_datetime)
```

高度な例

開始日を 1900-01-01 より前の日付に変更する

Analytics の最早の開始日である 1900 年 1 月 1 日より前の値に開始日を変更するには、日付の算術を行います。

1. デフォルトの開始日を使ってシリアル日付時刻を変換します。
2. 実際の開始日から 1900-01-01 までの日数を差し引きます。

次の例は、開始日として 1899-01-01 を使用した場合に `20131231t180000` と評価されます。

```
STODT(42003.75000) - 365
```

備考

機能の仕組み

STODT( ) 関数を使用すると、シリアル日付時刻を標準の日付時刻に変換することができます。Analytics のシリアル日付時刻は、1900 年 1 月 1 日からの経過日数を表し、小数点以下は、24 時間を 1 とする、24 時間の小数を表します。

シリアル日付時刻	相当する標準の日付時刻
1.25	1900 年 1 月 2 日午前 6 時 0 分 0 秒

シリアル日付時刻	相当する標準の日付時刻
365.75000	1900 年 12 月 31 日午後 6 時 0 分 0 秒
42003.79167	2014 年 12 月 31 日午後 7 時 0 分 0 秒
42003.802431	2014 年 12 月 31 日午後 7 時 15 分 30 秒
42003.00000	2014 年 12 月 31 日午前 12 時 0 分 0 秒
42003.50000	2014 年 12 月 31 日午後 12 時 0 分 0 秒
0.0	有効でない

シリアル日付時刻の詳細については、シリアル日付時刻を参照してください。

Analytics のシリアル日付と Excel シリアルの日付との比較

Analytics のシリアル日付は Microsoft Excel のシリアル日付に似ています。注意すべき類似のキーポイントが 1 つと、相違のキーポイントが 1 つあります。これら 2 つのポイントには関連性がありません。

類似のポイント

Analytics と Excel はどちらも、1900 年を 366 日ある閏年として扱います。1900 年は実際には閏年ではありませんでしたが、Excel は Lotus 1-2-3 との互換性を保つために、この年を閏年として処理しました。

差異の点

Analytics のシリアル日付は、Excel のシリアル日付から 1 日ずれています。Excel では、1900 年 1 月 1 日はシリアル日付 '1' になります。Analytics では、1900 年 1 月 1 日はカウントされないので、1900 年 1 月 2 日がシリアル日付 '1' になります。

開始日

一部のソースデータファイルは 1900 年 1 月 1 日以外の開始日を使用する場合があります。*開始日*は、ソースデータファイルの開始日と一致させることができます。シリアル日付時刻の計算基準となる開始日。

ソースデータファイルの開始日	次を指定します。	詳細
1900 年 1 月 1 日	STODT(*日付時刻フィールド*)	1900 年 1 月 1 日はデフォルト開始日であるため、*開始日*を指定する必要はありません。
1901 年 1 月 1 日	STODT(*日付時刻フィールド*, `19010101`)	*開始日* `19010101` を指定し、ソースデータファイルで使用される 1901 年 1 月 1 日の開始日と合わせます。
1899 年 1 月	STODT(*日付時刻フィールド*) - 365	1900 年 1 月 1 日より前の*開始日*を指定することはできません。

ソース データ ファイルの開始日	次を指定します。	詳細
1 日		ソース データ ファイルが 1900 年 1 月 1 日より前の開始日を使用する場合は、STODT( ) 関数の出力結果から適当な日数を減算する日付時刻式を作成することができます。

他の日付時刻変換関数

シリアルから日付時刻への変換

関数	説明
STOD( )	シリアル日付、つまり、整数で表される日付を日付値に変換します。"Serial to Date" の省略形です。
STOT( )	シリアル時刻、つまり、24 時間を 1 として、24 時間が小数部分で表される時刻を時刻値に変換します。"Serial to Time" の省略形です。

文字または数値から日付時刻への変換

関数	説明
CTOD( )	文字または数値の日付値を日付に変換します。また、文字または数値の日付時刻値から日付を抽出し、それを日付として返すこともできます。"Character to Date" の省略形です。
CTODT( )	文字または数値の日付時刻値を日付時刻に変換します。"Character to Datetime" の省略形です。
CTOT( )	文字または数値の時刻値を時刻に変換します。また、文字または数値の日付時刻値から時刻を抽出し、それを時刻として返すこともできます。"Character to Time" の省略形です。

日付時刻から文字への変換

関数	説明
DATE( )	指定された日付または日付時刻から日付を抽出し、それを文字データとして返します。また、現在のオペレーティング システム日付を返すこともできます。
DATETIME( )	日付時刻を文字列に変換します。また、現在のオペレーティング システムの日付時刻を返すこともできます。
TIME( )	指定された時刻または日付時刻から時刻を抽出し、それを文字データとして返します。また、現在のオペレーティング システムの時刻を返すこともできます。

STOT( ) 関数

シリアル時刻、つまり、24 時間を 1 として、24 時間が小数部分で表される時刻を時刻値に変換します。"Serial to Time" の省略形です。

構文

```
STOT(シリアル時刻)
```

パラメーター

名前	種類	説明
シリアル時刻	数値	変換するフィールド、式、またはリテラル値。 *シリアル時刻*には、シリアル時刻またはシリアル日付時刻を指定できます。シリアル日付時刻の時刻部分のみが考慮されます。日付部分は無視されます。

出力

日付時刻。時刻値は、現在 Analytics に設定されている時刻の表示書式を使用して出力されます。

例

基本的な例

`t060000` が返されますが、表示上は、現在の Analytics 時刻表示書式である hh:mm:ss PM を適用した 06:00:00 AM となります。

```
STOT(0.25000)
```

`t191530` が返されますが、表示上は、現在の Analytics 時刻表示書式である hh:mm:ss PM を適用した 07:15:30 PM となります。

```
STOT(0.802431)
```

Login_time フィールドの各シリアル時刻値に相当する標準の時刻が返されます。

```
STOT(Login_time)
```

備考

STOT( ) の使用に適する場面

STOT( ) 関数は、シリアル時刻を標準の時刻に変換する場合に使用できます。

シリアル時刻とは

Analytics のシリアル時刻は、24 時間を 1 として、24 時間制の時刻を小数として表したものです。

例：

- 1 時間に相当するシリアル時刻は 1/24、つまり 0.04167 です。
- 1 分に相当するシリアル時刻は 1/1440、つまり 0.0006945 です。

シリアル時刻の先頭には、'0'(ゼロ) と小数点、または小数点のみを使用できます。

1.000000 は有効なシリアル時刻ではありません。

シリアル時刻を計算する目的のために、24 時間は 1 に等しいとしていますが、1.000000 は有効なシリアル時刻ではありません。有効なシリアル時刻は 1 未満のすべての小数です。例：0.75000(午後 6 時)

Analytics はシリアル値 1.000000 を、1900 年 1 月 2 日午前 12 時に対応するシリアル日付時刻として扱います。STOT( ) は日付時刻の日付部分を無視するため、STOT(1.000000) は STOT(0.000000) と同等となり、どちらも標準時刻の午前 12 時に相当します。

シリアル時刻とそれに相当する標準の時刻

シリアル時刻	相当する標準の時刻
0.00	12:00:00 AM
0.0006945	12:01:00 AM
0.04167	01:00:00 AM
0.0423645	01:01:00 AM
0.042998	01:01:55 AM
0.25	06:00:00 AM
0.50	12:00:00 PM

シリアル時刻	相当する標準の時刻
0.75	06:00:00 PM
0.79167	07:00:00 PM
0.802431	07:15:30 PM
1.00	12:00:00 AM

他の日付時刻変換関数

シリアルから日付時刻への変換

関数	説明
STOD( )	シリアル日付、つまり、整数で表される日付を日付値に変換します。"Serial to Date" の省略形です。
STODT( )	シリアル日付時刻、つまり、整数部分と24 時間の小数部分で表される日付時刻を日付時刻値に変換します。"Serial to Datetime" の省略形です。

文字または数値から日付時刻への変換

関数	説明
CTOD( )	文字または数値の日付値を日付に変換します。また、文字または数値の日付時刻値から日付を抽出し、それを日付として返すこともできます。"Character to Date" の省略形です。
CTODT( )	文字または数値の日付時刻値を日付時刻に変換します。"Character to Datetime" の省略形です。
CTOT( )	文字または数値の時刻値を時刻に変換します。また、文字または数値の日付時刻値から時刻を抽出し、それを時刻として返すこともできます。"Character to Time" の省略形です。

日付時刻から文字への変換

関数	説明
DATE( )	指定された日付または日付時刻から日付を抽出し、それを文字データとして返します。また、現在のオペレーティング システム日付を返すこともできます。
DATETIME( )	日付時刻を文字列に変換します。また、現在のオペレーティング システムの日付時刻を返すこともできます。

関数	説明
TIME( )	指定された時刻または日付時刻から時刻を抽出し、それを文字データとして返します。また、現在のオペレーティング システムの時刻を返すこともできます。

STRING( ) 関数

数値を文字列に変換します。

構文

```
STRING(数値, 長さ<,書式>)
```

パラメーター

名前	種類	説明
数値	数値	文字に変換する数値。
長さ	数値	出力文字列の文字数。
書式 省略可能	文字	出力文字列に適用する書式。例：`"(9,999.99)"`

出力

文字。

例

基本的な例

文字列を書式設定しない

" 125.2" が返されます。

```
STRING(125.2, 6)
```

*長さ*が*数値*内の数字と書式文字の数よりも短いので、-1 が切り詰められて、"25.2" が返されます。

```
STRING(-125.2, 4)
```

"-125.2" が返されます。

```
STRING(-125.2, 7)
```

文字列を書式設定する

" (125.20)" が返されます。

```
STRING(-125.2, 10, "(9,999.99)")
```

*長さ*が*数値*内の数字と書式文字の数よりも短いので、1 が切り詰められて、"25.20" が返されます。

```
STRING(125.2, 6, "(9,999.99)")
```

フィールドへの入力値

文字列型の長さ 10 文字の数値を **Employee_number** フィールドが返されます。必要に応じて、戻り値では空白が埋め込まれたり切り捨てが行われたりします。

```
STRING(Employee_number, 10)
```

備考

戻り値へのスペースの追加と戻り値の切り捨て

STRING( ) は*数値*を*長さ*で指定された長さの文字列に変換します。

- *数値*が*長さ*より短い場合は、戻り値の先頭にスペースが追加されます。
- *数値*が*長さ*より長い場合、戻り値は左側から切り詰められます。

戻り値の書式設定

オプションの*書式*パラメーターは、ドル記号、パーセント記号、小数点、カンマ、負数のインジケーター、あるいはかっこのような書式を戻り値に追加します。*書式*は二重引用符で囲む必要があります。

数字の 9 は数字書式のプレースホルダーとして働きます。適切に表示するために正しい数の 9 を含んでいることを確認してください。*長さ*の値を指定するとき、小数位や、ドル記号、負数のためのかっこなどの書式文字列も考慮に入れる必要があります。

関連する関数

STRING( ) 関数は VALUE( ) 関数の逆関数です。後者は、文字データを数値データに変換します。

SUBSTR( ) 関数

文字列のうちの指定された部分文字列を返します。

構文

```
SUBSTR(文字列, 開始位置, 長さ)
```

パラメーター

名前	種類	説明
文字列	文字	部分文字列を抽出する値。
開始	数値	部分文字列の開始文字位置。
長さ	数値	部分文字列の文字数。 *長さ*が 0 の場合は、出力は空です。

出力

文字。

例

基本的な例

リテラル文字の入力値

"BCD" が返されます。

```
SUBSTR("ABCDEF", 2, 3)
```

"EF" が返されます。

```
SUBSTR("ABCDEF", 5, 10)
```

構造的な文字データを解析する

"189543" が返されます。

```
SUBSTR("***189543***", 4, 6)
```

"MM/DD/YYYY" として書式設定された日付が含まれる文字フィールドのうち、4 桁の年

```
SUBSTR(DATE, 7, 4)
```

高度な例

フィールド長を大きくする

文字フィールドの長さを大きくするのにSUBSTR( ) を使用できます。フィールド長を大きくすることは、2 つのフィールドを結合したり追加したりする前に必要に応じて行うことができる一般的な調整作業です。

次の例は、**Product_Description** フィールドにスペースを追加して、長さが50 文字の演算フィールド、**Product_Description_Long** を作成しています。

```
DEFINE FIELD Product_Description_Long COMPUTED SUBSTR(Product_Description, 1, 50)
```

備考

機能の仕組み

SUBSTR( ) 関数は、*文字列*値内の、*開始位置*で指定された桁位置から始まる文字列を返します。*長さ*には、返される文字数を指定します。

SUBSTR( ) がスペースを処理する方法

*文字列*値の先頭、末尾、または内部のスペースは文字として処理されます。*開始位置*および*長さ*で取り込まれたスペースは出力文字列に含まれます。

スペースの追加の動作

*長さ*の値が*開始*位置から*文字列*の終わりまでの文字数(スペースを含む) を超える場合は、出力の右側にスペースが追加される場合とされない場合があります。

スペースが追加される場合とは

フィールドを作成するコマンド内でSUBSTR( ) を使用する場合は、出力にスペースが追加されます。

演算フィールドを作成する場合のスペースの追加

次の例は、長さが24文字の物理フィールド、**Product_Description**に基づいて、長さが50文字の演算フィールド、**Product_Description_Long**を作成しています。

```
DEFINE FIELD Product_Description_Long COMPUTED SUBSTR(Product_Description, 1, 50)
```

物理フィールドを抽出する場合のスペースの追加

次の例は、長さが24文字の物理フィールド、**Product_Description**に基づいて、長さが50文字の演算フィールド、**Product_Description_Long**を新しいテーブルに抽出しています。

```
EXTRACT FIELDS SUBSTR(Product_Description, 1, 50) AS "Product_Description_Long" TO New_Table
```

スペースが追加されない場合とは

変数定義または式でSUBSTR( )を使用する場合は、出力にはスペースが追加されません。

変数を定義する場合にはスペースが追加されない

次の例は、**Product_Description**のフィールド長に基づいて、長さが24文字の変数、*v_prod_desc*を作成しています。

```
ASSIGN v_prod_desc = SUBSTR(Product_Description, 1, 50)
```

> **メモ**
>
> SUBSTR( )の長さに50文字を指定していても、出力は**Product_Description**のフィールド長に制限されます。

関連する関数

SUBSTR( )とSPLIT( )はともに、長いソース文字列からデータのセグメントを返します。

- SUBSTR( )は数値文字の位置を使ってセグメントを識別します。
- SPLIT( )は区切り文字を使ってセグメントを識別します。

TAN( ) 関数

ラジアン単位で表された角度のタンジェントを、小数点以下 15 桁の精度で返します。

構文

```
TAN(ラジアン)
```

パラメーター

名前	種類	説明
ラジアン	数値	ラジアン単位の角度。

出力

数値。

例

基本的な例

指定したラジアン数 (45 度に相当) に対し、その正接である 0.999999999999999 が返されます。

```
TAN(0.785398163397448)
```

45 度に対し、その正接である 0.999999999999999 が返されます。

```
TAN(45 * PI( )/180)
```

高度な例

入力値として度を使用する

45 度に対し、その正接を小数点以下 3 桁に丸めた 1.000 が返されます。


```
DEC(TAN(45 * PI( )/180),3)
```

備考

Mantissa Arc Test の実行

Analytics の 3 つの三角関数 SIN( )、COS( )、および TAN( ) は、ベンフォードの法則に関連する Mantissa Arc Test の実行をサポートします。

度のラジアンへの変換

入力値が度で表されている場合は、PI( ) 関数を使用して角度をラジアンに変換できます。(*度* * PI( )/180) = *ラジアン*となります。必要に応じて、DEC( ) 関数を使用して、戻り値を丸めたり切り捨てたりすることができます。

TEST( ) 関数

指定された文字列がレコード内の特定のバイト位置に現れるかどうか示す論理値を返します。

構文

TEST(*バイト位置*, *文字列*)

パラメーター

名前	種類	説明
バイト位置	数値	*文字列*の先頭文字の位置を確認する、テーブルレイアウトでの左からの連番です。 先頭文字の位置を*文字列*の開始位置とすることができない場合は、その*文字列*がレコードの別の位置に存在しているとしても、この関数はF(false) と評価されます。
文字列	文字	検索する文字列。 検索は大文字と小文字を区別します。大文字と小文字の両方が混在している可能性がある場合は、UPPER( ) 関数を使用してすべての文字を大文字に変換します。

出力

論理。指定された文字列がレコード内の指定されたバイト位置から始まっている場合はT(true) 、そうでない場合はF(false) を返します。

例

基本的な例

次のようなレコードを例に説明します。

```
Department: Marketing
....|....|....|....|....|
```

T が返されます。

```
TEST(5, "Department")
```

レコードにおける "Department" の開始バイト位置が 6 でなく 5 のため、F が返されます。

```
TEST(6, "Department")
```

この関数は大文字と小文字が区別されるため、F が返されます。

```
TEST(5, "DEPARTMENT")
```

高度な例

ページ見出しになっているレコードを抽出する

TEST( ) を使用することで、"Page:" で始まるすべてのレコードを抽出するフィルターを作成できます。

```
SET FILTER TO TEST(1, "Page:")
```

TIME( ) 関数

指定された時刻または日付時刻から時刻を抽出し、それを文字データとして返します。また、現在のオペレーティングシステムの時刻を返すこともできます。

構文

```
TIME(<時刻/日付時刻> <,書式>)
```

パラメーター

名前	種類	説明
時刻/日付時刻 省略可能	日付時刻	時刻を抽出するフィールド、式、またはリテラル値。これを省略した場合は、現在のオペレーティングシステムの時刻が hh:mm:ss 書式で返されます。
書式 省略可能	文字	出力文字列に適用する書式。例："hh:mm:ss" このパラメーターを省略した場合は、現在の Analytics の時刻表示書式が使用されます。*時刻/日付時刻*を省略した場合に、*書式*を指定することはできません。

出力

文字。

例

基本的な例

リテラルの入力値

現在の Analytics 時刻表示書式である hh:mm:ss を適用した時刻、"23:59:59" が返されます。

```
TIME(`20141231 235959`)
```

"11:59 P" が返されます。

```
TIME(`20141231 235959`, "hh:mm A")
```

現在のオペレーティングシステム時刻がhh:mm:ss書式(24時間制)で文字列として返されます。

```
TIME()
```

入力値用フィールド

Receipt_timestampフィールドの各値に対して現在のAnalyticsの時刻表示書式を適用した文字列が返されます。

```
TIME(Receipt_timestamp)
```

Receipt_timestampフィールドの各値に対して、指定した時刻表示書式を適用した文字列が返されます。

```
TIME(Receipt_timestamp, "hh:mm:ss")
```

高度な例

実行するコマンドまたはスクリプトの経過時間を計算する

特定のAnalyticsコマンドまたはスクリプト全体の実行にかかる時間を計算するためにTIME()関数を使用することができます。

時間を計測するコマンドの直前、またはスクリプトの開始時に、現在のオペレーティングシステムの時間を格納する変数を作成するために次の行を指定します。

```
ASSIGN Time_started = TIME()
```

コマンドの実行直後、またはスクリプトの終了時に次の2行を指定します。

最初の行は、コマンドまたはスクリプトが完了した後にオペレーティングシステムの時間を格納する変数を作成します。2行目は終了と開始時間の差を計算して、結果を見やすい書式に変換します。

> **ヒント**
>
> コマンドやスクリプトの経過時間は、CALCULATEログエントリをダブルクリックしても見ることができます。

```
ASSIGN Time_finished = TIME()
CALCULATE STOT(CTOT(Time_finished) - CTOT(Time_started))
```

コマンドまたはスクリプトが日付をまたいで実行される場合は、代わりにこの2行を使用します。

```
CALCULATE `T000000` - (CTOT(Time_started) - CTOT(Time_finished))
```

備考

出力文字列の長さ

出力文字列の長さは常に14文字です。指定した出力*書式*、またはAnalyticsの時刻表示書式が12文字より短い場合は、出力文字列の末尾に空白が埋められます。

パラメーターの詳細

*時刻/日付時刻*に指定されたフィールドは、フィールド定義で正しく書式を定義してさえいれば、どのような時刻書式または日付時刻書式でも使用することができます。

*書式*を使用して出力文字列の表示方法を制御する場合は、次の表内の書式に制限されます。時刻とAM/PMの書式を任意に組み合わせて使用することができます。AM/PM書式は省略可能で、最後に置きます。

*書式*は、一重引用符または二重引用符を使用して指定します。例：`"hh:mm:ss AM"`

時刻書式	AM/PM書式	例
hh:mm:ss	なし 24時間制	"hh:mm:ss"
hhmmss	AMまたはPM 12時間制	"hhmmss PM"
hh:mm	AまたはP 12時間制	"hh:mm A"
hhmm		
hh		

リテラル時刻または日付時刻値の指定

*日付時刻*にリテラルの時刻値または日付時刻値を指定する場合は、次の表内の書式に制限されます。また、`` `20141231 235959` `` のように、値を逆引用符で囲む必要があります。

日付または時刻の個々の要素の間に、スラッシュ(/) やコロン(:) のような区切り文字をいっさい使用しないでください。

- **時刻値** - 以下の表に示す任意の時刻の書式を使用することができます。関数が正しく動作するためには、単独の時刻値の前に区切り文字を使用する必要があります。有効な区切り文字は文字 't' または 'T' です。24時間形式で時刻を指定する必要があります。UTC(Coordinated Universal Time：協定世界時) からのオフセットは、プラス記号(+) またはマイナス記号(-) で始める必要があります。
- **日付時刻値** - 以下の表に示す日付、区切り文字、および時刻の書式を任意に組み合わせて使用す

ることができます。日付を時刻の前に置く必要があり、かつ、2つの間に区切り文字を使用する必要があります。有効な区切り文字は空白スペース1つ、あるいは文字 't' または 'T' です。

形式の例	リテラル値の例
thhmmss	`t235959`
Thhmm	`T2359`
YYYYMMDD hhmmss	`20141231 235959`
YYMMDDthhmm	`141231t2359`
YYYYMMDDThh	`20141231T23`
YYYYMMDD hhmmss+/-hhmm (UTC オフセット)	`20141231 235959-0500`
YYMMDD hhmm+/-hh (UTC オフセット)	`141231 2359+01`
メモ UTC オフセットが設定されているデータのメインの時刻書式で hh を単独で使用しないでください。たとえば、「hh+hhmm」という使い方は避けてください。信頼できない結果になる可能性があります。	

関連する関数

日付時刻値として現在のオペレーティングシステム時刻付を返すには、TIME( ) の代わりにNOW( ) を使用してください。

他の日付時刻変換関数

日付時刻から文字への変換

関数	説明
DATE( )	指定された日付または日付時刻から日付を抽出し、それを文字データとして返します。また、現在のオペレーティングシステム日付を返すこともできます。
DATETIME( )	日付時刻を文字列に変換します。また、現在のオペレーティングシステムの日付時刻を返すこともできます。

文字または数値から日付時刻への変換

関数	説明
CTOD( )	文字または数値の日付値を日付に変換します。また、文字または数値の日付時刻値から日付を抽出し、それを日付として返すこともできます。"Character to Date" の省略形です。
CTODT( )	文字または数値の日付時刻値を日付時刻に変換します。"Character to Datetime" の省略形です。
CTOT( )	文字または数値の時刻値を時刻に変換します。また、文字または数値の日付時刻値から時刻を抽出し、それを時刻として返すこともできます。"Character to Time" の省略形です。

シリアルから日付時刻への変換

関数	説明
STOD( )	シリアル日付、つまり、整数で表される日付を日付値に変換します。"Serial to Date" の省略形です。
STODT( )	シリアル日付時刻、つまり、整数部分と24 時間の小数部分で表される日付時刻を日付時刻値に変換します。"Serial to Datetime" の省略形です。
STOT( )	シリアル時刻、つまり、24 時間を 1 として、24 時間が小数部分で表される時刻を時刻値に変換します。"Serial to Time" の省略形です。

TODAY( ) 関数

現在のオペレーティングシステム日付を日付時刻データ型で返します。

構文

```
TODAY()
```

パラメーター

この関数にはパラメーターはありません。

出力

日付時刻。

例

基本的な例

現在のオペレーティングシステム日付が、現在のAnalytics日付書式を使用して表示された日付時刻値 (`20141231` など) として返されます。

```
TODAY()
```

備考

関連する関数

現在のオペレーティングシステム日付を文字列として返させたい場合には、TODAY( ) でなくDATE( ) を使用してください。

TRANSFORM( ) 関数

指定された文字列内の双方向テキストの表示方向を逆にします。

構文

```
TRANSFORM(元の文字列)
```

パラメーター

名前	種類	説明
元の文字列	文字	双方向のテキストを内容とした文字列。

出力

文字。

例

基本的な例

入力文字列において、文字列 "XZQB" は入力文字列におけるヘブライ語/双方向文字を表します。入力文字列には、それ以外の文字としては、通常の文字が含まれています。

出力文字列では、"XZQB" の方向は逆になり、"BQZX" が返されます。"XZQB" 以外の文字列は変更されません。

"ABC BQZX 123" が返されます。

```
TRANSFORM("ABC ZYX 123")
```

備考

機能の仕組み

TRANSFORMS( ) 関数は双方向データを識別して、そのテキストを右から左へ正しく表示します。

この関数で処理されるその他すべての文字列は、変更されず、引き続き左から右へ表示されます。

TRANSFORMS( ) の使用に適する場面

TRANSFORMS( ) 関数は、アラビア語やヘブライ語の文字が正しく表示されるように、文字の表示方向を変更する場合に使用できます。

TRIM( ) 関数

入力文字列から末尾のスペースを除去した文字列を返します。

構文

```
TRIM(文字列)
```

パラメーター

名前	種類	説明
文字列	文字	末尾のスペースを除去する値。

出力

文字。

例

基本的な例

両方の例において、先頭のスペースと単語間のスペースがTRIM() 関数によって除去されないことに注意してください。

" Vancouver" が返されます。

```
TRIM("  Vancouver  ")
```

" New York" が返されます。

```
TRIM("  New York")
```

高度な例

改行なしスペースの削除

改行なしスペースはTRIM( ) 関数で削除されません。

末尾の改行なしスペースを削除する必要がある場合は、次の式を使用して演算フィールドを作成します。

```
DEFINE FIELD Description_cleaned COMPUTED TRIM(REPLACE(Description, CHR(160), CHR(32)))
```

REPLACE( ) 関数がすべての改行なしスペースを標準のスペースに置換してから、TRIM( ) がすべての末尾の標準スペースを削除します。

備考

機能の仕組み

TRIM( ) 関数は末尾のスペースだけを除去します。文字列の内部スペースおよび先頭のスペースは除去されません。

関連する関数

TRIM( ) 関数の関連関数として、文字列の先頭のスペースを除去するLTRIM( ) 関数と、先頭と末尾の両方のスペースを除去するALLTRIM( ) 関数があります。

UNSIGNED( ) 関数

符号なしデータ型に変換された数値データを返します。

構文

```
UNSIGNED(数値, 結果の長さ)
```

パラメーター

名前	種類	説明
数値	数値	変換する値。
結果の長さ	数値	出力文字列で使用するバイトの数。

出力

数値。

例

基本的な例

000075 が返されます。

```
UNSIGNED(75, 3)
```

```
UNSIGNED(-75, 3)
```

```
UNSIGNED(7.5, 3)
```

2456 が返されます(*結果の長さ*が 2 のため、4 桁しか格納できないので、1 が切り詰められます) 。

```
UNSIGNED(12456, 2)
```

000000012456 が返されます。

```
UNSIGNED(-12.456, 6)
```

備考

符号なしデータとは

符号なしデータ型は、メインフレームオペレーティングシステムで使用されるデータ型で、1 バイトにつき 2 桁の数字を格納する、最小の記憶域を使用する形式で数値を格納します。符号なしデータ型はパックデータ型と同じですが、このデータ型は、値が正負いずれであるかを示すために最終バイトを使用しません。

UNSIGNED( ) の使用に適する場面

UNSIGNED( ) 関数は、メインフレームシステムへのエクスポートのために、数値データを符号なし形式に変換する場合に使用できます。

戻り値が切り捨てられる場合とは

*結果の長さ*の値が*数値*の長さより短い場合は、余分な桁が切り詰められます。

UPPER( ) 関数

アルファベット文字を大文字に変換した文字列を返します。

構文

```
UPPER(文字列)
```

パラメーター

名前	種類	説明
文字列	文字	大文字に変換する値。

出力

文字。

例

基本的な例

"ABC" が返されます。

```
UPPER("abc")
```

"ABC 123 DEF" が返されます。

```
UPPER("abc 123 DEF")
```

"ABCD 12" が返されます。

```
UPPER("AbCd 12")
```

Last_Name フィールドのすべての値を大文字に変換した値が返されます。

```
UPPER(Last_Name)
```

備考

機能の仕組み

UPPER( ) 関数は*文字列*のすべてのアルファベットを大文字に変換します。アルファベット以外の文字は変換されません。

UPPER( ) の使用に適する場面

UPPER( ) は、フィールド、変数、または式のすべての文字を大文字に統一する必要がある場合に使用できます。大文字への統一は、値を比較する場合には特に重要です。

また、UPPER( ) はレポート内の値を大文字テキストとして書式設定する場合にも使用できます。

UTOD( ) 関数

書式設定された日付が含まれるUnicode文字列をAnalytics日付値に変換します。"Unicode to Date" の省略形です。

> **メモ**
>
> この関数はAnalyticsのUnicode版に固有です。非Unicode版ではサポートされない関数です。
>
> この関数は、デフォルトのインストールとは異なる言語および書式の日付を操作する場合に使用します。変換元の文字列をデフォルトの言語で表す場合は、代わりにCTOD( )を使用します。

構文

UTOD(*文字列* <,*ロケール*> <,*スタイル*>)

パラメーター

名前	種類	説明
文字列	文字	日付に変換するUnicode文字列。 Unicode文字列に日付時刻値を入れることはできますが、値の時刻部分は無視されます。単独の時刻値はサポートされていません。 *文字列*は、日付のロケールの*スタイル*に必要な入力書式と一致している必要があります。
ロケール 省略可能	文字	出力文字列の言語およびロケール(と、オプションで、特定の国や地域に関連付けられている言語バージョン)を指定するコード。 たとえば、"zh" では中国語が指定され、"pt_BR" ではブラジルのポルトガル語が指定されます。 これを省略した場合は、コンピューターに設定されているデフォルトのロケールが使用されます。言語コードを指定したが、国コードを指定しなかった場合は、言語コードにデフォルトで設定されている国コードが適用されます。 *日付*を指定しなかった場合に、*ロケール*を指定することはできません。 ロケールコードに関する情報については、www.unicode.org を参照してください。
スタイル 省略可能	数値	Unicode文字列に使用する日付書式スタイル。この書式スタイルは、指定したロケールの標準書式スタイルと一致します。 ◦ 0 - "Sunday, September 18, 2016" などの完全仕様の書式 ◦ 1 - "September 18, 2016" などの長い書式

名前	種類	説明
		◦ 2 - "Sep 18, 2016" などの中ぐらいの長さの書式 ◦ 3 - "9/18/16" などの短い数値型書式 これを省略した場合は、デフォルト値の **2** が使用されます。*ロケール*を指定しなかった場合に、*スタイル*を指定することはできません。 **ヒント** 入力文字列に必要な書式を判定するには、以下のいずれかを行います。 • スタイルとロケールを指定して DTOU( ) 関数を実行して、値の例を生成します。 この値を出力するには、コマンド ラインで DISPLAY コマンドを実行します。 `DISPLAY DTOU(`20160909`, "es_MX", 3)` • 特定のロケールでのスタイルの標準的な日付書式については、信頼できる情報源を参照してください。

出力

日付時刻。日付値は、現在 Analytics に設定されている日付の表示書式を使用して出力されます。

例

基本的な例

> **メモ**
>
> どの例でも、現在の Analytics 日付表示書式は DD MMM YYYY であるとします。
>
> 下の例で、中国語 ("zh") および簡体字中国語 ("zh_CN") のロケールコードはさまざまな入力文字列に一致しますが、互いに交換することはできません。
>
> また、正しい*スタイル*を指定する必要があります。*スタイル*2 を指定した場合、長い形式の Unicode 日付文字列 (つまり、*スタイル*は 1) は Analytics 日付を返しません。

リテラルの入力値

31 Dec 2014 として表示される `20141231` を返します。

```
UTOD("31 de dezembro de 2014", "pt_BR", 1)
```

31 Dec 2014 として表示される `20141231` を返します。

```
UTOD("31 grudnia 2014", "pl", 1)
```

フィールドの入力値

Invoice_date フィールドの各 Unicode 文字列に対応する数値の日付が返されます。

```
UTOD(Invoice_date, "zh", 1)
```

入力では完全な日付スタイルが使用されます。

31 Dec 2014 として表示される `20141231` を返します(地域 ID 未指定)。

```
UTOD("星期三, 2014 十二月 31", "zh", 0)
```

31 Dec 2014 として表示される `20141231` を返します(地域 ID 指定)。

```
UTOD("2014年12月31日 星期三", "zh_CN", 0)
```

入力では長い形式の日付スタイルが使用されます。

31 Dec 2014 として表示される `20141231` を返します(地域 ID 未指定)。

```
UTOD("2014 十二月 31", "zh", 1)
```

31 Dec 2014 として表示される `20141231` を返します(地域 ID 指定)。

```
UTOD("2014年12月31日", "ja_JP", 1)
```

備考

Unicode 文字列を正しく変換する

日付を含んでいる Unicode 文字列を Analytics 日付に正しく変換するには、*ロケール* パラメーターと*スタイル* パラメーターを、Unicode 文字列内の日付の言語、国/地域(該当する場合)、およびスタイルと一致するように指定する必要があります。

関連する関数

UTOD( ) は、日付を Unicode 文字列に変換する DTOU( ) 関数の逆関数です。UTOD( ) に指定する国/地域およびスタイルが不確かな場合は、DTOU( ) を使って、何を指定したら UTOD( ) で変換したい入力 Unicode

文字列の形式に一致する出力 Unicode 文字列を生成できるか、さまざまなパラメーターを試してみることができます。

VALUE( ) 関数

文字列を数値に変換します。

構文

```
VALUE(文字列, 小数位)
```

パラメーター

名前	種類	説明
文字列	文字	変換するフィールド、リテラル、または式の値。
小数位	数値	出力に含める小数点以下桁数。

出力

数値。

例

基本的な例

-123.400 が返されます。

```
VALUE("123.4-", 3)
```

123456.00 が返されます。

```
VALUE("$123,456", 2)
```

-77.45 が返されます。

```
VALUE("77.45CR", 2)
```

-123457 が返されます。

```
VALUE(" (123,456.78)", 0)
```

フィールドへの入力値

小数位のない数値としての文字型の値が**Salary**(給与)フィールドに返されます。

```
VALUE(Salary, 0)
```

備考

機能の仕組み

この関数は文字データを数値データに変換します。VALUE( ) 関数は、Analyticsコマンドで使用するために、文字式やフィールド値を数値に変換する必要がある場合に使用できます。

入力数値の書式設定

VALUE( ) はどのような書式の数値でも受け付けます。入力として、句読点、先頭または末尾の記号、かっこなど、Printデータ型で受け付けられる数値書式はすべて使用できます。

負の値

VALUE( ) 関数は、かっこやマイナス記号などさまざまな負数のインジケーターを解釈することができます。さらに、CR(貸方)およびDR(借方)を解釈することができます。例:

-1000.00が返されます。

```
VALUE("(1000)", 2)
```

```
VALUE("1000CR", 2)
```

小数値と整数値

*文字列*値に小数点以下の桁を含めない場合には、文字列の数値は整数として処理されます。例:

123.00が返されます。

```
VALUE("123", 2)
```

*小数位*に指定された小数点以下の桁数がフィールドや式の小数点以下桁数より少ない場合、結果は丸められます。例:

"10.6" が返されます。

```
VALUE("10.56", 1)
```

関連する関数

VALUE( ) 関数は STRING( ) 関数の逆関数です。後者は、数値データを文字データに変換します。

VERIFY( ) 関数

物理データフィールドのデータが有効かどうか示す論理値を返します。

構文

```
VERIFY(フィールド)
```

パラメーター

名前	種類	説明
フィールド	文字 数値 日付時刻	物理データフィールドである必要があります。

出力

論理。フィールドのデータが有効な場合は **T**(true) 、そうでない場合は **F**(false) を返します。

例

基本的な例

VERIFY( ) 関数が False と評価したレコードを新しい Analytics テーブルに抽出します。

```
EXTRACT RECORD IF NOT VERIFY(Address) TO "InvalidEntries.fil"
```

備考

VERIFY( ) 関数は、フィールドのデータがそのフィールドに指定されたデータ型と一致しているかどうかを判断します。

VERIFY( ) の使用に適する場面

この関数は、VERIFY コマンドや [**オプション**]ダイアログ ボックス([**ツール > オプション**]) の [**数値**]タブにある [**データを検証する**]オプションを用いるよりも、検証するフィールドをより厳密に制御できるようにします。この関数を使用して、個々のフィールドのエラーを検出して状況に応じた処理を行えます。

この関数が true と評価される場合とは

関数が true に評価するための条件は次のとおりです。

- 文字フィールドには、文字、数字、および記号などの、有効な印刷可能な文字のみが入っている
- 数値フィールドには、数字、小数点、および通貨記号などの、有効な数値文字のみが入っている
- 日付時刻フィールドには、有効な日付、日付時刻、あるいは時刻のみが入っている

演算フィールドと式

演算フィールドと式は常に T(true) と評価されます。そのため、まず演算フィールドや式を物理フィールドに変換しない限り、それらに対してこの関数を使用することはできません。 [**抽出**]ダイアログ ボックスの [**フィールド**]オプションを使用して演算フィールドおよび式を抽出することにより、それらを物理フィールドに変換します。

WORKDAY( ) 関数

2 つの日付の間の就業 (稼働) 日数を返します。

構文

WORKDAY(*開始日*, *終了日* <,*休日*>)

パラメーター

名前	種類	説明
開始日	日付時刻	就業日の計算対象期間の開始日。開始日は期間内に含まれます。
終了日	日付時刻	就業日の計算対象期間の終了日。終了日は期間内に含まれます。
休日 省略可能	文字	週末または休日に当たるため、計算から除外する曜日。*休日*を省略する場合は、土曜日と日曜日がデフォルトの休日として使用されます。 *休日*を入力する場合は、次の略語をカンマまたはスペースで区切ります。 ◦ Mon ◦ Tue ◦ Wed ◦ Thu ◦ Fri ◦ Sat ◦ Sun *休日*では大文字と小文字が区別されません。英語版以外の Analytics を使用している場合でも、略語は英語で入力する必要があります。 "Fri, Sat, Sun"

> **メモ**
>
> *開始日*や*終了日*には日付時刻値を指定することもできますが、値の時刻部分は無視されます。
>
> *開始日*が*終了日*より後の場合は、負の数値が返されます。

出力

数値。就業日の計算対象期間内の就業日数。

例

基本的な例

リテラルの入力値

2015年3月2日月曜日から2015年3月8日日曜日までの就業日数、5が返されます。

```
WORKDAY(`20150302`, `20150308`)
```

日曜日のみが休日の場合の、2015年3月2日月曜日から2015年3月8日日曜日までの就業日数、6が返されます。

```
WORKDAY(`20150302`, `20150308`, "Sun")
```

金曜日と土曜日が休日の場合の、2015年3月1日日曜日から2015年3月7日土曜日までの就業日数、5が返されます。

```
WORKDAY(`20150301`, `20150307`, "Fri Sat")
```

フィールドの入力値

開始日フィールドの各日付から2015年12月31日までの就業日数が返されます。

```
WORKDAY(Start_date, `20151231`)
```

Start_dateフィールドの各日付から**End_date**フィールドの対応する日付までの就業日数が返されます。

- 法定祝日は就業日合計に含まれるので、別個の計算を使用して除外しなければならない場合があります。
- 戻り値が負の場合は、開始日が終了日よりも後であることを示します。

```
WORKDAY(Start_date, End_date)
```

備考

日付書式

*開始日*または*終了日*に指定されたフィールドは、フィールド定義で正しく書式を定義してさえいれば、どのような日付書式でも使用することができます。

*開始日*または*終了日*にリテラルの日付値を指定する場合は、YYYYMMDD とYYMMDD 書式に制限されます。また、`20141231` のように、値を逆引用符で囲む必要があります。

土曜日と日曜日以外の休日

土曜日と日曜日以外の休日を指定すると、月曜日から金曜日までの就業日または週に5 日の就業日に基づかないデータで WORKDAY( ) 関数を使用できます。

たとえば、Sun を休日に指定すると、月曜日から土曜日までの週 6 日を就業日とすることができます。

法定祝日の考慮

WORKDAY( ) 関数は法定祝日を考慮しません。つまり、期間内に1 日以上の法定祝日が含まれている場合、戻り値に期間内の実際の就業日数が反映されないことがあります。

ScriptHub の「就業日数の計算」スクリプト。

法定祝日を考慮する必要がある場合は、ScriptHub で、ユーザー定義の休日のリストを受け入れる「就業日の計算」スクリプトを使用します。

より期間が長く、多数の祝日が含まれるデータの場合、一般的に、このスクリプトを使用する方が簡単です。詳細については、Analytics のヘルプトピック「ScriptHub からのスクリプトのインポート」を参照してください。

四半期などの3、4 日しか祝日を含まない短い期間の場合は、以下で説明する条件付き演算フィールドを簡単に作成できることがあります。

法定祝日を減算するための条件付き演算フィールド

必要に応じて、条件付き演算フィールドを作成し、WORKDAY( ) 関数で返された値から法定祝日を差し引くことができます。

たとえば、2015 年の第 1 四半期の場合、特定の期間に含まれる以下の各祝日について、WORKDAY( ) の戻り値を 1 ずつ減算できます。

- 2015/01/01
- 2015/01/19
- 2015/2/16

次の例では、この四半期内に開始日と終了日がある複数の期間を示します。

まず、**Workdays** などの演算フィールドを作成し、四半期中の特定の期間の就業日数を計算します。

```
DEFINE FIELD Workdays COMPUTED WORKDAY(Start_date, End_date)
```

次に、先の演算フィールド(**Workdays**) に返された値を調整する、**Workdays_no_holidays** などの条件付き演算フィールドを作成します。

```
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
```

メモ

この条件付き演算フィールド内での条件の順序は重要です。

Analytics では、複数の条件を上から順に評価します。あるレコードに対して真と評価される最初の条件が、そのレコードの条件付き演算フィールドの値を割り当てます。真と評価される後続の条件は、割り当てられた値を変更しません。

YEAR( ) 関数

指定された日付または日付時刻から年を抽出し、それを YYYY 書式の数値として返します。

構文

```
YEAR(日付/日付時刻)
```

パラメーター

名前	種類	説明
日付/日付時刻	日付時刻	年を抽出するフィールド、式、またはリテラル値。

出力

数値。

例

基本的な例

2014 が返されます。

```
YEAR(`20141231`)
```

```
YEAR(`141231 235959`)
```

Invoice_date フィールドの各値に対して年が返されます。

```
YEAR(Invoice_date)
```

備考

パラメーターの詳細

*日付/日付時刻*に指定されたフィールドは、フィールド定義で正しく書式を定義してさえいれば、どのような日付書式または日付時刻書式でも使用することができます。

リテラル日付または日付時刻値の指定

*日付/日付時刻*にリテラルの日付値または日付時刻値を指定する場合は、次の表内の書式に制限されます。また、`20141231` のように、値を逆引用符で囲む必要があります。

日付または時刻の個々の要素の間に、スラッシュ(/) やコロン(:) のような区切り文字をいっさい使用しないでください。

- **日付時刻値** - 以下の表に示す日付、区切り文字、および時刻の書式を任意に組み合わせて使用することができます。日付を時刻の前に置く必要があり、かつ、2 つの間に区切り文字を使用する必要があります。有効な区切り文字は空白スペース 1 つ、あるいは文字 't' または 'T' です。
- **時刻値** - 24 時間形式で時刻を指定する必要があります。UTC(Coordinated Universal Time: 協定世界時) からのオフセットは、プラス記号 (+) またはマイナス記号 (-) で始める必要があります。

形式の例	リテラル値の例
YYYYMMDD	`20141231`
YYMMDD	`141231`
YYYYMMDD hhmmss	`20141231 235959`
YYMMDDthhmm	`141231t2359`
YYYYMMDDThh	`20141231T23`
YYYYMMDD hhmmss+/-hhmm (UTC オフセット)	`20141231 235959-0500`
YYMMDD hhmm+/-hh (UTC オフセット)	`141231 2359+01`
メモ UTC オフセットが設定されているデータのメインの時刻書式で hh を単独で使用しないでください。たとえば、「hh+hhmm」という使い方は避けてください。信頼できない結果になる可能性があります。	

ZONED( ) 関数

数値データを文字データに変換し、出力の先頭にゼロを追加します。

構文

```
ZONED(数値, 長さ)
```

パラメーター

名前	種類	説明
数値	数値	文字に変換する数値。 **メモ** 数字の文字列を含む文字値に先頭ゼロを追加する場合は、値を ZONED( ) の入力として使用する前に、VALUE( ) 関数を使用して文字を数値データ型に変換する必要があります。詳細については、"VALUE( ) 関数" ページ 815を参照してください。
長さ	数値	出力文字列の長さ。

出力

文字。

例

基本的な例

整数の入力値

"235" が返されます。

```
ZONED(235, 3)
```

次の例では、*長さ*が*数値*の桁数より大きいので、先頭に2つのゼロが追加されて "00235" が返されます。

```
ZONED(235, 5)
```

次の例では、*長さが数値*の桁数より小さいので、左端の桁が切り詰められて"35"が返されます。

```
ZONED(235, 2)
```

小数の入力値

次の例では、ZONEDデータ型では小数点がサポートされないため、"23585"が返されます。

```
ZONED(235.85, 5)
```

負の入力値

次の例では、数値が負であるのと、右端の桁である4が"M"で表されるのとで、"64489M"が返されます。

```
ZONED(-6448.94, 6)
```

次の例では、*長さが数値*の桁数より小さいので左端の2桁が切り詰められるほか、数値が負であるのと、右端の桁である1が"J"で表されるのとで、"489J"が返されます。

```
ZONED(-6448.91, 4)
```

高度な例

数値を含む文字フィールドに先頭ゼロを追加する

Employee_Numberフィールドには値"254879"があります。値を先頭ゼロ埋めの10桁の文字列に変換する必要があります。

> **ヒント**
>
> VALUE( )関数を使用して文字を数値データに変換してから、ZONED( )への入力値として数値を使用します。

```
COMMENT "0000254879"を返します
ASSIGN v_str_length = 10
ASSIGN v_num_decimals = 0
ZONED(VALUE(Employee_Number, v_num_decimals), v_str_length)
```

テーブルを結合する際にキーフィールドを一致させる

2つのテーブル、**Ar**と**Customer**がある場合に、これらを**CustNo**フィールドで結合してさらに分析する必要があ

るとします。これら 2 つのテーブルにはどちらにも **CustNo** フィールドがありますが、このフィールドのデータ書式が異なっています。

- **Ar テーブルの CustNo フィールド** - 数値フィールド(値の例 : 235)
- **Customer テーブルの CustNo フィールド** - 5 文字の文字フィールド。数値文字フィールドですが、1 ～ 9 でない先頭の桁にはゼロが追加されます("00235" など) 。

結合を行う場合、データ型と長さが等しくなるようにフィールド同士を一致させるには、ZONED( ) 関数を使って **Ar** テーブルの **CustNo** キーフィールドを長さ 5 の文字フィールドに変換することで、その書式を **Customer** テーブルの CustNo キーフィールドの書式に一致させます。

```
OPEN Ar PRIMARY
OPEN Customer SECONDARY
JOIN PKEY ZONED(CustNo,5) FIELDS CustNo Due Amount SKEY CustNo UNMATCHED TO Ar_
Cust OPEN PRESORT SECSORT
```

備考

機能の仕組み

この関数は数値データを文字データに変換し、先頭にゼロを追加して出力します。この関数は、小切手番号や、発注番号、請求書番号フィールドなどの、先頭にゼロを必要とするフィールドを一致させるためによく利用されます。

ZONED( ) の使用に適する場面

この関数は、正の数値を先頭にゼロを付けた文字値に変換する場合に使用できます。この関数は、キーフィールドとして使用されるフィールドのデータを正規化する場合に役立ちます。

たとえば、あるテーブルでは請求書番号は数値フィールドに 100 という形式で格納されており、別のテーブルでは文字フィールドに 00100 という形式で格納されているとします。ZONED 関数を使用して、数値 100 を文字値 00100 に変換できます。これで請求書番号を比較することができます。

文字列の長さと戻り値

*長さ*が*数値*の桁数より大きい場合は、出力値の先頭にゼロが追加されます。*長さ*が*数値*の桁数より小さい場合は、出力の左側から切り詰められます。*数値*と*長さ*が同じ長さの場合、ゼロは追加されません。

10 進数

ZONED データ形式は明示的な小数点を含んでいません。

負の数値

入力値である*数値*が負の場合には、結果では右端の桁が文字として表示されます。

- 右端の桁が0の場合は、文字 "}" として表示
- 右端の桁が1～9の場合は、"J" ～ "R" の間の文字として表示

ZONED( ) とUnicode 版 Analytics

Unicode 版 Analytics を使用している場合、ZONED( ) 関数で返された値を正しく表示するには、BINTOSTR( ) 関数を使用する必要があります。また、ZONED( ) 関数の戻り値を別の関数のパラメーターとして使用する場合も、BINTOSTR( ) 関数を使用する必要があります。

ZSTAT( ) 関数

標準 Z 統計量を返します。

構文

ZSTAT(*実際値*, *期待値*, *母集団*)

パラメーター

名前	種類	説明
実際値	数値	◦ **パラメーターを数値として指定する場合** -は、先頭の桁や先頭の桁の組合せなど、実際の数値を指定します。 ◦ **パラメーターを比率として指定する場合** -は、検査する値の予期される比率は 0 ～ 1 の範囲(0 以上 1 以下) で指定します。
期待値	数値	◦ **パラメーターを数値として指定する場合** -は、先頭の桁や先頭の桁の組合せなど、予期される数値を指定します。 ◦ **パラメーターを比率として指定する場合** -は、検査する値の予期される比率は 0 ～ 1 の範囲(0 より大きく 1 未満) で指定します。
母集団	数値	検査する項目の総数。この値は 0 より大きい正の整数でなければなりません。

出力

数値。

例

高度な例

値が数値のパラメーター

過去 10 年のデータを調べた結果、通常は月ごとの高度障害保険請求件数の分布がほぼ一定であることが判明しました。今年の 4 月、5 月、6 月には請求件数が例年より 10 パーセント多く、例年は 1 か月平均 200 件ですが今年は 220 件ありました。7 月と 8 月の件数はやや少なく、それぞれ 193 件と 197 件でした。

今年の請求件数の合計は2,450件でした。これらの高い数値結果と低い数値結果が有意なものであったかどうかを調べるには、Z統計を使用します。

4月から6月までの実際の請求件数は660件で予想より高いです。この期間の請求件数の期待値は年間の請求件数である2.450件の25パーセント、つまり612.5件です。これらの統計量に対するZ統計量は2.193と計算されます。

```
ZSTAT(660, 612.5, 2450)
```

Z統計量1.96の有意性は0.05で、Z統計量2.57の有意性は0.01です。したがって、請求件数が偶然高くなる確率は1/20と1/100の間です。

7月から8月までの実際の請求件数は390件で予想より低いです。この期間の請求件数の期待値は年間の請求件数である2,450件の1/6、つまり408.33件です。これらの比率に対するZ統計量は0.967と計算されます。

```
ZSTAT(390, 408.33, 2450)
```

これはあまり有意な結果ではありません。1.000未満のZ統計量は非常に一般的で、通常は無視できる範囲の値です。

値が比率のパラメーター

過去10年のデータを調べた結果、通常は月ごとの高度障害保険請求件数の分布がほぼ一定であることが判明しました。今年の4月、5月、6月には請求件数が例年より10パーセント多く、例年は1か月平均200件ですが今年は220件ありました。7月と8月の件数はやや少なく、それぞれ193件と197件でした。今年の請求件数の合計は2,450件でした。これらの高い数値結果と低い数値結果が有意なものであったかどうかを調べるには、Z統計を使用します。

4月から6月までの請求件数の実際値は予想より高く、比率660/2450で表されます。この期間の請求件数の期待値は年間の請求件数である2,450件の25パーセントです。これらの比率に対するZ統計量は2.193です。

```
ZSTAT((1.00000000 * 660 / 2450), 0.25, 2450)
```

Z統計量1.96の有意性は0.05で、Z統計量2.57の有意性は0.01です。したがって、請求件数が偶然高くなる確率は1/20と1/100の間です。

7月から8月までの実際の請求件数は390件と少ないです。この期間の請求件数の期待値は年間の請求件数である2,450件の1/6、つまり16.6667パーセントです。これらの比率に対するZ統計量は0.967です。

```
ZSTAT((1.00000000 * 390 / 2450), 0.16667, 2450)
```

これはあまり有意な結果ではありません。1.000未満のZ統計量は非常に一般的で、通常は無視できる範囲の値です。

備考

機能の仕組み

ZSTAT( ) 関数は、デジタル解析をはじめとするさまざまな問題解決のタスクに使用する標準 Z 統計量を計算します。小数点以下 3 桁の精度で結果を出力します。

ZSTAT( ) を使用する

ZSTAT( ) 関数を使用すると、指定した期間やカテゴリで得られる結果についての発生の頻度を評価することができます。結果の Z 統計量が大きいほど発生の可能性は低くなります。

たとえば、Z 統計量 1.96 の有意性は 0.05(20 回に 1 回発生すると予測される) であるのに対し、Z 統計量 2.57 の有意性は 0.01(100 回に 1 回発生すると予測される) です。Z 統計量については、統計関係の書籍を参照してください。

ZSTAT( ) の入力値を指定する

ZSTAT( ) のパラメーターには数値か比率のいずれかを指定できます。

- 両方の入力値に数値を指定した場合、この関数は浮動小数点演算を使用して Z 統計量を計算します。
- 両方の入力値に比率を指定した場合、この関数は固定小数点演算を使用して Z 統計量を計算します。この場合、小数の乗数を使用して、丸めを制御する必要があります。
- *実際値*または*期待値*を求める式の中で式を使用する場合、小数の乗算によって結果の精度を指定する必要があります。Analytics では小数点以下 8 桁が使用されるので、1.00000000 の乗算によって、得られる範囲で最高の精度の値が返されます。

アナリティクス

アナリティクス スクリプト

スクリプトは Analytics でのみ実行されるように制限されていません。標準のスクリプトは、**アナリティクス スクリプト**に変換すると、HighBond のロボット モジュールまたは Analytics Exchange でスケジュールおよび実行できるようになります。アナリティクス スクリプトは、ACL の独立型コンポーネントである [分析アプリ]ウィンドウで実行します。

アナリティクス スクリプトとは

アナリティクス スクリプトまたは「アナリティクス」は、アナリティクス ヘッダーを持つ標準のスクリプトです。アナリティクス ヘッダーは、ロボットで、AX Server や分析アプリウィンドウでスクリプトの実行を可能にする一連の宣言タグです。アナリティクス ヘッダーには、即時、または予定時刻のいずれかに、無人でのアナリティクス スクリプトの実行を可能にする、ユーザーが事前に設定する入力パラメーターが含まれます。

> **ヒント**
>
> アナリティクス スクリプトは、より容易に開発を行える環境がサポートされる Analytics で開発およびテストされるのが普通です。AX クライアントを使用し、AX Server に保管されている既存のアナリティクス スクリプトに簡単なアップグレードを行います。

分析アプリとは

分析アプリは Analytics プロジェクトであり、Analytics Exchange または分析アプリウィンドウで使用するパッケージです。分析アプリには、1 つまたは複数のアナリティクスのスクリプトや、データ テーブルと解釈も搭載されています。

> **メモ**
>
> 分析アプリは通常、組織の社内スクリプトの専門家によって、または Galvanize コンサルタントとの取り決めによって、作成されるか開発されます。

標準のスクリプトをアナリティクス スクリプトに変換する

アナリティクス スクリプトは標準のスクリプトとして作成を開始します。標準のスクリプトを AX Server のロボット、または分析アプリウィンドウで実行するには、それをアナリティクス スクリプトに変換する必要があります。

1. Analytics でスクリプトを作成してテストします。
2. 該当するアナリティクス ヘッダー タブを追加し、スクリプトをアナリティクス スクリプトにします。
3. アナリティクス スクリプトを AX Server または分析アプリウィンドウで実行できるようにパッケージ化します。ロボットでは、アナリティクス スクリプトを実行できるようにパッケージ化することはできません。

アナリティクス ヘッダーの追加

アナリティクス ヘッダーは、スクリプトの先頭行にあるコメント ブロック内に定義します。スクリプトがアナリティクス スクリプトであることをアナリティクス ヘッダーに宣言することが、最低限必要です。

```
COMMENT
//ANALYTIC 欠落している小切手の識別
このアナリティクス スクリプトは、欠落している小切手番号を検出します。
END
```

詳細については、"アナリティクス ヘッダーの追加" ページ 876を参照してください。

アナリティクス スクリプトの配布と実行

組織で使用される Galvanize 製品およびコンポーネントによっては、アナリティクス スクリプトを配布および実行するには、複数のオプションがあります。

製品/コンポーネント	アナリティクス スクリプトを配布、実行する方法
HighBond のロボット モジュール	◦ ロボットの開発モードで 1 つ以上のアナリティクス スクリプトをバージョンとしてコミットし、本番モードの有効なバージョンをスケジュールおよび実行する
AX Server	これらのモジュールのいずれか: ◦ Analytics プロジェクト(`.acl` ファイル) を直接 AX Server にインポートし、AX Client を使用してアナリティクス スクリプトをスケジュールおよび実行する ◦ プロジェクトを圧縮された分析アプリ ファイル (.aclapp ファイル) にパッケージ化し、AX Server にインポートして、AX Web Client を使用してアナリティクス スクリプトを実行する 詳細については"分析アプリのパッケージ化" ページ 885を参照してください。
分析アプリ ウィンドウ	◦ プロジェクトを圧縮された分析アプリ ファイル (.aclapp ファイル) にパッケージ化し、プロジェクトを分析アプリ(`.aclx` ファイル) として開き、分析アプリ ウィンドウでアナリティクス スクリプトを実行する 詳細については"分析アプリのパッケージ化" ページ 885を参照してください。

アナリティクス スクリプトを実行する環境の判別

スクリプトを、Analytics、Analytics Exchange または分析アプリ ウィンドウで実行できるアナリティクス スクリプトを作成するには、スクリプト実行時にランタイム環境を判別する必要があります。この情報を使用することで、スクリプトが実行される場所に応じてどのコマンドを実行すべきかを決定できます。

スクリプトが実行される場所を判定するには、FTYPE( ) 関数を使用します。

```
FTYPE("ax_main") = "b"
```

スクリプトがAnalytics Exchangeまたは分析アプリウィンドウで実行される場合には、この式はtrue(T) として評価されます。Analyticsで実行されるスクリプトの場合には、この式はfalse(F) として評価されます。詳細については、"FTYPE( ) 関数" ページ 567を参照してください。

AX Serverでスクリプトを実行しているユーザーの識別

AX Serverでアナリティクス スクリプトを実行する場合、システム変数 *AXRunByUser* を使って、同じスクリプトを現在実行しているユーザーの名前(形式 : *<ドメイン>\<ユーザー名>*) を指定することができます。

```
EXTRACT FIELDS TIME() AS "時刻", DATE() AS "日付", AXRunByUser AS "現在のユーザー"
TO R_RunRecord APPEND
```

> **メモ**
>
> *AXRunByUser* は、アナリティクス スクリプトをAX Serverで実行しているときにのみ使用できます。この変数は、Analyticsでスクリプトを実行しているときには認識されません。

アナリティクス ヘッダーおよびアナリティクス タグ

アナリティクス ヘッダーは、Analytics スクリプトの最初にコメント ブロックで囲まれた一連のタグです。アナリティクス タブは、アナリティクスの実行またはスケジュール前にユーザーが入力する入力パラメーターと出力パラメーターを指定します。

アナリティクス ヘッダーは、ロボット、AX Server、または分析アプリ ウィンドウで実行するすべてのアナリティクス スクリプトで必要です。

アナリティクス ヘッダーの定義

アナリティクス ヘッダーは、スクリプトの先頭行から始まるコメント ブロック内に定義する必要があります。次を除き、タグは任意の順序でアナリティクス ヘッダーに配置できます。

- ANALYTIC タグは先頭に配置する必要がある
- FIELD タグは関連付けられた TABLE タグの直後に配置する必要がある

例

このアナリティクス ヘッダーは、スクリプトで使用するテーブルおよびフィールドと開始日パラメーターを特定します。

```
COMMENT
//ANALYTIC Identify missing checks
 このアナリティクスは見つからないチェック番号を特定します。
//TABLE v_table_payments 支払いテーブル
 支払いを一覧表示していて小切手番号列が含まれるテーブルを選択します。
//FIELD v_check_num CN Check Number
 小切手番号が含まれているフィールドを選択します
//PARAM v_start_date D OPTIONAL 開始日 ( 任意 )
 分析の開始日を入力
END
```

タグ形式

ヘッダーの各タグは次の形式を使用します。

```
//タグ名 属性 説明テキスト
```

// タグ インジケーターは、スクリプト行でスペース以外の最初の文字でなければなりません。タグは、タグ インジケーターの直後に、間にスペースも文字も入れずに続ける必要があります。

タグ表記

コンポーネント	表記
タグ名	タグ名は大文字と小文字が区別されません。 Analytics のコマンド名と関数名とは異なり、タグ名は省略できません。
タグ属性	タグの属性を指定するときには、スペースを含め、任意で引用符で値を囲むことができます。
タグ説明	説明は省略可能です。説明を指定する場合には複数行を指定できますが、クライアント アプリケーションでは改行が保持されません。

Analytics でのテスト入力値の指定

特殊割り当て演算子 := を使用し、定義が必要なアナリティクス タグのテスト値を指定できます。

- FILE
- TABLE
- FIELD
- PARAM

次の構文を使用して、Analytics でのアナリティクス スクリプトの実行とテストを行います。

```
//TABLE v_AnalysisTable "分類するテーブル" := "Trans_May"
```

Analytics でスクリプトを実行するときには、パラメーターは割り当てで指定された値を取得します。アナリティクスがクライアント アプリケーションで実行されるときには、テスト値が無視され、ユーザー定義入力パラメーターが使用されます。

割り当て演算子とその前のタグ構文の間にはスペースを残す必要があります。割り当て値は、Analytics 全体で必要なデータ型に適切な修飾子を使用する必要があります。詳細については、"データ型" ページ 21を参照してください。

使用可能なアナリティクス タグの完全な一覧

タグ	説明
"ANALYTIC" ページ 840	Analytics スクリプトをロボット、AX Server または分析アプリ ウィンドウで実行可能なアナリティクスとして指定します。
入力タグ	
"FILE" ページ 842	ロボットまたは AX Server で実行するアナリティクスで使用する入力を提供する、Excel ファイルや区切り文字付きファイルなどの、Analytics ファイル以外のファイルを指定します。

タグ	説明
	◦ ロボットのファイルはアナリティクスと同じロボットの [**入力/出力**] タブにある必要があります ◦ このファイルは、アナリティクスが配置される AX Server 内フォルダーの**関連ファイル** サブフォルダーにある必要があります。
"TABLE" ページ 844	ユーザーがアナリティクスの入力として選択する Analytics テーブルを定義します。 TABLE タグの後には、順次行に入力されたゼロ個以上の FIELD タグを続けることができます。
"FIELD" ページ 846	ユーザーがアナリティクスの入力として選択するフィールドを定義します。 このフィールドは、先行する TABLE タグで定義したテーブルに含まれます。最初の FIELD タグは、TABLE タグの直後に続ける必要があります。その後続けて、追加の FIELD タグを連続した行に入力できます。
"PARAM" ページ 848	アナリティクスのための入力パラメーターを作成し、その入力値の要件を定義します。 入力パラメーターは、ユーザーがアナリティクスを実行またはスケジュールする際に使用する実際の値を指定できるようにするプレースホルダーです。
"PASSWORD" ページ 860	アナリティクスのパスワード入力パラメーターを作成します。パラメーターは、ACLScript コマンドで後から使用するための暗号化されたパスワードの格納場所を提供します。 ユーザーは、アナリティクスをスケジュールまたは開始するときに、必要なパスワード値を指定するように要求されます。これにより、アナリティクスを実行するときにユーザーの介入が不要になります。
出力タグ	
"DATA" ページ 863	アナリティクスによって出力された Analytics テーブルを、デプロイメント環境のデータサブフォルダー(記憶位置) にコピーすることを指定します。 通常は、後続のアナリティクスの入力テーブルとして利用できるよう、Analytics テーブルを格納します。
"RESULT" ページ 867	エンド ユーザーからクライアント アプリケーションでアクセスできるようにするアナリティクス出力結果を指定します。 出力結果は、存在していても、自動的に利用可能にはなりません。利用可能にしたい結果項目ごとに、RESULT タグを使用する必要があります。
"PUBLISH" ページ 871	アナリティクスが処理を完了したときに、どの Analytics テーブルを AX Exception に公開するかを定義するメタデータを含んでいるファイルを指定します。

ANALYTIC

Analytics スクリプトをロボット、AX Server または分析アプリウィンドウで実行可能なアナリティクスとして指定します。

構文

```
//ANALYTIC <TYPE IMPORT|PREPARE|ANALYSIS> 名前
 <説明>
```

パラメーター

名前	説明
TYPE 省略可能	アナリティクス スクリプトの種類を次の 3 つのタイプのいずれかとして指定します。 ◦ インポート- データ ソースからデータを取得します。インポート アナリティクスの出力は、生のデータ テーブルです。 ◦ 準備- 生データを分析に適したものにするために必要な方法で生データを変換します。準備アナリティクスの出力は、分析テーブルです。 ◦ 分析- 分析テーブルのデータでテストを実行します。分析アナリティクスの出力は、1 つまたは複数の結果テーブルです。 指定されたタイプのアナリティクスは、ロボット、AX Web Client および分析アプリウィンドウの対応する [**インポート**]、 [**準備**]、または [**分析**] の領域に配置されます。この配置は、ユーザーが適切な順序でアナリティクスを実行するための指針となります。この順序は省略可能です。また、アナリティクスの機能のタイプも省略可能です。 TYPE を省略した場合は、アナリティクスが [**分析**] セクションに表示されます。
名前	アナリティクスの名前。 この名前は、クライアント アプリケーションにおいてアナリティクスを識別するものです。このアナリティクス名は、スクリプトを初めに作成したときに、Analytics で指定したスクリプト名とは別のものです。 **メモ** 同じ名前のプロジェクトまたは分析アプリのアナリティクスの名前は一意である必要があります。複数のアナリティクスで同じ *名前* が使用されていると、アナリティクス スクリプトをコミットしようとするか、その分析アプリをインポートするか開こうとしたときにエラーが発生します。 Windows のファイル名の中で使用できない文字 (**< > : " / \ | ? ***) はアナリティクス名の中で使用しないでください。使用すると、エラーが発生し、アナリティクスの結果をエクスポートできなくなります。名前として TYPE という値を使用しないでください。 クライアント アプリケーションでは、名前は英数字順にリストされます。単一のロボットまたは分析アプリで、複数のアナリティクスを正しい順序で実行するようユーザーを導くには、各領域内のアナリティクス名を順序付けるためのプレフィックスを追加します。たとえば、01_analyze_

名前	説明
	POs、02_analyze_invoices などのようにします。この場合、順序付けは、名前の順序で暗黙に設定されるので、省略可能です。
説明 省略可能	アナリティクスの説明や、アナリティクスを正常に実行するために必要と思われるその他の情報の説明。 この説明は、クライアント アプリケーション内にアナリティクスと一緒に表示されます。説明を複数行で指定することはできますが、行を飛ばすことはできません。説明は、関連する ANALYTIC タグの下の行に入力する必要があります。

例

基本的なアナリティクス ヘッダー

次のアナリティクス ヘッダーには、アナリティクスの名前と説明が表示されています。

```
COMMENT
//ANALYTIC 欠落している小切手の識別
 この分析は、欠落している小切手番号を識別します。
END
```

アナリティクス ヘッダーおよびアナリティクス タイプ

次のアナリティクス ヘッダーは、準備アナリティクスと、スクリプトの機能の説明を指定しています。

```
COMMENT
//ANALYTIC TYPE PREPARE 住所データの標準化
  このアナリティクスは重複分析を準備するために住所フィールドをクリーンアップして標準化します。
END
```

備考

ACLScript の COMMENT コマンドはスクリプトの 1 行目に入力し、その後 2 行目に ANALYTIC タグを続ける必要があります。ほかの場所で ANALYTIC タグが使用されている場合、そのタグは無視されます。

Analytics プロジェクト内の 1 つ以上のスクリプトに ANALYTIC タグを含めることができます。

FILE

ロボットまたはAX Serverで実行するアナリティクスで使用する入力を提供する、Excelファイルや区切り文字付きファイルなどの、Analyticsファイル以外のファイルを指定します。

- ロボットのファイルはアナリティクスと同じロボットの［**入力/出力**］タブにある必要があります
- このファイルは、アナリティクスが配置されるAX Server内フォルダーの**関連ファイル**サブフォルダーにある必要があります。

> **メモ**
>
> 分析アプリウィンドウで実行されるアナリティクスの非Analytics入力ファイルを指定するには、"PARAM" ページ 848を参照してください。

構文

```
//FILE ファイル名
```

パラメーター

名前	説明
ファイル名	アナリティクスの入力ファイルとして使用する、ロボットまたは**関連ファイル**サブフォルダーにある項目の名前。*ファイル名*にパスを含めることはできません。 ファイル名を指定する際には、ワイルドカードを使用できます。ゼロ個以上の文字の代わりに単一のアスタリスク(*) を使用します。 例： ◦ Inv12* は、`Inv12`、`Inv123`、`Inv1234` のすべてと一致します。 ◦ *.* は、ロボットまたは**関連ファイル**フォルダー内のあらゆる拡張子の全ファイルと一致します。 ◦ Inv_*.* は `Inv_Jan.pdf` および `Inv_Feb.xls` と一致します。 **ヒント** //FILEタグを使ってAnalytics初期設定ファイル(`.prf`) を参照できます。これを行うと、AX Server上のグローバル初期設定ファイルでなく**関連ファイル**サブフォルダーの初期設定ファイルによって、ランタイム環境の設定が構成されます。初期設定ファイルは、Analytics Exchangeのインストールと互換性があるAnalyticsの最新バージョンの初期設定ファイルである必要があります。

例

基本的な例

特定のファイルを指定します。

```
//FILE FlaggedAccounts.csv
```

"Flagged" で始まるすべての CSV ファイルを指定します。

```
//FILE Flagged*.csv
```

すべてのファイルを指定します。

```
//FILE *.*
```

高度な例

取り込んだファイルからのデータのインポート

AX Server で従業員データの月次分析を実行します。分析アプリのアナリティクスの 1 つにより、関連ファイル フォルダーに毎月格納される区切り文字付きファイルから分析対象データがインポートされます。

```
COMMENT
//ANALYTIC TYPE IMPORT employee_import
  関連ファイル フォルダーに保存された区切りファイルから従業員レコードをインポートします。
//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"
```

備考

FILE タグは、分析アプリウィンドウで実行されるアナリティクスでは使用できません。分析アプリウィンドウで実行されるアナリティクスの入力ファイルを指定するには、PARAM タグを使用します。詳細については、"PARAM" ページ 848を参照してください。

TABLE

ユーザーがアナリティクスの入力として選択するAnalyticsテーブルを定義します。

TABLEタグの後には、順次行に入力されたゼロ個以上のFIELDタグを続けることができます。

> **メモ**
>
> TABLEタグは、選択できるようにするために、テーブルが格納場所に予め存在することを要求します。詳細については、"DATA" ページ 863を参照してください。

構文

```
//TABLE id 名前
 <説明>
```

パラメーター

名前	説明
id	ユーザーによって選択された入力テーブル名を格納する変数。そのテーブルを参照するこの変数の値をアナリティクス スクリプト内で使用します。
名前	テーブルに関連付ける名前。 この値は、クライアント アプリケーション内で、アナリティクスを実行するユーザーがテーブルの選択を求められるときに表示されます。
説明 省略可能	テーブルの目的を明記する説明テキスト。説明を複数行で指定することはできますが、行を飛ばすことはできません。 この値は、クライアント アプリケーション内でユーザーがテーブルの選択を求められるときに表示されます。この説明により、ユーザーに正しいテーブルを選択するよう促すことができます。たとえば、"給与情報を含んでいるテーブルを選択してください" などと指定します。 説明は、関連するTABLEタグの下の行に入力する必要があります。

例

基本的な例

ユーザーが正しい入力テーブルを選択できるようにするための説明があるTABLEタグ:


```
//TABLE v_table_payments 支払テーブル
 支払が記載されており、小切手番号の列があるテーブルを選択します。
```

高度な例

TABLE タグで定義したテーブルのスクリプトでの使用

次のスクリプトは、プロジェクト内のデータテーブルからユーザーが選択したテーブルに対して AGE コマンドを実行します。

```
COMMENT
//ANALYTIC スクリプト例
//TABLE v_table_payments 支払いテーブル
 支払いを一覧表示していて小切手番号列が含まれるテーブルを選択します。
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

ユーザーがアナリティクスの入力として選択するフィールドを定義します。

このフィールドは、先行するTABLEタグで定義したテーブルに含まれます。最初のFIELDタグは、TABLEタグの直後に続ける必要があります。その後続けて、追加のFIELDタグを連続した行に入力できます。

> **メモ**
>
> TABLEタグは、選択できるようにするために、テーブルが格納場所に予め存在することを要求します。詳細については、"DATA" ページ 863を参照してください。

構文

```
//FIELD id型 名前
 <説明>
```

パラメーター

名前	説明
id	ユーザーによって選択された入力フィールド名を格納する変数。そのフィールドを参照するこの変数の値をアナリティクス スクリプト内で使用します。
種類	選択できるフィールドの型。次の一覧から、任意の型または型の組み合わせを選択できます。 ◦ C - 文字データ ◦ N - 数値データ ◦ D - 日付時刻データの日付、日付時刻、または時刻サブタイプ ◦ L - 論理データ テーブル内の演算フィールドは、指定された*型*に関係なく選択できます。
名前	フィールドに関連付ける名前。 この値は、クライアント アプリケーション内でユーザーがフィールドの選択を求められるときに表示されます。
説明 省略可能	フィールドの目的を明記する説明テキスト。説明を複数行で指定することはできますが、行を飛ばすことはできません。 この値は、クライアント アプリケーション内でユーザーがフィールドの選択を求められるときに表示されます。この説明により、ユーザーに正しいフィールドを選択するよう促すことができます。たとえば、"支払金額を含んでいるフィールドを選択してください" などと指定します。 説明は、関連するFIELDタグの下の行に入力する必要があります。

例

基本的な例

文字フィールドを指定します。

```
//FIELD v_name C Name Field
```

文字または数値フィールドを指定します。

```
//FIELD v_inv_num CN Invoice Number
```

高度な例

2 つの FIELD タグを指定した TABLE

次のアナリティクス ヘッダーは、スクリプトが実行される際にユーザーが *v_table_payments* テーブル内の 2 つの入力フィールドを指定できるようにしています。

```
COMMENT
//ANALYTIC テスト アナリティクス
//TABLE v_table_payments 支払いテーブル
  支払いを一覧表示していて小切手番号列が含まれるテーブルを選択します。
//FIELD v_check_num CN 小切手番号フィールド
//FIELD v_payment_date D 支払い日フィールド
   小切手支払い日を含む列を選択します。
END

OPEN %v_table_payments%
EXTRACT FIELDS %v_check_num%, %v_payment_date% TO t_analyze
```

PARAM

アナリティクスのための入力パラメーターを作成し、その入力値の要件を定義します。

入力パラメーターは、ユーザーがアナリティクスを実行またはスケジュールする際に使用する実際の値を指定できるようにするプレースホルダーです。

構文

```
//PARAM id 型 <OPTIONAL> <MULTI> <SEPARATOR 値> <QUALIFIER 値> <VALUES 値リスト>
ラベル
 <説明>
```

パラメーター

名前	説明
id	ユーザーによって選択または指定されたアナリティクスの入力値を格納する変数。 例： ◦ v_start_date ◦ v_regions ◦ v_input_file パラメーターの一意の識別子にもなります。 **メモ** アナリティクスが実行されたとき、ユーザーが入力値を提供する場合にのみ、この変数は作成されます。パラメーターが省略可能で、ユーザーが入力をスキップした場合、変数は作成されません。 アナリティクスの以降のロジックで変数の存在が必要となる場合は、変数の存在をテストすることができるので、存在しなければ、変数を作成して初期化します。詳細については、"省略可能な入力パラメーターの設計" ページ 854を参照してください。
種類	パラメーターのデータ型。これにより、入力できる入力値の種類を制御します。 次の型を、大文字を使用して指定することができます。 ◦ C - 文字データ ◦ N - 数値データ ◦ D - 日付時刻データの日付サブタイプ ◦ DT- 日付時刻データの日付時刻サブタイプ ◦ T - 日付時刻データの時刻サブタイプ ◦ L - 論理データ

名前	説明
	メモ アナリティクスが正常に実行されるようにするには、文字の入力値を修飾することが必要です。 **PARAM... F の仕組み** また、ファイル アップロード ユーティリティまたは Windows のファイル ブラウザーが開くように指定することもできます。 ◦ F- ファイル アップロード ユーティリティまたは Windows ファイル ブラウザーを開き、ユーザーが AX Web Client または分析アプリ ウィンドウでアナリティクスを実行する際に入力ファイルとして Analytics 以外のファイルを選択できるようにします。 選択する際、文字入力値としてファイル名が自動的に入力されます。F のみを指定します。F C は指定しないでください。 例： `//PARAM v_input_file F...` 詳細については、"アナリティクスで Analytics 以外の入力ファイルを指定または選択する" ページ 858を参照してください。 **メモ** *型* F は、ロボットまたは AX Client で実行されるアナリティクスでは使用できません。これらの環境で実行されるアナリティクスの入力ファイルを指定するには、FILE タグを使用します。詳細については、"FILE" ページ 842を参照してください。
OPTIONAL 省略可能	パラメーターは任意であり、ユーザーは値を入力しなくてもいいことを示します。 詳細については、"省略可能な入力パラメーターの設計" ページ 854を参照してください。
MULTI 省略可能	パラメーターは複数の入力値を受け入れることを示します。 MULTI は、VALUES オプションの有無にかかわらず使用できます。 MULTI ✔ VALUES ✔ — ユーザーは値のリストから 1 つ以上の値を選択できます。 MULTI ✔ VALUES ✘ — ユーザーは 1 つ以上の値を手動で入力できます。 詳細については、"MULTI および VALUES オプションの要約" ページ 855を参照してください。 MULTI は、*型* が L(ローカル) の場合には使用できません。 **複数文字入力値** MULTI を指定し、*型* が C(文字) である場合は、入力値の文字列に区切り文字とテキスト修飾子が自動的に挿入されるよう、SEPARATOR と QUALIFIER オプションを指定することもできます。

名前	説明
	メモ アナリティクスが正常に実行されるようにするには、複数の文字の入力値を区切ったり修飾したりすることが必要です。区切り文字および修飾子は自動的に挿入するか、ユーザーが手動で挿入できます。
SEPARATOR *値* 省略可能	SEPARATOR は、*MULTI* が指定され、型が C(文字) も場合にのみ使用できます。 処理するアナリティクスに渡される区切り文字付きの一覧を作成するときに、複数の文字入力値の間に区切り文字を自動的に挿入することを指定します。 *値*には、使用する区切り文字を指定します。よく使われる分離記号、すなわち区切り文字はカンマ,です。 SEPARATOR を省略した場合は、デフォルトで、1 つのスペースが区切り文字として使用されます。スペース文字を*値*として指定することはできません。 詳細については、"文字入力値の区切りと修飾" ページ 856を参照してください。
QUALIFIER *値* 省略可能	QUALIFIER は、*MULTI* が指定され、型が C(文字) も場合にのみ使用できます。 処理するアナリティクスに渡される区切り文字付きの一覧を作成するときに、各文字入力値の始まりと終わりにテキスト修飾子の文字を自動的に挿入することを指定します。修飾子文字内に囲まれたあらゆるテキストが、プレーン テキストとして扱われます。 *値*には、使用する修飾子文字を指定します。よく使われる修飾子は一重引用符'です。 QUALIFIER を省略した場合に使用されるデフォルトの修飾子はありません。スペース文字を*値*として指定することはできません。 詳細については、"文字入力値の区切りと修飾" ページ 856を参照してください。 **メモ** アナリティクスの入力パラメーターは現在のところ、テキスト修飾子として二重引用符(") の使用をサポートしていません。代わりに、一重引用符(') を使用することができます。二重引用符を指定すると、PARAM タグが正しく動作しなくなります。
VALUES *値リスト* 省略可能	ユーザーがアナリティクスを実行するときに選択できる値のリスト。 次の構文を使用して、値を指定します。 `VALUES |値 1|値 2|値 3|値 n|` VALUES は MULTI オプションの有無にかかわらず使用できます。 VALUES ✔ MULTI ✔ : ユーザーは値のリストから 1 つ以上の値を選択できます。 VALUES ✔ MULTI ✘ : ユーザーは値のリストから単一の値を選択できます。 詳細については、"MULTI および VALUES オプションの要約" ページ 855を参照してください。 ***値リストの値の形式***

名前	説明
	<table><tr><td>文字値</td><td>◦ スペースと句読点を含めることができます</td></tr><tr><td>数値</td><td>◦ 正数でも負数でも指定できます。 ◦ 10進表記法を用いて、桁区切り記号を付けないで指定する必要があります。 例：1500.00 または -1500.00</td></tr><tr><td>日付時刻値</td><td>◦ **日付** -は、書式 MM/DD/YYYY を使用して指定する必要があります (例：12/31/2014) ◦ **日付時刻** -は、書式 MM/DD/YYYY hh:mm:ss を使用して指定する必要があります (例：12/31/2014 23:59:59) ◦ **時刻** -は、書式 hh:mm:ss を使用して指定する必要があります(例：23:59:59)</td></tr><tr><td>論理値</td><td>VALUE は、*型*が L(ローカル) の場合には使用できません。</td></tr></table>
ラベル	パラメーターのユーザー インターフェイスのラベル。 クライアント アプリケーションで、*ラベル*は入力フィールドで表示されます。
説明 省略可能	パラメーターに関する詳細情報を提供する説明テキスト。 クライアント アプリケーションで、*説明*は入力フィールドで表示されます。 *説明*はユーザーを支援する手順を提供できます。たとえば、"給与支払期間の締切日を入力してください" などと指定します。 *説明*は、関連する PARAM タグより後の次の行に入力する必要があります。説明を複数行で指定することはできますが、行を飛ばすことはできません。説明をクライアント アプリケーションで表示する場合には、改行は保持されません。

例

基本的な例

ユーザーが任意で日付の範囲を指定できるようにする

```
//PARAM v_start_date D OPTIONAL 開始日 ( 任意 )
  分析の開始日を入力
//PARAM v_end_date D OPTIONAL 終了日 ( 任意 )
  分析の終了日を入力します
```

ユーザーに処理する取引の最大数を選択するように要求する

```
//PARAM v_maxTrans N VALUES |250|500|750|1000| 処理する最大取引数
```

ユーザーに商業カテゴリコードを 1 つ以上指定するように要求する

```
//PARAM v_codes C MULTI SEPARATOR , QUALIFIER ' 含める MC コード
 商業カテゴリコードを 1 つ以上指定します。コードの後に Enter を押して、各コードが新しい行に移動するようにします。コードを引用符で囲まないでください。
```

ユーザーに商業カテゴリコードを 1 つ以上選択するように要求する

```
//PARAM v_codes C MULTI SEPARATOR , QUALIFIER ' VALUES |4121 タクシー/リムジン|5812 レストラン|5813 飲み屋 - アルコール飲料|5814 ファスト フード レストラン| 含める MC コード
 商業カテゴリコードを 1 つ以上選択します。
```

高度な例

ユーザーに金額の範囲を指定するように要求する

最低金額～最高金額の範囲内に含まれる、テーブル内のレコードを分類化する必要があります。この範囲は変更される場合があるため、スクリプト作成者は、アナリティクスを実行するユーザーがスクリプトをスケジュールまたは実行する際にこの範囲を定義できるようにする入力パラメーターを提供します。

```
COMMENT
//ANALYTIC テスト アナリティクス
//PARAM v_min_amount N 最低額
 最低額を入力します
//PARAM v_max_amount N 最高額
 最高額を入力します
END

CLASSIFY ON %v_FieldA% IF BETWEEN(AMOUNT, v_min_amount, v_max_amount) SUBTOTAL AMOUNT TO "Classified_%v_AnalysisTable%.FIL"
```

ユーザーが任意で 1 つ以上の顧客番号を除外できるようにする

テーブル内のレコードを分類化する必要がある場合に、ユーザーに分析から特定の顧客を除外できるようにさせたいとします。

これを行うには、スクリプト作成者はオプションの文字パラメーターを提供します。作成するスクリプトでは、顧客番号の値が指定されたかどうかをテストし、指定されている場合には、指定された顧客番号を分類化コマンドの対象から除外します。

```
COMMENT
//ANALYTIC テスト アナリティクス
//PARAM v_cust_no C OPTIONAL MULTI SEPARATOR , QUALIFIER ' 除外する顧客番号 ( 省略可能 )
 1つ以上の顧客番号を指定します。番号の指定後にEnterを押して、各番号を新しい行に移動します。番号を引用符で囲まないでください。
END

IF FTYPE("v_cust_no") = "U" v_cust_no = ""
GROUP IF v_cust_no = ""
 CLASSIFY ON %v_FieldA% SUBTOTAL AMOUNT TO "Classified_%v_AnalysisTable%.FIL"
ELSE
 CLASSIFY ON %v_FieldA% IF NOT MATCH(CUSTNO, %v_cust_no%) SUBTOTAL AMOUNT TO "Classified_%v_AnalysisTable%.FIL"
END
```

ユーザーが入力ファイルを選択できるようにする(AX Web Client または分析アプリウィンドウのみ)

分析アプリを、それを分析アプリウィンドウで実行する同僚に配布するとします。スクリプト作成者は、同僚がアプリでアナリティクス スクリプトを実行する際に、データのインポート元となる Microsoft Excel ファイルを選択できるよう、Windows ファイル エクスプローラーを提供する必要があります。

```
COMMENT
//ANALYTIC テスト アナリティクス
//PARAM v_input_file F
 入力ファイル 入力ファイルを選択します
END

IMPORT EXCEL TO Trans_May_raw Trans_May_raw.fil FROM "%v_input_file%" TABLE "Trans2_May$" CHARMAX 100 KEEPTITLE
```

ユーザーに入力ファイルのパスと名前を指定するように要求する(分析アプリウィンドウのみ)

分析アプリを、それを分析アプリウィンドウで実行する同僚に配布するとします。スクリプト作成者は、同僚がアプリでアナリティクス スクリプトを実行する際に、インポート ファイルのファイル パスおよびファイル名を指定するようにしなければなりません。

```
COMMENT
//ANALYTIC テスト アナリティクス
//PARAM v_input_file C 入力ファイル パスと名前
 絶対ファイル パスとファイル名を入力します 例 : C:\Users\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
```

オプションのパラメーターへのデフォルト値の使用

取引レコードを結果テーブルに抽出するアナリティクスを作成するとします。スクリプトを実行するユーザーに対し、日付範囲を提供するだけでなく、抽出するレコードをフィルターリングするためのエンティティの一覧をも提供するオプションを用意する必要があります。

そのようにするには、以下の3つのオプション パラメーターを作成します。

- v_start_date
- v_end_date
- v_entity_list

スクリプトの冒頭の数行で、これらの値が設定されているかどうかをテストします。設定されていない場合は、最早日付および最遅日付のデフォルト値と、v_entity_list に対してテストするというデフォルト フラグを設定します。

EXTRACT コマンドでは、レコードをフィルターリングする値を使用します。

```
COMMENT
//ANALYTIC test
 このアナリティクスは PARAM をテストします
//RESULT TABLE t_results
//PARAM v_start_date D OPTIONAL 開始日を入力
//PARAM v_end_date D OPTIONAL 終了日を入力
//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'")
```

備考

省略可能な入力パラメーターの設計

PARAM タグで OPTIONAL を使用した場合、そのアナリティクスの入力パラメーターに関連付けられた変数は、アナリティクスの実行時に作成されるかどうかわかりません。

- ユーザーが入力値を指定した場合は、**変数が自動的に作成**-されます
- ユーザーが省略可能なパラメーターをスキップし、入力値を指定しなかった場合には、**変数は作成されません**-

パラメーター変数の存在のテスト

アナリティクスの以降のロジックが、パラメーター変数の内容を評価できるものとしている場合は、それが空やNULLの状態を評価するものであっても、パラメーター変数の存在をテストする必要があります。パラメーター変数が存在しない場合は、作成して、ヌルに初期化する必要があります。

IFコマンドとFTYPE( )関数またはISDEFINED( )関数を使用し、テストを実行して、存在しない場合は変数を作成します。

```
IF FTYPE("変数名") = "U" 変数名 = ""
```

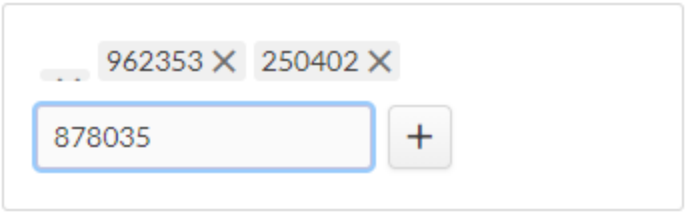
```
IF NOT ISDEFINED("var_name") var_name = ""
```

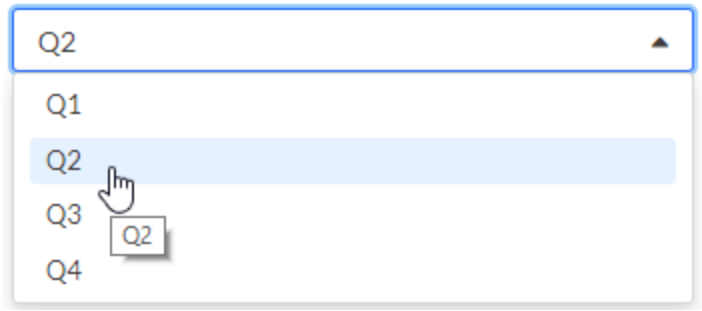
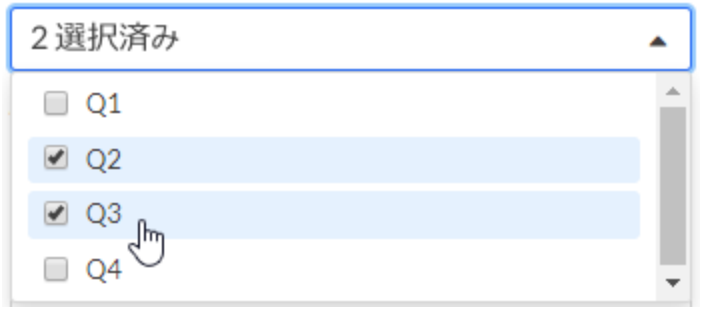
テストを実行するとき

テストは、アナリティクス ヘッダーより後、パラメーター変数の存在に左右されるAnalytics Scriptのロジックより前に実行します。

MULTIおよびVALUESオプションの要約

以下の表は、ユーザー インターフェイスのユーザー入力コントロールに対する、MULTIおよびVALUEオプションの効果の要約です。

ユーザー入力コントロール(ロボット)	パラメーター設計	MULTI	VALUES
	フィールドに単一の入力値を手動で入力する	✗	✗
	フィールドに1つ以上の入力値を手動で入力する	✓	✗

ユーザー入力コントロール(ロボット)	パラメーター設計	MULTI	VALUES
	ドロップダウン リストの値から単一の入力値を選択する	✗	✓
	チェックリストの値から 1 つ以上の入力値を選択する	✓	✓

文字入力値の区切りと修飾

文字入力値が複数存在する場合は、値を区切り文字で区切る必要があります。また、アナリティクスを正常に実行させるために、値を修飾する必要があります。

入れ子になったテキスト修飾子を使用しない

文字の入力パラメーターを作成するとき、またアナリティクスを実行するユーザーに文字入力値の入力方法を指示するときには、余分なテキスト修飾子、つまり入れ子になったテキスト修飾子(修飾子の中の修飾子) を作成しないように注意する必要があります。余分なテキスト修飾子により、入力パラメーターが正しく動作しなくなります。

テキスト修飾子を挿入する方法

文字入力値の前後にテキスト修飾子を挿入するための方法は 4 種類あります。方法によっては、入力値の間に区切り文字も挿入されます。

アナリティクスを開発するときには、別の方法で実験し、ユーザーが入力する文字値に適切なものを見つけなければならない場合があります。

> **メモ**
>
> MULTI オプションとVALUES オプションをどのように使用しているかによって、1 つまたは複数の方法を適用できない場合があります。
>
> 各入力パラメーターはこれらの方法の**1 つのみ**を使用する必要があります。

1	SEPARATOR および QUALIFIER を使用する	PARAM タグに SEPARATOR および QUALIFIER オプションを含めます。 例： `//PARAM v_regions C MULTI SEPARATOR , QUALIFIER '` MULTI なしで VALUES を使用する場合は適用されません。 **ヒント** 可能な限り、この方法を使用します。これは、必要とする労力が最も少なく、また、エラーを起こす度合いが最も低いです。
2	区切り文字と修飾子を手動で指定する	アナリティクスのユーザーに、実際の入力値に加えて区切り文字と修飾子を手動で指定するように要求します。 例： `'North America','Europe','Asia'` MULTI の有無にかかわらず、VALUES を使用する場合は適用されません。
3	値リストに修飾子を含める	VALUES オプションと一緒に指定される*値リスト*の各値に修飾子を含めます。 例： `VALUES |'Asia'|'Europe'|'Middle East'|'North America'|` VALUES なしで MULTI を使用する場合は適用されません。
4	パラメーター変数を修飾子で囲む	Analytics スクリプトの構文で、パラメーター変数をテキスト修飾子で囲みます。 例： `IF MATCH(REGIONS, "%v_regions%")` この方法は、MULTI なしで VALUES を使用している場合にのみ使用します。
	メモ アナリティクスの入力パラメーターは現在のところ、テキスト修飾子として二重引用符(") の使用をサポートしていません。その代わりに、*値リスト*で、あるいは入力値の前後に手動で修飾子を指定するときは、QUALIFIER オプションによって一重引用符(') を使用できます。二重引用符は、Analytics スクリプトの本文ではテキスト修飾子として使用できます。	

別の方法を使用するとき

以下の表には、テキスト修飾子を挿入するさまざまな方法をどのような場合に使用するかまとめてあります。

	MULTI ✔ VALUES ✘	MULTI ✘ VALUES ✔	MULTI ✔ VALUES ✔
方法 1	使用した場合は、方法 2	該当なし	使用した場合は、方法 3

	MULTI ✔ VALUES ✘	MULTI ✘ VALUES ✔	MULTI ✔ VALUES ✔
SEPARATOR および QUALIFIER オプションを使用する	を使用しない		を使用しない
方法 2 区切り文字と修飾子を手動で指定する	使用した場合は、方法 1 を使用しない	該当なし	該当なし
方法 3 *値リスト*に修飾子を含める	該当なし	使用した場合は、方法 4 を使用しない	使用した場合は、方法 1 を使用しない
方法 4 パラメーター変数を修飾子で囲む	使用しない	使用した場合は、方法 3 を使用しない	使用しない

アナリティクスで Analytics 以外の入力ファイルを指定または選択する

以下の表では、Analytics 以外の入力ファイルを指定または選択するためのさまざまな方法を説明しています。選択する方法は、アナリティクスを実行するために使用されるクライアント アプリケーションによって多少異なります。

方法	詳細	ロボット	AX Client	AX Web Client	分析アプリ ウィンドウ
PARAM タグと 'F' データ型	◦ AX Web Client ユーザーはファイル アップロード ユーティリティを使用して、入力ファイルを選択します ファイル名は自動的にアナリティクス入力値として指定 ファイルが AX Server 上の該当する**関連ファイル** サブフォルダーにアップロードされます。 ◦ 分析アプリ ウィンドウ ユーザーは、Windows ファイル ブラウザーを使用して、入力ファイルを選択します ファイル パスとファイル名は自動的にアナリティクス入力値として指定されます。 この方法は、柔軟性、使いやすさ、精度を兼ね備えていることから、最適のオプションと言えるでしょう。			✔	✔
PARAM タグと 'C' データ型	ユーザーは、入力ファイル パスとファイル名をアナリティクス入力値として手動で指定します。 この方法では、ファイル パスとファイル名が事前に指定されていないので、柔軟性があります。ただしこれは、ユーザーにこれらの値を手動入力するように要求する				✔

方法	詳細	ロボット	AX Client	AX Web Client	分析アプリウィンドウ
	ので、手間がかかりエラーが発生しやすくなります。				
FILE タグ (詳細については、"FILE" ページ 842を参照します)	◦ ロボットの入力ファイルはロボットの [**入力/出力**] タブにある必要があります ◦ **AX Client**、**AX Web Client** - 入力ファイルは、AX Server 上の該当する**関連ファイル** サブフォルダーになければなりません。	✔	✔	✔	
アナリティクスでハードコードされている入力ファイルパスとファイル名	この方法は、PARAM タグの使用を回避しますが、あまり柔軟性がありません。ユーザーはアナリティクスが実行されるコンピューターごとに、入力ファイルのファイルパスとファイル名がアナリティクスに指定されたものと同一であるかどうかを確認する必要があります。				✔

PASSWORD

アナリティクスのパスワード入力パラメーターを作成します。パラメーターは、ACLScriptコマンドで後から使用するための暗号化されたパスワードの格納場所を提供します。

ユーザーは、アナリティクスをスケジュールまたは開始するときに、必要なパスワード値を指定するように要求されます。これにより、アナリティクスを実行するときにユーザーの介入が不要になります。

構文

```
//PASSWORD インデックス 名前
 <説明>
```

パラメーター

名前	説明
インデックス	パスワードに関連付ける数値識別子。値は1から10までのいずれかでなければなりません。
名前	パスワード プロンプトのラベル。*名前*は、アプリケーションがユーザーにパスワードの値の入力を求めるときに、クライアント アプリケーションで表示されます。
説明 省略可能	必要なパスワードやその他の情報に関する説明テキスト。 説明を複数行で指定することはできますが、行を飛ばすことはできません。説明は、関連するPASSWORDタグの下の行に入力する必要があります。

例

Direct Link SAP クエリのパスワード入力パラメーターを作成する

アナリティクス ヘッダーは、SAP パスワードを入力するようにユーザーに指示するパスワード入力パラメーターを指定します。保存されたパスワードは、スクリプトの本文の後続のRETRIEVEコマンドで使用されます。

```
COMMENT
//ANALYTIC SAP パスワードの例
//PASSWORD 1 SAP パスワード:
//DATA RSADMIN
END
SET SAFETY OFF
```

```
RETRIEVE RSADMIN PASSWORD 1
OPEN RSADMIN
SET SAFETY ON
```

> **メモ**
>
> RETRIEVE コマンドのパスワード入力パラメーターとパスワード パラメーターは、同じ数値 ID を使用してリンクされます。
>
> ```
> //PASSWORD 1 SAP パスワード:
> .
> .
> .
> RETRIEVE RSADMIN PASSWORD 1
> ```

リザルトにエクスポートするためのパスワード入力パラメーターを作成する

アナリティクス ヘッダーは、HighBond パスワードを入力するようにユーザーに指示するパスワード入力パラメーターを指定します。保存されたパスワードは、スクリプトの本文の後続の EXPORT コマンドで使用されます。

```
COMMENT
//ANALYTIC HighBond パスワードの例
//PASSWORD 3 HighBond パスワード:
END
SET SAFETY OFF
OPEN AR_Exceptions
EXPORT FIELDS No Due Date Ref Amount Type ACLGRC PASSWORD 3 TO "10926@us"
SET SAFETY ON
```

備考

パスワードの格納と暗号化

パスワードの値は、個々のユーザーと関連付けられ、格納時に暗号化されます。パスワードは、アナリティクス処理を通じて安全性を確保し続け、展開環境で作成されるすべての一時ファイルで暗号化されます。

Analytics でのテスト

Analytics 内に 1 つ以上の PASSWORD タグを持つアナリティクスをテストすると、Analytics により、PASSWORD コマンドが自動的に生成され、適切なパスワードを入力するように求められます。この自動生成されたコマンドにより、テスト目的でアナリティクスのスクリプト部で PASSWORD コマンドを挿入し、アナリティクスをユーザーに提供する前にそれを削除する必要がありません。

自動生成された PASSWORD コマンドは、パスワード値なしでログに保存されます。

パスワード値は、Analytics でアナリティクスを実行するときに保存されるため、カーソル位置からのアナリティクスの実行またはステップなど、アナリティクスを実行するたびにパスワードを指定する必要があります。

DATA

アナリティクスによって出力されたAnalyticsテーブルを、デプロイメント環境のデータサブフォルダー(記憶位置)にコピーすることを指定します。

通常は、後続のアナリティクスの入力テーブルとして利用できるよう、Analyticsテーブルを格納します。

> **メモ**
>
> Professional EditionのACL Roboticsには、Analyticsテーブルの格納場所は含まれていません。//DATAタグは、Professional Editionで実行されるアナリティクス実行では無視されます。
>
> 記憶容量が必要な場合は、Enterprise Editionにアップグレードが可能です。

構文

```
//DATA 名前
```

パラメーター

名前	説明
名前	格納するAnalyticsテーブルの名前。*名前*値にスペースを含めることはできません。 **メモ** *名前*値はアナリティクス スクリプトのAnalytics出力テーブルの名前と正確に一致する必要があります。*名前*でテーブル名を指定していません。スクリプトで指定された名前と一致しています。 ソース データ ファイルの名前ではなく、テーブルの名前である必要があります。 正しい記述: `//DATA Missing_Checks` 間違った記述: `//DATA Missing_Checks.fil` **メモ** 指定する値と同じ名前の既存のAnalyticsテーブルが上書きされます。 **ワイルドカード文字** テーブル名の一部が変化する可能性がある場合は、*名前*にワイルドカード文字を使用することができます。たとえば、テーブル名が月によって異なる場合(invoices-jan、invoices-febなど)

名前	説明
	は、「invoices-*」と指定すれば、月の接尾辞に関係なくテーブルがデータ サブフォルダーにコピーされます。 分析スクリプト内のすべての Analytics 出力テーブルをデータサブフォルダーにコピーするために、単一のワイルドカード文字を指定できます： `//DATA *` **注意** ワイルドカード文字を使用するときには注意してください。指定するワイルドカード パターンが意図していないテーブルと一致する場合、既存のデータ テーブルが意図せず上書きされる場合があります。 ベストプラクティスとして、*名前*値をできるかぎり具体的にします。必要な場合にのみワイルドカード文字を使用してください。 **ロボット へのアップロード** ロボット へのアップロードについては、"クラウド ベースのロボット モジュールへのアップロード" ページ 866を参照してください。

例

Analytics テーブルを格納場所にコピーする

次のアナリティクス ヘッダーは、関連付けられたスクリプトで出力されるインボイステーブルが格納場所にコピーされることを指定します：

```
COMMENT
//ANALYTIC テーブルのインポート
//DATA Invoices
END
IMPORT DELIMITED TO Invoices "Invoices.fil" FROM "Invoices.csv" 0 SEPARATOR "," QUALIFIER
"" CONSECUTIVE STARTLINE 1 KEEPTITLE ALLCHAR ALLFIELDS
```

備考

出力テーブルの格納

出力テーブルは自動的に格納場所にコピーされません。格納したいテーブルごとにDATA タグを使用する必要があります。必要に応じて、1 つのアナリティクス ヘッダーに複数の DATA タグを含めることができます。

DATA タグを使用するとき

次の 2 つの状況では、DATA タグを使用して、Analytics テーブルを格納する必要があります。

- 出力テーブルが後続のアナリティクス スクリプトの入力として使用される
- ユーザーがアドホックでアナリティクスをスケジュールまたは実行するときに、入力テーブルまたはフィールドを選択できる

> **メモ**
>
> データ分析プロセス全体が単一のアナリティクス スクリプトを使用して完了する場合、DATA タグを使用する必要はありません。
>
> DATA タグは、結果テーブルを指定するために使用するものではありません。代わりに RESULT タグを使用します。詳細については、"RESULT" ページ 867を参照してください。

出力テーブルが後続のアナリティクス スクリプトの入力として使用される

展開環境および関連付けられたスクリプトの構造によっては、DATA タグを使用して、後続のアナリティクスで使用するための Analytics 出力テーブルを格納しなければならない場合があります。

アナリティクス処理中に、ロボットおよび AX Server は一時ディレクトリを使用して、Analytics 出力テーブルを格納してアクセスするため、DATA タグを使用する必要がない場合があります。

以下の表は指針です。

展開環境	DATA タグを使用する場合	DATA タグを使用する必要がない場合
ロボット (Enterprise Edition のみ)	◦ 1 つのロボット タスク内で出力された Analytics テーブルが、別のロボット タスクの入力として必要である	◦ Analytics テーブルが出力され、単一のロボット タスクでの一連のアナリティクス スクリプト実行中に後から入力される ◦ データ分析プロセス全体が単一のアナリティクス スクリプトを使用して完了する
AX Server	◦ 1 つのアナリティクス スクリプトによって出力された Analytics テーブルが、別のアナリティクス スクリプトの入力として必要である	◦ データ分析プロセス全体が単一のアナリティクス スクリプトを使用して完了する
分析アプリ ウィンドウ	◦ 1 つのアナリティクス スクリプトによって出力された Analytics テーブルが、別のアナリティクス スクリプトの入力として必要である	◦ データ分析プロセス全体が単一のアナリティクス スクリプトを使用して完了する

ユーザーが入力テーブルまたはフィールドを選択できる

TABLE および FIELD アナリティクス タグは、ユーザーが Analytics テーブルを選択し、テーブルからフィールドを選択できる入力パラメーターを作成し、アナリティクス スクリプトの入力として使用します。ただし、選択できるようにするために、テーブルが格納場所に予め存在する必要があります。

ユーザーが 1 つ以上の入力テーブルとフィールドを選択できるアナリティクスを開発している場合は、DATA タグの前のアナリティクスを実行し、適切なテーブルを保存場所に保存する必要があります。

ロボットのソース テーブル セクションで出力テーブルを見つける

任意で、src_ プレフィックスを出力テーブル名に追加し、ロボットの [**入力/出力**]タブの [**ソース テーブル**]セクションで出力テーブルを検索することができます。

```
//DATA src_Invoices
```

//DATA タグと付属するスクリプトの両方で、プレフィックスをテーブルに追加する必要があります。

[**ソース テーブル**]セクションでは、後続のスクリプトの入力を提供するテーブルを視覚的に区切ることができます。出力テーブル名に src_ プレフィックスがない場合、[**ソース テーブル**]セクションは [**入力/出力**]タブに表示されず、すべてのテーブルは [**他のテーブル**]セクションでデフォルトで検索されます。

クラウド ベースのロボット モジュールへのアップロード

ロボット インストールでアナリティクス スクリプトを実行し、DATA 指定すると、オンプレミス ロボット エージェントから HighBond のクラウド ベースのロボット モジュールにテーブル レイアウトのみ(ファイル名、データ型、ファイル長さ) をアップロードします。テーブル データは、ロボット エージェント ディレクトリ内の組織のネットワークにあります。

すべての情報は転送中に暗号化されます。

AX Server でのリンクまたは共有されたテーブルの上書き

AX Server で出力テーブルがリンクまたは共有されたテーブルを上書きする場合、テーブルがスタンドアロン テーブルに変更されます。

RESULT

エンド ユーザーからクライアント アプリケーションでアクセスできるようにするアナリティクス出力結果を指定します。

出力結果は、存在していても、自動的に利用可能にはなりません。利用可能にしたい結果項目ごとに、RESULT タグを使用する必要があります。

構文

```
//RESULT 種類 名前
 <説明>
```

パラメーター

名前	説明
種類	結果項目の種類： ◦ TABLE - Analytics テーブルと関連付けられたデータ ファイル (.fil) ◦ Log - アナリティクスのログ ファイル ◦ FILE - Analytics ファイル以外のファイル ロボットへのアップロードについては、"クラウド ベースのロボット モジュールへのアップロード" ページ 869を参照してください。
名前	結果項目の名前。*名前*値にスペースを含めることはできません。 **メモ** *名前*値はアナリティクス スクリプトの結果項目の名前と正確に一致する必要があります。*名前*で項目名を指定していません。スクリプトで指定された名前と一致しています。 ### テーブル名 *名前*値には、Analytics テーブル名を指定します。ソース データ ファイルの名前ではなく、テーブルの名前である必要があります。 正しい記述： `//RESULT TABLE Missing_Checks` 間違った記述： `//RESULT TABLE Missing_Checks.fil` #### ワイルドカード文字

名前	説明
	テーブル名の一部が変化する可能性がある場合は、*名前*にワイルドカード文字を使用することができます。たとえば、テーブル名が月によって異なる場合(invoices-jan、invoices-feb など)は、「invoices-*」と指定すれば、月の接尾辞に関係なく結果でテーブルを利用できるようになります。 **ログ名** 省略可能。*名前*値はアナリティクスログファイル名を指定します。*名前*を指定しない場合は、デフォルトログ名の *analytic_name*.log が使用されます。 **メモ** ログ名を指定する場合は、SET LOG TO *ログ名* がスクリプトに表示される必要があります。 **ファイル名** *名前*値には、Analytics ファイル以外のファイルの名前を指定します。 出力される非 Analytics ファイルの種類には、適切なファイル拡張子を指定する必要があります。 正しい記述: `//RESULT FILE Missing_Checks.xlsx` 間違った記述: `//RESULT FILE Missing_Checks` **ワイルドカード文字** *名前*の値の全部または一部にワイルドカード文字を使用して、特定の拡張子(*.xlsx)を持つすべてのファイルや、ファイル名の一部が変化する可能性のあるファイルを指定することができます。 たとえば、ファイル名が月によって異なる場合(invoices-jan.xlsx、invoices-feb.xlsx など)は、「invoices-*.xlsx」と指定すれば、月の接尾辞に関係なく結果でファイルを利用できるようになります。
説明 省略可能	結果やその他の情報に関する説明テキスト。説明を複数行で指定することはできますが、行を飛ばすことはできません。

例

Analytics テーブルを示す RESULT タグ:

```
//RESULT TABLE Missing_Checks
```

アナリティクスのログを示す RESULT タグ(名前はデフォルト値):


```
//RESULT LOG
```

アナリティクスのログを示す RESULT タグ(名前は指定された値) :

```
//RESULT LOG My_log_name
.
.
.
SET LOG TO My_log_name
```

特定の Excel ファイルを示す RESULT タグ:

```
//RESULT FILE Missing_Checks.xlsx
```

すべての Excel ファイルを示す RESULT タグ:

```
//RESULT FILE *.xlsx
```

備考

クラウド ベースのロボット モジュールへのアップロード

ロボット インストールで実行されるアナリティクス スクリプトを使用して、RESULT LOG または RESULT FILE を指定すると、アナリティクス ログ-ファイルまたは非 Analytics ファイルがオンプレミス ロボット エージェントから HighBond のクラウド ベースのロボット モジュールにアップロードされます。

ログの詳細については、"ログ ファイルの出力方法" 下を参照してください。

RESULT TABLE を指定すると、テーブル レイアウトのみ(ファイル名、データ型、フィールド長さ) がアップロードされます。結果データは、ロボット エージェント ディレクトリ内の組織のネットワークにあります。

すべての情報は転送中に暗号化されます。

ログ ファイルの出力方法

アナリティクス スクリプトのログ ファイルを出力する方法は、スクリプトの成否、およびスクリプトが実行されている環境によって異なります。

アナリティクス スクリプト	ロボット エージェント	AX Server	分析アプリ ウィンドウ
成功	◦ **RESULT LOG が指定される** クラウド ベースのロボット モジュー	◦ **RESULT LOG が指定される** AX Server へのログ ファイル出	◦ **RESULT LOG が指定される** 結果タブへのログ ファイル出力

アナリティクス スクリプト	ロボット エージェント	AX Server	分析アプリ ウィンドウ
	ルにアップロードされたログ ファイル ◦ **RESULT LOG が指定されない** ログ ファイルなし	力(クライアント アプリケーションで使用可能) ◦ **RESULT LOG が指定されない** ログ ファイルなし	◦ **RESULT LOG が指定されない** ログ ファイルなし
失敗	◦ **RESULT LOG タグが考慮されない** • ログ ファイルは自動的にロボット エージェント ベース データ フォルダーに出力される (configuration setting = "false") • ログ ファイルは自動的にクラウド ベースのロボット モジュールにアップロードされます (configuration setting = "true" (デフォルト)) ロボット エージェントの構成の構成設定 **UploadLogsWhenFailed** を参照してください。	◦ **RESULT LOG タグが考慮されない** AX Server への自動ログ ファイル出力(クライアント アプリケーションで使用可能)	◦ **RESULT LOG タグが考慮されない** 結果タブへの自動ログ ファイル出力

AX Server での結果ファイル サイズの制限

AX Server で実行されるアナリティクス スクリプトの結果ファイルは、最大 2 GB に制限されます。このサイズを超えた結果ファイルは保存されません。

AX Server でのスクリプト実行中の結果ファイルの保管および可用性

//RESULT FILE タグを使用した場合、スクリプトを実行すると、作成されたファイルは AX Web Client および AX Client からダウンロードすることができます。このファイルは、AX データベースに保管されますが、スクリプトが実行されていないときには AX Server のファイル システムには存在しません。

スクリプトの実行中、このファイルは AX Server のファイル システムに一時的に存在し、EXECUTE コマンドを使って呼び出すような外部プロセスを使って操作することができます。スクリプトの実行中、外部プロセスはアナリティクス ジョブのサブフォルダーにあるこのファイルにアクセスすることができます。

> **メモ**
>
> デフォルトでは、アナリティクス ジョブのサブフォルダーは `ACL\Data\jobs` 内に配置されます。スクリプトの実行後、アナリティクス ジョブのサブフォルダーは削除され、このファイルはデータベースに保管されます。

PUBLISH

アナリティクスが処理を完了したときに、どの Analytics テーブルを AX Exception に公開するかを定義するメタデータを含んでいるファイルを指定します。

構文

```
//PUBLISH ファイル名
```

パラメーター

名前	説明
ファイル名	AX Exception 用の公開メタデータを含んでいるファイルの名前。

例

アナリティクスの定義と、そのアナリティクスについて公開する詳細を指定するテキスト ファイル。

公開ファイルが AX フォルダーに格納されている場合、ファイルはアナリティクスの処理中に取得されるため、FILE タグが必要となります。

```
COMMENT
//ANALYTIC 結果の公開
//RESULT TABLE Results
//FILE ex_publish.txt
//PUBLISH ex_publish.txt
END
EXTRACT RECORD TO Results
```

コレクション内の **関連ファイル** サブフォルダーにアップロードされる `ex_publish.txt` ファイルには、次のテキスト行が含まれています。値は引用符で囲まれており、かつ、"テーブル名","エンティティ名","アナリティクス名" のような構文を使用しなければなりません。たとえば、次のように指定します。

```
"Results","MyEntity","MyAnalytic"
```

アナリティクスの開発

デバッグを容易にし、エラーを切り分けるため、アナリティクスヘッダーを追加する前にスクリプトの本文を記述します。アナリティクスヘッダーを追加したら、ログファイルと一時的なテスト値を使ってアナリティクスがどのように実行されるかをテストします。最後に、ターゲット環境にアナリティクスを展開します。

> **メモ**
>
> 次のワークフローをアナリティクス開発方法として推奨しますが、ご自分に最も適した方法で自由にアナリティクスを開発することができます。

アナリティクス スクリプトの作成とテストのワークフロー

Analytics スクリプトを作成する

ユーザー入力のためのカスタム ダイアログ ボックスや、その他スクリプトの実行中にユーザー操作を必要とする機能を一切使用せずに、Analytics でスクリプトを作成します。アナリティクスは、アナリティクスの実行に先立ってユーザー入力をできるようにしますが、スクリプトとは異なり、実行中のユーザー操作はサポートしていません。

Analytics スクリプトにテスト入力値を保存するには、スクリプトの先頭で、変数を一時的に作成します。

```
ASSIGN v_AnalysisTable = "Trans_May"
```

スクリプトがエラーなしで実行されるまで、スクリプトをテストしてデバッグします。

アナリティクス ヘッダーおよびアナリティクス タグの追加

スクリプトにアナリティクス ヘッダーを追加します。スクリプトの先頭からアナリティクス ヘッダー内の対応するタグへ変数名をコピーします。

```
//TABLE v_AnalysisTable "Table to classify"
```

詳細については、"アナリティクス ヘッダーの追加" ページ 876を参照してください。

アナリティクス結果にログを含める

ログは、アナリティクスが失敗した原因を診断する上で不可欠なツールですが、アナリティクスが成功しても、予期しない結果が生じた場合には重要となります。ログは、アナリティクスが失敗した場合には自動的に出力されますが、アナリティクスが成功した場合は、RESULT アナリティクス タグを指定した場合にのみ出力されます。

アナリティクス ヘッダーに次の行を含めて、アナリティクスを実行するときは、いつでもログを利用できるようにします。

```
//RESULT LOG
```

アナリティクス ヘッダーを検証する

アナリティクス ヘッダーを検証します。必要とする頻度でアナリティクス ヘッダーを検証することができます。

詳細については、"アナリティクス ヘッダーの検証" ページ 877を参照してください。

アナリティクス タグに一時的なテスト値を代入する

特殊な代入演算子(:=)を使用して、ユーザー入力を必要とするすべての分析タグに一時的なテスト値を代入します。スクリプトの先頭にある一時的な変数代入からテスト値をコピーすることができます。

```
//TABLE v_AnalysisTable "分類するテーブル" := "Trans_May"
```

一時的なテスト値の詳細については、"Analytics でのテスト入力値の指定" ページ 838を参照してください。

一時変数を削除する

スクリプトの先頭から一時変数を削除するか、または、それらを再度使用するかもしれない場合は、一時変数をコメントにします。

アナリティクスをステップ実行する

実行  をクリックするか、または **F10** キーを繰り返し押して、アナリティクスをステップ実行します。**ナビゲーター**の [**変数**]タブの内容をよく調べて、アナリティクス ヘッダー内のすべての変数が正しく作成され、テスト値が正しく割り当てられていることを確認します。

アナリティクスがエラーなしで実行されるまで、テストしてデバッグします。

テストが完了したら、すべてのアナリティクス タグから、一時的なテスト値と特殊な代入演算子を削除します。

> **メモ**
>
> アナリティクスが完了する前にそのアナリティクスを終了したい場合は、**Esc** キーを押し、確認プロンプトで [**はい**]をクリックします。

> **ヒント**
>
> コマンド ラインで DELETE ALL OK と入力することにより、Analytics プロジェクトからすべての格納された変数と変数の代入値を削除することができます。ステップ実行する前に [**変数**]タブをクリアすると、アナリティクスをクリーンに開始できます。

分析アプリのテストのワークフロー

AX Web Client または分析アプリウィンドウでアナリティクスを実行する場合は、分析アプリをテストする必要があります。

余分なテーブルレイアウトを削除する

分析アプリのすべてのアナリティクスと任意のサブスクリプトがテストおよびデバッグされ、正しく実行されたら、分析アプリに含めないテーブルレイアウトを Analytics プロジェクトから削除します。

余分なテーブルレイアウトがあると、AX Client、AX Web Client、および分析アプリウィンドウにおける分析アプリが乱雑になり、エンドユーザーを混乱させる可能性があります。

分析アプリウィンドウで分析アプリを開く

[**総覧**]タブで Analytics プロジェクトエントリを右クリックし、[**分析アプリとして開く**]を選択することにより、完成した分析アプリを分析アプリウィンドウで開きます。

> **メモ**
>
> 分析アプリが開かず、アナリティクスに同じ名前が含まれていることを示すエラーメッセージが表示された場合は、エラーメッセージで指定された各アナリティクスの ANALYTIC タグで *名前*の値を調べてください。アナリティクスの *名前*値は、Analytics プロジェクト内で一意でなければなりません。

アナリティクスを実行する

分析アプリ内のすべてのアナリティクスを実行して、それらが正しく機能していることを確認します。

ANALYTIC タグで TYPE オプションを使用して、インポート、準備、および分析のアナリティクスを作成している場合は、アナリティクスを実行する正しい順序に従ってください。

ログを調べる

アナリティクスが失敗した場合は、ログファイル(*アナリティクス名*.log) を開いて調べます。ログには、アナリティクスが失敗した理由を示す、赤い X 印が付けられたエントリが 1 つ含まれています。

- 誤って入力された入力値の場合は、正しく入力された入力値を使って直ちにアナリティクスを再実行します。
- スクリプト本文の構文または論理エラーについては、Analytics のエラーを修正してから、分析アプリウィンドウで分析アプリをもう一度開きます。

アナリティクスは成功しても、結果テーブルに期待していた結果が含まれていないことがあります。このような場合は、アナリティクスが意図したとおりに機能していることを確実にするために、ログエントリを順々に見直して、アナリティクスに渡された入力値をチェックします。

分析アプリのパッケージ化および検証

分析アプリをパッケージ化またはインポートする

分析アプリが目的どおりに機能していると確認できたら、それを配布して分析アプリウィンドウで使用する場合はパッケージ化し、AX Client または AX Web Client で使用する場合は AX Server にインポートします。詳細については、"分析アプリのパッケージ化" ページ 885を参照してください。

AX Server 分析アプリを実行する

AX で使用するためのアナリティクスを開発している場合は、AX Client と AX Web Client の両方を使用してすべてのアナリティクスを実行して、それらが目的どおりに機能していることを確認します。

アナリティクス ヘッダーの追加

アナリティクス ヘッダーは、スクリプトの先頭にコメントで囲んで追加する一連の宣言タグです。ヘッダーには、即時、または予定時刻のいずれかに、アナリティクスによる無人の実行を可能にする、ユーザーが事前に設定する入力パラメーターが含まれます。

Analytics プロジェクトでスクリプトを開発した後、スクリプトをロボットにコミットするか、AX Server または分析アプリウィンドウで、そのプロジェクトを分析アプリとして使用するには、スクリプトにアナリティクス ヘッダーを追加する必要があります。

アナリティクス ヘッダー要件

アナリティクス ヘッダーは特定の要件に適合する必要があります。適合しない場合は、実行時にアナリティクス スクリプトが失敗します。

アナリティクス ヘッダー構文の詳細と、アナリティクス タグの完全な一覧については、"アナリティクス ヘッダーおよびアナリティクス タグ" ページ 837を参照してください。

アナリティクス ヘッダーをコメント ブロックで囲む

アナリティクス ヘッダーは、スクリプトの先頭行から始まるコメント ブロック内に漏れなく定義する必要があります。

```
COMMENT
アナリティクス タグをここに指定します。
END
```

入力および出力を宣言

アナリティクス ヘッダーにおいて、アナリティクスに必要なすべての入力を宣言すると共に、ロボットまたは AX Server にコピーされるか、または分析アプリウィンドウに結果として書き出されるすべての出力も宣言する必要があります。

入力	出力
◦ テーブル ◦ フィールド ◦ のパラメーター ◦ 入力ファイル ◦ パスワードの確認	◦ データ ファイル ◦ 結果テーブル ◦ ログ ファイル

入力値を宣言

また、アナリティクス ヘッダーには、ユーザーがアナリティクスを実行またはスケジュールする際に、ユーザーが指定するすべての入力値も宣言する必要があります。

ユーザーが指定した入力値を受け入れて、変数に格納する入力パラメーターを追加するには、PARAM タグを使用します。たとえば、アナリティクスに日付範囲に基づいてデータを選択させる場合は、開始日と終了日をユーザーが指定できるように、これらのパラメーターを追加する必要があります。

アナリティクス ヘッダーの検証

1 つ以上のスクリプトにアナリティクス ヘッダーを追加したら、Analytics のツールを使用して、ヘッダー構文を検証し、それが正しいことを確認します。スクリプトをロボットにコミットするか、分析アプリをパッケージ化する前に、検証を実行し、アナリティクスの実行時に失敗しないようにします。

1 つのツールは、スクリプト レベルで個別のアナリティクス ヘッダーを検証します。他のツールは、1 度に 1 つのプロジェクトのすべてのアナリティクス ヘッダーを検証します。2 つの種類の検証の焦点は異なります。

個別のアナリティクス ヘッダーを検証する

アナリティクス ヘッダーのスクリプト レベルの検証は、個別のアナリティクス タブの構文に焦点を置き、エラーと該当する行番号を報告します。

1. アナリティクス ヘッダーを含んでいるスクリプトを開きます。
2. スクリプト エディター ツールバーで、 [**アナリティクス ヘッダーの検証**]をクリックします。

   アナリティクス ヘッダーが有効であることを知らせるメッセージか、または発生したエラーと発生場所の行番号を示すメッセージが表示されます。
3. アナリティクス ヘッダーにエラーがある場合は、エラーを修正してからもう一度 [**アナリティクス ヘッダーの検証**]をクリックし、これ以上エラーがないことを確認します。

> **ヒント**
>
> エラーの本質がエラー メッセージからは明らかでない場合は、関連するアナリティクス タグのヘルプ トピックを参照してください。トピックの構文をアナリティクス ヘッダーの構文と注意して比較します。エラーは、アナリティクス ヘッダー構文のわずかな違いが原因で発生することもあります。

プロジェクトのすべてのアナリティクス ヘッダーを検証する

アナリティクス ヘッダーのプロジェクト レベルの検証は次の 2 つのことを確認します。

- 1 つ以上のアナリティクス ヘッダーがプロジェクトに存在する
- 複数のアナリティクスの名前が一意である

> **メモ**
>
> **ナビゲーター**の [**概要**]タブのスクリプト名ではなく、ANALYTIC タブで指定された名前を参照します。

スクリプトをロボットにコミットするときに、プロジェクト レベルの検証が自動的に実行されます。 [**スクリプトの確認** ]ボタンを Analytics ツールバーに追加する場合に、手動で検証を実行することもできます。

1. 必要に応じて、 [**スクリプトの確認**]ボタンを Analytics ツールバーに追加します。
   a. ツール バー上の何もない場所をダブルクリックすると、 [**ツール バーの変更**]ダイアログ ボックスが開きます。
   b. [**利用できるツール バー ボタン**]リストで、 [**スクリプトの確認**]を選択し、 [**追加**]をクリックします。
   c. [**現在のツール バー ボタン**]リストで、 [**スクリプトの確認**]ボタンを選択し、 [**上へ**]または [**下へ**]をクリックして、ボタンの位置を変更します。

      上から下へのボタンの順序は、ツール バーの左から右への位置に対応しています。

   d. [**閉じる**]をクリックして、変更を保存します。

2. ツールバーで、 [**スクリプトの確認** ]をクリックします。

   プロジェクトのアナリティクス ヘッダーが有効であるというメッセージが表示されるか 1 つ以上のエラーを指定します。

3. アナリティクス ヘッダーにエラーがある場合は、エラーを修正してからもう一度 [**スクリプトの確認** ]をクリックし、これ以上エラーがないことを確認します。

アナリティクス ヘッダーの例

```
COMMENT
//ANALYTIC Identify missing checks
 このアナリティクスは見つからないチェック番号を特定します。
//TABLE v_table_payments 支払いテーブル
 支払いを一覧表示していて小切手番号列が含まれるテーブルを選択します。
//FIELD v_check_num CN Check Number
 小切手番号が含まれているフィールドを選択します
//PARAM v_start_date D OPTIONAL 開始日 ( 任意 )
 分析の開始日を入力
//PARAM v_end_date D OPTIONAL 終了日 ( 任意 )
 分析の終了日を入力
//PARAM v_region C MULTI SEPARATOR , QUALIFIER ' Region(s)
 分析に含める 1 つ以上の地域を入力
//RESULT LOG
//RESULT TABLE MissingChecks
//RESULT FILE MissingCheckDetails.xls
END
```

アナリティクス開発のベスト プラクティス

アナリティクスは、標準の Analytics スクリプトで使用できるコマンドのほとんどをサポートしています。開発者としては、アナリティクスがユーザーの操作なしに実行できるようにすると共に、アナリティクスが、展開環境でそれを処理するエンジンでサポートされていないコマンドを含まないようにする必要があります。

Analytics ではすべての Analytics 関数がサポートされます。

一般的なベストプラクティス

ロボットまたは分析アプリごとに 1 つの Analytics プロジェクトを使用する

ロボットまたは分析アプリごとに Analytics で新しい Analytics プロジェクトを作成します。プロジェクトには、ロボットまたは分析アプリを構成するすべてのアナリティクスと、サブスクリプトで必要となるすべてのが含まれている必要があります。分析アプリの場合、プロジェクトには、アナリティクスで必要となるすべてのデータ ファイルが含まれている必要があります。

ローカルでテストする

ターゲット環境にアナリティクスを展開する前に、すべてのアナリティクスをローカルでテストします。アナリティクスが想定通りに実行され、ユーザー操作が必要でないことを確認します。

詳細については、"アナリティクスの開発" ページ 872を参照してください。

テストに安定したデータ接続を使用する

ODBC データ ソースを使用しているかどうかについてローカルでアナリティクスをテストするには、お使いのローカル コンピューター上で、アナリティクスが実行する環境の接続と一致する、ODBC 接続を構成する必要があります。

分析アプリ ウィンドウで使用するために配布されたアナリティクスの場合、エンド ユーザーが自身のコンピューター上に等しい ODBC を構成する必要があります。

絶対ファイル パスの使用を避ける

アナリティクスが実行される環境に同一のファイル パスが存在していることが確実な場合以外、アナリティクスで絶対ファイル パス(`C:\results` など) を使用することは避けてください。

相対ファイル パス(`\results`など) を使用すると、アナリティクスをローカルで開発してテストし、他の環境に同じディレクトリ構造がなくても、別の環境に展開できます。

SET を使って初期設定を行う

SET コマンドを使用して、アナリティクスで必要となるすべての初期設定を指定します。アナリティクスの初期設定をしない場合は、Analytics のデフォルトの初期設定が使用されます。アナリティクス ヘッダーの後ろ、アナリティクス ロジックの前に SET コマンドを置きます。

結果テーブルやデータ テーブルでは演算フィールドは使用禁止

アナリティクス スクリプトを実行するセッション以外のセッションでも保持したいテーブル内では、演算フィールドを使用しないでください。

解釈内での使用目的に、あるいは後続のスクリプトの入力として保持される結果テーブルやデータ テーブルが演算フィールドを含んでいる場合には、予期しない値が表示される可能性があります。演算値は初期設定ファイル(.prf) に定義されている設定によって決まるか、または SET コマンドを使って作成されるため、環境に応じて、出力される値が異なってきます。

演算フィールドの値を保持する必要がある場合は、EXTRACT コマンドを FIELDS または ALL オプションとともに使用することで、そのフィールドを結果テーブルやデータ テーブルの物理フィールドに変換します。詳細については、"EXTRACT コマンド" ページ 193を参照してください。

データ接続のパスワードを暗号化する

アナリティクスにデータ ソースのパスワードをプレーン テキストで設定しないようにするには、PASSWORD アナリティクス タグを使用します。このタグは、アナリティクスを実行する前に、パスワードの入力をユーザーに求め、入力された値を暗号化します。

HighBond との間でインポートまたはエクスポートを行うときにパスワードを使用する

PASSWORD パラメーターは、HighBond との間でインポートまたはエクスポートを行うすべてのコマンドで必要です。

- IMPORT GRCRESULTS
- IMPORT GRCPROJECT
- EXPORT... ACLGRC

PASSWORD パラメーターがない場合、ロボット、Analytics Exchange、または分析アプリ ウィンドウでコマンドが失敗します。

アナリティクス スクリプトで、PASSWORD パラメーターを使用するときには、アナリティクス ヘッダーで関連付けられたパスワード入力パラメーターも指定する必要があります。詳細については、"PASSWORD" ページ 860を参照してください。

> **メモ**
>
> PASSWORD パラメーターは、Analytics でインポートおよびエクスポート コマンドを実行するときには不要です。現在のユーザーの HighBond アクセストークンが自動的に使用されます。

ユーザー操作の回避

アナリティクスは、ユーザーの操作なしに実行できるようにしておく必要があります。コマンドがダイアログ ボックスを作成しようとすると、開発環境のエンジンがアナリティクスを停止し、エラーがログに記録されます。

ユーザー操作コマンドをアナリティクス タグに置き換える

ユーザー操作を必要とする Analytics コマンドを使用しないようにします。アナリティクス ヘッダーの同等のアナリティクス タグで置換します。アナリティクス タグでは、アナリティクスを実行する前に、ユーザーが入力値を指定できます。

使用しない	置換
DIALOG	//TABLE, //FIELD, //PARAM
ACCEPT	//TABLE, //FIELD, //PARAM
PASSWORD	//PASSWORD
PAUSE	相当するオプションなし

ガイドライン

- アナリティクス処理が失敗しないようにするには、すべての操作コマンドを削除する
- 確認ダイアログ ボックスを表示しないで、必要に応じてファイルを上書きできるようにするため、アナリティクスの先頭に SET SAFETY OFF コマンドを追加すると共に、デフォルトの動作を復元するため、アナリティクスの終わりに SET SAFETY ON コマンドを追加する
- 確認ダイアログによってアナリティクスがクラッシュしないようにするには、通常、確認ダイアログ ボックスを表示する以下のコマンドの後に OK パラメーターを追加します。
  - RENAME
  - DELETE

スクリプト構文を確認する

Analytics は、アナリティクスが失敗する原因となるスクリプト構文、またはローカル環境とアナリティクスが展開される環境との間で調整が必要になるスクリプト構文を検出するツールを提供しています。このツールでは警告のみが表示されます。警告のあるアナリティクス スクリプトでも、コミットまたはインポートできます。

ツールの確認機能

このツールは、プロジェクトのすべてのスクリプトを確認し、次の3つの点を検出します。

- ユーザー操作を必要とするコマンド
- 絶対ファイルパス
- 外部スクリプトの呼び出し

ツールは、次の項目について、プロジェクトのすべてのスクリプトを確認します。

- ユーザー操作を必要とするコマンド
- 絶対ファイルパス
- 外部スクリプトの呼び出し
- HighBondとの間でインポートまたはエクスポートを実行するすべてのコマンド

確認が実行されるタイミング

スクリプト構文確認は、スクリプトをロボットにコミットするときに、自動的に実行されます。

構文の確認はデフォルトで有効です。オフにする場合は、[**オプション**]ダイアログボックス([**ツール > オプション > インターフェイス**])で、[**スクリプトをコミットする前に、スクリプト構文チェックを無効にする**]を選択します。

確認を手動で実行する

スクリプト構文の確認を手動で実行できます。Analyticsツールバーに**スクリプトの確認**  ボタンを最初に追加しなければならない場合があります。

1. 必要に応じて、[**スクリプトの確認**]ボタンをAnalyticsツールバーに追加します。
   a. ツールバー上の何もない場所をダブルクリックすると、[**ツールバーの変更**]ダイアログボックスが開きます。
   b. [**利用できるツールバーボタン**]リストで、[**スクリプトの確認**]を選択し、[**追加**]をクリックします。
   c. [**現在のツールバーボタン**]リストで、[**スクリプトの確認**]ボタンを選択し、[**上へ**]または[**下へ**]をクリックして、ボタンの位置を変更します。

      上から下へのボタンの順序は、ツールバーの左から右への位置に対応しています。

   d. [**閉じる**]をクリックして、変更を保存します。
2. ツールバーで、[**スクリプトの確認** ]をクリックします。

   プロジェクトのスクリプト構文が有効であること、または1つ以上の警告を示すメッセージが表示されます。
3. 次のいずれかを実行します。
   - 警告があるスクリプト構文を修正し、[**スクリプトの確認** ]をもう一度クリックして、警告が表示されないことを確認します。
   - 展開環境にディレクトリ構造、またはアナリティクスで指定されたパスまたは外部スクリプトと一致する外部スクリプトがあることを確認します。

AX Server でのアナリティクス実行のベスト プラクティス

Analytics を使用した開発

アナリティクスとそれらに関連するスクリプトをまず Analytics で開発してから、AX Server にインポートします。

便利な機能として、AX Client スクリプト エディターでは、新しいアナリティクスやサブスクリプトを追加したり、既存のアナリティクスやサブスクリプトを編集したりすることができます。この機能は、アナリティクスの動作を微調整する場合に便利です。Analytics にエクスポートして AX Server に再インポートする必要がなくなります。それどころか、微調整レベルを超えるアナリティクス開発作業でさえも Analytics では行いやすくなりました。

Analytics プロジェクトおよびその関連ファイルを保存する

データベース プロファイル ファイルなどの関連ファイルは Analytics プロジェクトと同じフォルダーに保存されますが、別途 AX Server にもインポートする必要があります。

AX Server でサポートされないコマンドの使用を避ける

- Analytics Server Edition for z/OS にリンクされたデータベース サーバー テーブルへの直接アクセス
- NOTIFY コマンドは、SMTP メッセージングのみサポートします。MAPI および VIM メール プロトコルはサポートされていない
- PRINT または TO PRINT コマンドを使用する場合は、通常使うプリンターがサーバーに設定されている必要がある
- SAVE GRAPH および PRINT GRAPH コマンドはサポートされていない
- アナリティクス内で SET LEARN コマンドを使用することはできない

AX Server テーブルトランザクションを最小限度に抑える

AX Server のテーブルがアクセスされる回数を最小化することによって、アナリティクスのパフォーマンスを最適化します。

1. FILTER コマンドを使用すると、必要なレコードを選択します。
2. EXTRACT コマンドを使用すると、必須フィールドのみを抽出します。

最小限度のデータ セットが、AX Engine によってアナリティクスが実行されているサーバー上でローカルに処理されます。

データ ファイルが AX Server と同じサーバー上、つまり、アナリティクスを処理する AX Engine Node 上に置かれておらず、AX Server Configuration Web アプリケーションで［**Copy analytic data**］オプションが選択されていない場合には、このようにアナリティクスを最適化することが重要です。

非効率的なアナリティクスの例

```
OPEN LargeTable
SET FILTER TO trans_date >= `20091201` AND trans_date < `20100101`
COUNT
TOTAL amount
CLASSIFY ON account ACCUMULATE amount TO TransClassAccount
```

効率的なアナリティクスの例

```
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
```

バックグラウンド モードで SAP データにアクセスする

Direct Link を使用して SAP ERP システム内のデータにアクセスするには、Background モードを使用します。

分析アプリのパッケージ化

分析アプリウィンドウまたはAX Serverでアナリティクススクリプトを実行するには、パッケージ化された分析アプリ(.aclappファイル)を作成します。Analyticsプロジェクトを新しい.aclappファイルにパッケージ化するか、Analyticsプロジェクトを既存の分析アプリ(.aclxファイル)とマージし、既存の解釈を含めます。

> **メモ**
>
> Analyticsプロジェクト内の少なくとも1つのスクリプトにアナリティクスヘッダーが含まれていれば、そのプロジェクトを分析アプリとしてパッケージ化することができます。分析アプリをパッケージ化する前に、分析アプリ内の各アナリティクスのアナリティクスヘッダーを確認してください。

パッケージ化された分析アプリを使用する理由

分析アプリウィンドウユーザーへの配布

.aclxファイルを抽出して分析アプリウィンドウでないようを開けるユーザーにプロジェクトを配布するために、パッケージ化された分析アプリを使用します。

分析アプリウィンドウで、分析スクリプトを実行し、テーブルとアナリティクス結果に基づいて解釈を作成できます。

AX Serverへのインポート

AX Serverへのインポート対象としてAnalyticsプロジェクトを準備するには、パッケージ化された分析アプリを使用します。プロジェクトのアナリティクススクリプトとともにインポートするテーブルとデータファイルを選択できます。

パッケージ化された分析アプリ(.aclapp)を使用して、既存の解釈をインポートできます。既存の分析アプリ(.aclx)内の解釈を追加するには、Analyticsプロジェクトをパッケージ化し、このパッケージ化された分析アプリ(.aclapp)をプロジェクトフォルダーの既存の.aclxファイルとマージする必要があります。

> **メモ**
>
> 既存の分析アプリ(.aclxファイル)を使用するときには、Analyticsプロジェクトの内容が優先されるため、.aclファイルに存在しない.aclxにスクリプトまたはテーブルは、結果のパッケージ化された分析アプリ(.aclappファイル)に追加されません。

ファイルサイズ制限

パッケージ化された分析アプリを正常に使用するには、分析アプリをパッケージ化する前に、パッケージに含まれるすべてのファイルサイズの合計が800 MBを超えていないことを確認する必要があります。パッケージ化されたファイルがこの組み合わせられたサイズを超える場合、分析アプリを解凍するときにデータファイルが破損する可能性があります。

新しい分析アプリをパッケージ化する

1. Analytics で、**ナビゲーター**の [**総覧**]タブでプロジェクト エントリを右クリックし、 [**分析アプリのパッケージ化**]を選択します。

   Analytics プロジェクトは、ツリー ビューにおける最上位のフォルダーです。

2. [**テーブルの選択**]ダイアログ ボックスで、次の手順を実行します。
   a. 分析アプリに 1 つ以上のプロジェクト テーブルと、関連するデータ ファイルを含めたい場合は、そのテーブルと含めるデータ ファイルを選択します。

      > **メモ**
      >
      > 通常は、分析アプリ内の 1 つ以上のアナリティクスが必要とする静的なテーブルとデータ ファイルのみ、たとえば、業者マスター テーブルや商業カテゴリコードの一覧などを含める必要があります。

   b. [**転送先**]をクリックし、パッケージ化した分析アプリを保存したい場所に移動します。
   c. [**名前を付けて保存**]ダイアログ ボックスで、`.aclapp`ファイル拡張子の**ファイル名**を入力して [**保存**]をクリックします。
   d. [**OK**]をクリックします。

結果 - パッケージ化した分析アプリが、指定した場所に保存されます。他のユーザーは、この場所からパッケージ化された分析アプリを取得することができます。あるいは、電子メールや他の適切な方法でそれを配布することもできます。このファイルは AX Server にインポートすることもできます。

分析アプリを解釈でパッケージ化

1. インポートする解釈を含む分析アプリ(.aclx ファイル) がコンピューターの Analytics プロジェクト フォルダーにあり、Analytics プロジェクト(`.acl` ファイル) と同じファイル名を使用していることを確認します。

2. Analytics で、**ナビゲーター**の [**総覧**]タブでプロジェクト エントリを右クリックし、 [**分析アプリのパッケージ化**]を選択します。

   Analytics プロジェクトは、ツリー ビューにおける最上位のフォルダーです。

3. [**テーブルの選択**]ダイアログ ボックスで、次の手順を実行します。
   a. 分析アプリに 1 つ以上のプロジェクト テーブルと、関連するデータ ファイルを含めたい場合は、そのテーブルと含めるデータ ファイルを選択します。

      > **メモ**
      >
      > 通常は、分析アプリ内の 1 つ以上のアナリティクスが必要とする静的なテーブルとデータ ファイルのみ、たとえば、業者マスター テーブルや商業カテゴリコードの一覧などを含める必要があります。

   b. 省略可能。既存の分析アプリから解釈を含めるには、 [**解釈を含める**]を選択します。

      新しいパッケージに存在しないテーブルまたはスクリプトに関連付けられた解釈は含まれません。

c. [**転送先**]をクリックし、パッケージ化した分析アプリを保存したい場所に移動します。
d. [**名前を付けて保存**]ダイアログボックスで、`.aclapp`ファイル拡張子の**ファイル名**を入力して[**保存**]をクリックします。
e. [**OK**]をクリックします。

結果 - パッケージ化した分析アプリが、指定した場所に保存されます。他のユーザーは、この場所からパッケージ化された分析アプリを取得することができます。あるいは、電子メールや他の適切な方法でそれを配布することもできます。このファイルは AX Server にインポートすることもできます。

サンプル アナリティクス スクリプト(分 析 アプリ)

アナリティクス スクリプト のサンプルには、1 つのインポート アナリティクス(2 つのバージョン) 、1 つの準 備 アナリティクス、そして 1 つの分 析 アナリティクスが含 まれています。 アナリティクス スクリプト は、次 の環 境 またはクライアント アプリケーションのいずれかで実 行 できます。

- ロボット
- AX Server:
  - AX Client
  - AX Web Client
- 分 析 アプリ ウィンドウ

アナリティクス スクリプト の順 序

3 つのアナリティクスは、合 わせて作 業 するために設 計 され、次 の順 序 で実 行 する必 要 があります。

順番検査	アナリティクスタイプ	アナリティクス名
1	IMPORTIMPORT	インポート アナリティクス Robots_AX のサンプル または サンプル インポート アナリティクス Web_AA_Window
2	PREPARE	準 備 アナリティクスのサンプル
3	分 析	分 析 アナリティクスのサンプル

インポート アナリティクスのサンプル

サンプルの Excel ファイル `Trans_May.xls` からデータをインポートし、それを新しい Analytics テーブル `Trans_May_raw`(生 データ テーブル) に保 存 します。

このアナリティクスの 2 つのバージョンが提 供 されます。

アナリティクス名	使 用 場 所	インポート ファイルの要 件
インポート アナリティクス Robots_AX のサンプル	◦ ロボット ◦ AX Client	◦ ロボット の `Trans_May.xls` ファイルはアナリティクスと同じロボット の **[入 力/出 力]**タブにある必 要 があります ◦ AX Client `Trans_May.xls` は、アナリティクスが配 置 される AX フォルダーの**関 連 ファイル** サブフォルダーにある必 要 があります。
サンプル インポート アナリ	◦ AX Web Client	

アナリティクス名	使用場所	インポート ファイルの要件
ティクス Web_AA_Window	◦ 分析アプリ ウィンドウ	

ロボットまたは AX Client 用インポート アナリティクスのサンプル

```
COMMENT
//ANALYTIC TYPE IMPORT インポート アナリティクス Robots_AX のサンプル
  このアナリティクスは、サンプルの Excel ファイル Trans_May.xls からデータをインポートし、それを新しい
Analytics テーブル "Trans_May_raw"( 生データ テーブル) に保存します。
//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
```

AX Web Client 用または分析アプリ ウィンドウ用のインポート アナリティクスのサンプル

```
COMMENT
//ANALYTIC TYPE IMPORT インポート アナリティクス Web_AA_Window のサンプル
  このアナリティクスは、サンプルの Excel ファイル Trans_May.xls からデータをインポートし、それを新しい
Analytics テーブル "Trans_May_raw"( 生データ テーブル) に保存します。
//PARAM v_input_file F
  入力ファイル 入力ファイルを選択します
//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
```

準備アナリティクスのサンプル

分析対象となる生データテーブルを準備し、それを新しいAnalyticsテーブル`Trans_May_prepared`(分析テーブル)に保存します。分類化では最大64文字のフィールド長しかサポートされないので、このアナリティクスでは"Description"フィールドの短い方のバージョンを定義します。

```
COMMENT
//ANALYTIC TYPE PREPARE 準備アナリティクスのサンプル
  このアナリティクスは分析対象となる生データテーブルを準備し、それを新しいAnalyticsテーブル
"Trans_May_prepared"(分析テーブル)に保存します。分類化では最大64文字のフィールド長しかサ
ポートされないので、このアナリティクスでは"Description"フィールドの短い方のバージョンを定義します。
//TABLE v_RawTable 準備するテーブル
 準備する未加工データテーブルを選択します
//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
```

分析アナリティクスのサンプル

分析テーブルを分類化し、その結果を新しいAnalyticsテーブル`Classified_Trans_May_prepared`(結果テーブル)に出力します。ユーザーは、テーブルを分類化するために使用するフィールドを指定できます。また、処理されるレコードを制限するために、商業カテゴリコード、顧客番号と、日付および取引金額の範囲を指定することができます。

```
COMMENT
//ANALYTIC TYPE ANALYSIS 分析アナリティクスのサンプル
 このアナリティクスは分析アナリティクスのサンプル- 分析テーブルを分類化し、その結果を新しい
Analyticsテーブル"Classified_Trans_May_prepared"(結果テーブル)に出力します。処理されるレコー
ドを制限するために、商業カテゴリコード、顧客番号と、日付および取引金額の範囲を指定できます。
//TABLE v_AnalysisTable 分類するテーブル
 分類する分析テーブルを選択します
 //FIELD v_FieldA C 分類するフィールド
 分類するフィールドを選択します
//PARAM v_codes C MULTI SEPARATOR , QUALIFIER ' VALUES |4112 Passenger Railways|4121
Taxis/Limousines|4131 Bus travel|4215 Courier Services - Air or Ground|4411 Cruise Lines|4457
```

```
Boat Leases and Boat Rentals|4722 Travel Agencies 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
 含める取引先カテゴリコードを１つ以上指定します
//PARAM v_cust_no C OPTIONAL MULTI SEPARATOR , QUALIFIER ' 除外する顧客番号(省略可能)
 １つ以上の除外する顧客番号を指定します。番号の指定後にEnterを押して、各番号を新しい行に移動します。番号を引用符で囲まないでください。
//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|05/0-9/2003|05/10/2003|05/11/2003|05/12/2003|05/13/2003|05/14/2003|05/15/2003|05/16/2003|05/17/200-3|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
 開始日を選択します
//PARAM v_end_date D End Date
 終了日を入力するか、またはカレンダーから終了日を選択します
//PARAM v_min_amount N 最低額
 最低額を入力します
//PARAM v_max_amount N 最高額
 最高額を入力します
//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_AnalysisTable%.FIL" OPEN
END
SET SAFETY ON
```

付録

システム要件

Analytics をインストールする前に、コンピューターが最低ソフトウェアおよびハードウェア要件を満たしていることを確認します。

ソフトウェア要件

> **メモ**
>
> 一部のソフトウェア前提条件は、コンピューターに存在しない場合に自動的にインストールされます。自動的にインストールされる前提条件の一覧については、オンラインドキュメントを参照してください。

要件	追加情報
次のうち、いずれかのオペレーティング システム: ◦ Microsoft Windows 10(64 ビット) ◦ Microsoft Windows 8.1(64 ビット) ◦ Microsoft Windows 7 Service Pack 1 (SP1) (32 ビットまたは 64 ビット)	ACL for Windows は Windows の 64 ビット バージョンで実行することができる 32 ビット アプリケーションです。 **メモ** Windows 7 に ACL for Windows をインストールするには、サービス パック 1 がインストールされている必要があります。ACL for Windows には Microsoft .Net 4.6 が必要ですが、SP1 の前の Windows 7 のバージョンにはインストールできません。 Windows XP はサポート対象のオペレーティング システムから外れました。
Microsoft Windows 8.1 を使用している場合 Windows 8.1 更新 KB2919355	**重要** Windows 8.1 更新 KB2919355 は Microsoft .NET Framework 4.6.x で必要であり、ACL for Windows 14 で必要です。 Windows 8.1 を使用し、.NET 4.6.x をインストールしてなく、更新 KB2919355 を実行していない場合は、ACL for Windows インストーラーが終了し、.NET 4.6.2 前提条件のインストール中にエラー メッセージが表示されます。 ACL for Windows インストールを続行する前に、更新 KB2919355 をダウンロードしてインストールする必要があります。 あるいは、ACL for Windows インストールを開始する前に更新 KB2919355 をインストールし、エラー メッセージを回避できます。 **注意** インストール処理中に特定の時点でコンピューターを再起動するように指示された場合は、すぐに再起動してください。**コンピューターの再起動のメッセージは無視しないでください。** 指示されたときにコンピューターを再起動しないと、.NET、他の前提条件、または ACL for Windows のインストールで問題が生じるおそれがあります。
R プログラム言語と統合する Analytics 関数を使用するには、次	インストールされているバージョンの R のビットはオペレーティング システムのビットと一致している必要があります。

要件	追加情報
のものをインストールし、構成する必要があります(32 ビット/64 ビット) 。 以下のバージョンの R はテスト済みであり、Analytics で動作します。 ◦ 3.3.2 ◦ 3.3.1 ◦ 3.2.5 ◦ 3.2.3 CRAN R (32 ビット/64 ビット) または Microsoft R (64 ビットのみ) を使用できます。 **メモ** その他のバージョンの R も同様に動作するはずです が、保証の限りではありません。 現時点では、R 3.5.0 以降は Analytics と連携することができません。	CRAN R パッケージの 1 つを使用している場合は、R バイナリフォルダーへのパスをご利用のコンピューターの PATH 環境変数に追加する必要があります。 例: ◦ `C:\Program Files\R\R-<version>\bin\i386` (32-bit) ◦ `C:\Program Files\R\R-<version>\bin\x64` (64-bit) **メモ** R 言語と Analytics 関数を統合して使用しない場合は、この言語をインストールする必要はありません。
Python プログラム言語と統合する Analytics 関数を使用するには、次のものをインストールし、構成する必要があります。 ◦ Python バージョン 3.3.x(32 ビット) ◦ PYTHONPATH 環境変数 ◦ ACLPYTHONDLL 環境変数	Python バージョン 3.5.x は完全にテストされてサポートされています。3.3.x や 3.6.x など別のバージョンを使用することもできますが、これらのバージョンでは 3.5.x と同じテスト結果やサポートを Analytics で保証することはできません。 Python をインストールする場合は、お使いのシステムでも実行できるよう Python を設定する必要があります。詳細については、"Python の Analytics 連携用設定" ページ 899を参照してください。 **メモ** Python 言語と Analytics 関数を統合して使用しない場合は、この言語をインストールする必要はありません。 Analytics のインストール ディレクトリに格納されている Python エンジンのローカル コピーは、Analytics の Python 関数と使用したり、一般的な Python で使用したりするためのものではありません。これらの用途には、Python の別のインスタンスをインストールする必要があります。
ACL Connector for Oracle を使用するには、次の項目をインストールする必要があります。 ◦ Oracle Instant Client 11g または 12c	◦ ACL Connector for Oracle を使用しない場合は、Oracle Instant Client をインストールする必要はありません。 ◦ Oracle Instant Client が何ビット版であるかと、お使いのオペレーティング システムが何ビット版であるかが一致する必要があります。32 ビット版の Instant Client を 64 ビット マシンにインストールしても、接続が失敗するためです。 ◦ コネクターを Analytics Exchange と併用して、AX サーバーの後に Oracle Instant Client をインストールする場合は、Analytics Exchange サービスを再起動してから、コネクターを使用する必要があります。

ハードウェア要件

> **メモ**
>
> 本番環境での最善のAnalyticsのパフォーマンスには、最小の仕様よりも多くのリソースを必要とする場合があります。

コンポーネント	最小	推奨
プロセッサ	1.8 GHz	
メモリ (RAM)	2 GB	◦ 64ビットのオペレーティングシステム：8 GB以上を推奨(特に大容量ファイルの並べ替えをする場合) ◦ 32ビットのオペレーティングシステム：4 GB(特に大容量ファイルの並べ替えをする場合)
ハード ディスク領域 (Analytics アプリケーション ファイル)	1.1 GB	
ハード ディスク領域 (ソフトウェア必須コンポーネント)	8 GB	
ハード ディスク領域 (データ記憶領域)		100 GB以上 Analytics アプリケーション ファイルと前提条件をインストールするために必要なハード ディスク領域の他に、コンピューターがデータの抽出、フラットファイル、および結果を保存するために使用される場合は、追加の大きい領域が必要です。

ACL for Windows のインストール

ACL for Windows のインストール手順に従い、Analytics のコピーをインストールまたはアップグレードします。インストールが完了したら、ライセンスをアクティブ化します。

> **メモ**
>
> Analytics をインストールまたはアップグレードする場合、インストールまたはアップグレード中に指定した Analytics 作業ディレクトリにある既存の Analytics サンプル データ ファイルは上書きされます。
>
> サンプル プロジェクトやデータファイルに対して行ったどのような変更も保持したい場合は、インストールまたはアップグレードを実行する前に、他の場所にファイルを保存するか、それらが含まれるフォルダーの名前を変更してください。関連付けられたコマンド ログ ファイルを維持したい場合は、ログと同じ操作を実行します。

インストーラーの取得

組織の Galvanize 窓口当者からインストーラーを取得します。お客様が窓口担当者の場合：

- **インターネット アクセス** - Launchpad のダウンロード ページ(http://www.aclgrc.com/) からインストーラーを取得します
- **インターネット アクセスがない** - Launchpad からインストーラーを取得できるように Galvanize の担当者に依頼します

インストーラーの展開

1. ACL for Windows インストール パッケージ(`ACLforWindows141.exe`) をダブルクリックします。
2. セキュリティ警告のダイアログ ボックスが表示された場合は、リストされた情報を確認して、 [**はい**]をクリックします。
3. インストールに使用したい言語を選択して、 [**OK**]をクリックします。
4. [**展開場所の設定**]ページで、 [**展開**]をクリックします。

   ファイルが展開された後、インストーラーが自動的に開始されます。

前提条件のインストール

必須コンポーネントをインストールするよう求められた場合は、 [**インストール**]をクリックします。

必須コンポーネントがインストールされた後、インストーラーは自動的に進みます。

ACL for Windows をインストールする

画面の手順に従い、ACL for Windows インストールを実行します。

[**ACL エディションの選択**]ページで、インストールするエディションを選択します。

- **非 Unicode**
- **Unicode**

Unicode と非 Unicode 版の詳細については、"Unicode および非 Unicode 版" ページ 901を参照してください。

ライセンスのアクティブ化

インターネットに接続している場合は、初めて ACL for Windows を起動し、組織にサインインするときにアクティブ化できます。ライセンスをオフラインでアクティブ化するには、Galvanize サポートまでお問い合わせください。

Analytics の開始

Analytics を起動するには、次のうち 1 つを実行します。

新しいからの Analytics プロジェクトを作成するには	[**作成する**]から、[**アナリティクス プロジェクト**]をクリックします
既存の Analytics プロジェクトを開くには	[**開く**]から、[**アナリティクス プロジェクト**]をクリックします
最近またはサンプルの Analytics プロジェクト(.acl) を開くには	[**最近のアナリティクス ファイル**]、または [**サンプル ファイル**]から、プロジェクト名をクリックします

Python の Analytics 連携用設定

Analytics で動作するように Python を設定するには、正しいバージョンの Python をインストールし、実行ファイルをシステム PATH 環境変数に追加する必要があります。ACLPYTHONDLL および PYTHONPATH システム環境変数を設定する必要もあります。

機能の仕組み

Python スクリプトを実行するには、Analytics では Python 実行可能ファイルを呼び出し、実行を指示するスクリプトを見つけられる必要があります。Analytics では、Python を探すには PATH 環境変数を使用し、スクリプトを探すには PYTHONPATH 環境変数を使用します。

Python バージョン 3.3.x(32 ビット) 以降のインストール

1. [Python ダウンロード ページ]から、Python 3.5 の最新バージョンをお使いのマシンにダウンロードします。
2. お使いのマシンでインストーラーをダブルクリックします。
3. インストーラーで、 **[Python *バージョン番号* の PATH への追加]**を選択します。
4. **[インストール]**をクリックし、画面の指示に従います。
5. Analytics から Python スクリプトを実行する前に、マシンを再起動します。

ACLPYTHONDLL および PYTHONPATH 環境変数の設定

1. お使いの Windows オペレーティング システムに、Python スクリプトを格納する 1 つまたは複数のフォルダーを作成します。

   例 -`C:\python_scripts`。
2. お使いの Windows オペレーティング システムから、 **[システムのプロパティ]**ダイアログ ボックスを開き、 **[環境変数]**をクリックします。
3. **[システム変数]**セクションで、 **[新規]**をクリックし、次の変数を入力します。

変数名	変数値	例
PYTHONPATH	Python スクリプトを格納するために作成したフォルダーへの絶対パス。複数のフォルダーのパスをセミコロンで区切ります。	C:\python_scripts;C:\dev;C:\tmp
ACLPYTHONDLLACL	Analytics と一緒に使用する、Python インストール フォルダー内の Python	.c:\python_install\python33.dll

変数名	変数値	例
	DLL ファイルの絶対パスとファイル名。 Python 3.3.x を使用する場合は、次の制限が適用されます。 ◦ ヨーロッパ向けのプラットフォームのパスで Unicode 文字がサポートされません。 ◦ アジア向けのプラットフォームのパスで拡張文字がサポートされません。 **メモ** Python により、この DLL はインストール フォルダーでなくシステム フォルダー(`c:\windows\system3-2\python33.dll`) に追加されます。システム フォルダーからインストール フォルダーにこの DLL をコピーし、変数値として使用することで、この DLL に Analytics がアクセスできるようにします。 また、インストール フォルダーに読み取り専用設定があれば、その解除も必要です。 この値を設定しない場合は、Analytics により、サポートされているデフォルトのバージョン、3.5.x DLL `python35.dll`	

が使用されます。

4. 変数を保存するには、 [OK]をクリックして、 [**システムのプロパティ**]ダイアログ ボックスで [OK]をクリックします。

Analytics の Python 関数における Python の使用

Analytics から、お使いの PYTHONPATH に存在するスクリプトの関数を呼び出すには、Analytics の Python 関数を使用します。

> **メモ**
>
> Python スクリプトに編集を行う場合は、Analytics プロジェクトでビューを更新し、最新バージョンの Python スクリプトを使用する必要があります。ビューを更新する最も簡単な方法は、操作中のテーブルを閉じてから、もう一度開くことです。

Unicode および非 Unicode 版

Unicode 版の Analytics 製品では、Unicode データが含まれるファイルを表示したり操作したりできます。Unicode とは、ほとんどの世界の言語をサポートする文字暗号化を行う業界標準の方法です。

非 Unicode 版または Unicode 版をインストールすべきですか?

Analytics には Unicode 版と非 Unicode 版があります。いずれのエディションも同じインストールパッケージに含まれ、インストール時にインストールするエディションを指定します。

Unicode データの表示または分析のための要件がある場合を除き、非 Unicode 版をインストールする必要があります。Unicode データは、Analytics の Unicode 版でのみ開くことができます。

グローバル情報システムのある環境で作業する、または複数の言語を含むデータを分析する場合に、Unicode データを見る可能性が高くなります。

Unicode 版が必要なとき

次のデータを表示または分析するには、Unicode 版をインストールする必要があります。

- アジア文字
- 非 Unicode、または従来の文字エンコーディングの組み合わせ

  たとえば、次の文字エンコーディングの 2 つ以上から成る言語の組み合わせ

  - ラテン 1(英語および西欧)
  - ラテン 2(中欧)
  - キリル文字
  - ギリシャ文字
  - アラビア文字

> **メモ**
>
> 中国語、日本語、ポーランド語の Analytics ユーザーインターフェイスを使用するための唯一の選択肢は、Unicode エディションをインストールすることです。Unicode エディションが必要な理由は、データの言語でなくユーザーインターフェイスの言語に関連しています。

アナリティクスの Unicode への変換

非 Unicode の既存のアナリティクスおよびスクリプトを Unicode 版の Analytics に移行する場合、これらは自動的に Unicode に変換されます。ただし、これらを Unicode データに適用した場合でもスクリプトのロジックが変わらないことを確認する必要があります。

Unicode とは

Unicode は、各文字を表すために 2 バイト以上を使用するテキストのエンコードの標準であり、すべての言語の文字が単一の文字セットに格納されています。Unicode 版の Galvanize 製品を使用すると、現在使用されている全言語で、Unicode でエンコードされたデータを含んでいるファイルやデータベースの表示および操作が行えます。

> **メモ**
>
> Analytics および AX Engine は、リトルエンディアン(LE) でエンコードされた Unicode データをサポートしています。これらの製品を使用して、ビッグエンディアン(BE) でエンコードされたデータを分析することはできません。

Unicode 版 Analytics Exchange への移行

- Unicode スクリプトの暗号化は現在サポートされていません
- Analytics プロジェクト ファイルとログ ファイルは Unicode データ(UTF-16 LE) としてエンコードされているため、非 Unicode 版 のAnalytics では使用できません
- Analytics で印刷イメージ ファイルまたは区切り文字付きファイルを定義するとき、ファイルに ASCII または EBCDIC でエンコードされたテキストが含まれている場合は、このデータを格納する Analytics テーブルのフィールドには、デフォルトで Unicode データ型が割り当てられます

アナリティクスに必要な変更

バイト数で値を指定するすべてのパラメーターの更新

非 Unicode 版 Analytics での文字の長さは 1 バイトです。Unicode データの場合、Unicode エディションでの文字の長さは 2 バイトです。非 Unicode 版 Analytics で、フィールド長または開始位置をバイトで指定するときには、バイト数は文字数と同じです。これは Unicode 版 Analytics における Unicode データには当てはまりません。

アナリティクスを Unicode 版 Analytics で使用できるように変換するには、バイト数でフィールドの長さや開始位置を指定したすべてのパラメーターの長さを調整する必要があります。たとえば、非 Unicode 版 Analytics で 7 の WID 値を指定する IMPORT DELIMITED コマンドの場合、Unicode 版 Analytics で同じ結果を出力するには、WID の値を 2 倍の 14 にする必要があります。

また、Unicode データの場合、フィールドの奇数の開始バイト位置とフィールド長の偶数のバイト数を指定します。偶数の開始位置または奇数の長さを指定すると、文字が正しく表示されない可能性があります。

IMPORT PRINT および IMPORT DELIMITED の全インスタンスを再作成する

IMPORT PRINT コマンドと IMPORT DELIMITED コマンドの全インスタンスを再作成する必要があります。これは、Unicode 版の Analytics でデータ定義ウィザードを使用してソースデータファイルをインポートし、AX Server にプロジェクトを再インポートすることによって行います。データ定義ウィザードを使用すると、すべての構文が有効であることが保証されます。

ZONED() および EBCDIC() 関数の全インスタンスの変更

次のように ZONED() および EBCDIC() 関数の全インスタンスを変更し、関数で返される ASCII 値が正しく Unicode データに変換されるようにする必要があります。

- **演算フィールド** -は、ZONED( ) や EBCDIC( ) インスタンスを BINTOSTR( ) 関数で囲みます
- **静的な式** -では ZONED( ) インスタンスを BINTOSTR( ) 関数で囲みます

```
BINTOSTR(ZONED(%result%, 5), 'A')
```

OPEN FORMAT コマンドのすべてのインスタンスを変更します

OPEN FORMAT コマンドのすべてのインスタンスを変更する必要があります。SKIP パラメーターを使用して、開いている Unicode ファイルの最初の 2 バイトをスキップする必要があります。これは、UTF-16 エンコードファイルの最初の 2 バイトはバイトオーダーマークとして予約されており、ファイル内のテキストとは異なるものであるため、必要となります。

非 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
```

変換されたアナリティクスの検証

Unicode 版のアナリティクスによって生成される結果が、非 Unicode 版のアナリティクスによって生成された結果とまったく同じになるかどうかを検証します。これを行う最良の方法は、Diff ツールを使用して、分析で生成され

たログファイルを比較することです。Diff ツールは、ファイル間のあらゆる相違点を識別します。

結果がまったく同じでなかった場合の処置

Unicode 版のアナリティクスを使用しても非 Unicode 版と同じ結果を生成できなかった場合には、スクリプトのログ出力をアナリティクスのステップごとに比較することで、問題の切り分けができる可能性があります。

Unicode 互換性チェック

Unicode 版にアップグレードする際は、スクリプトに追加したすべてのカスタム ロジックが、Unicode データに対して実行されたときに同じ結果を生じるかどうかを検証する必要があります。スクリプトが Unicode データに対して実行されたときに、スクリプトが影響を受ける可能性があることが予測できる領域があります。

ビットおよび文字関数

以下に挙げる関数はそれぞれ、バイト位置やバイト数に基づいて値を返します。非 Unicode 版で使用される 1 バイト文字の表現から、Unicode データに使用される 2 バイト文字エンコードへ移動しても、これらの関数が正しく使用されているかどうかをチェックする必要があります。

- ASCII( )
- BIT( )
- BYTE( )
- CHR( )
- DIGIT( )
- HEX( )
- MASK( )
- SHIFT( )

バイト長と文字長は等しくない

以下の関数について、データ内の文字数とバイト数が一対一対応でと見なす方法で使用されていないことを保証するために、これらの関数がスクリプトで使用されている方法を確認する必要があります。

文字とバイト間を一対一対応と見なすロジックがあるインスタンスを見つけた場合は、そのロジックが Unicode データで正常に動作するよう、つまり、1 文字を表すために 2 バイトを使用するように調整する必要があります。STRING(1000, 4) 内の 4 など、文字列関数のパラメーターとして指定された数値は文字数を表すため、このような関数を標準的な方法で使用しても問題は発生しません。

変換関数

- PACKED( )
- STRING( )
- UNSIGNED( )
- VALUE( )
- ZONED( )

文字列関数

- AT( )
- BLANKS( )
- INSERT( )
- LAST( )
- LENGTH( )
- REPEAT( )
- SUBSTRING( )

その他の関数

- FILESIZE( )
- LEADING( )
- OFFSET( )
- RECLEN( )

Unicode 固有の関数を代用する

Galvanize の Unicode 製品は、非 Unicode データと Unicode データ間の変換をサポートする、6 個の Unicode 固有の関数をサポートしています。Galvanize の Unicode 製品で使用できる関数は、次のとおりです。

- **BINTOSTR()** - 関数は、ZONED または EBCDIC データを対応する Unicode 文字列に変換します。これにより、ZONED または EBCDIC のデータとしてエンコードされた値を正しく表示できるようになります。
- **DHEX( )** - は、指定された Unicode のフィールド値に相当する 16 進数の値を返します。これは HTOU ( ) の逆関数です。
- **DBYTE( )** - は、レコード内の指定された位置にある 2 バイト文字に相当する Unicode 文字を返します。
- **DTOU( )** - は、指定されたロケール設定に基づいて、日付値を正しい Unicode 文字列表示に変換します。
- **HTOU( )** - は、指定された 16 進数の文字列に相当する Unicode 文字列を返します。これは DHEX ( ) の逆関数です。
- **UTOD( )** - は、ロケール固有の Unicode 文字列を Analytics の日付値に変換します。

AX Server でのR スクリプトの実行

分析アプリとともに外部Rスクリプトを関連ファイルとしてインポートし、そのRスクリプトを作成したアナリティクス内から呼び出すことで、サーバーでRスクリプト言語の統計分析機能を利用できます。Rスクリプトを実行できるようにAX Server 環境を準備するには、まずRをインストールし、次いでファイル拡張子ホワイトリストに `.r` 拡張子を追加します。

前提条件

AX Server でRスクリプトを実行するには、以下の手順を行う必要があります。

1. お使いのAX Server のコンピューターに、サポートされているバージョンのRスクリプト言語をインストールします。
2. AX Server 上のファイル拡張子ホワイトリストに `.r` 拡張子を追加します。
3. Analytics で、作業してAX Server にインポートするプロジェクトを作成します。

> **メモ**
>
> これらの前提作業を行う助けとして、Analytics Exchange 管理者に問い合わせたり以下を参照したりしてください。
>
> - AX Server の要件
> - ファイル拡張子のホワイトリスト

Analytics プロジェクトのディレクトリへのR スクリプトの追加

Analytics で Analytics プロジェクトを作成後、使用したいRスクリプトをコピーし、プロジェクト フォルダーに貼り付けます。これにより、スクリプトを、ローカルである Analytics でテストしてから Analytics Exchange にインポートできます。

R ファイルの例

以下のRファイルの例には、2つの文字列を連結し、スペース文字で結合された単一の文字列を返す小さなスクリプトが含まれています。これらの例は、Rでデータを分析する方法ではなく、AX Server でRスクリプトを実行する方法を示すことが目的です。

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))
```

Analytics スクリプトを作成する

Analytics プロジェクト内で、AX Server で実行するアナリティクスとして使用する新しいスクリプトを作成します。このスクリプトで以下のことを行います。

1. 1 つのレコードから成る一時テーブル、t_tmp を開きます。

   Analytics では EXTRACT コマンドを実行するためにテーブルを開く必要がありますが、この t_tmp テーブルはその目的のためにのみ使用されます。

2. EXTRACT コマンドを使って各 R スクリプトを実行し、結果をテーブルに書き込みます。

アナリティクス ヘッダーの追加

分析アプリをインポート後、AX Server で Analytics スクリプトを実行できるように、スクリプトの先頭に必要なアナリティクス ヘッダー タグを追加します。アナリティクス内から実行する R スクリプトには、FILE タグを追加する必要があります。

```
COMMENT
//ANALYTIC R 統合テスト
  AX Server での R 統合を検証
//DATA t_tmp
//FILE analysis_a.r
//FILE analysis_b.r
//RESULT TABLE results
END
```

スクリプト ロジックの追加

```
SET SAFETY OFF
DEL ALL OK
CLOSE PRIMARY SECONDARY

OPEN t_tmp

COM **** R スクリプトを実行し、結果をテーブルに書き込む
```

```
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
```

このアナリティクス スクリプトの全容

AX Server で実行するこのアナリティクスの全容は次のとおりです。

```
COMMENT
//ANALYTIC R 統合テスト
 AX Server でのR 統合を検証
//DATA t_tmp
//FILE analysis_a.r
//FILE analysis_b.r
//RESULT TABLE results
END

SET SAFETY OFF
DEL ALL OK
CLOSE PRIMARY SECONDARY

OPEN t_tmp

COM **** R スクリプトを実行し、結果をテーブルに書き込む
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
```

Analytics プロジェクトと関連 R ファイルのインポート

アナリティクス スクリプトを作成したら、以下の手順を実行します。

1. AX Client で、Analytics プロジェクトを格納するコレクションおよびフォルダーを作成します。
2. プロジェクトと R ファイルをインポートするには
   a. 作成したフォルダーを右クリックし、[**インポート**]を選択します。
   b. ローカルコンピューター上の Analytics プロジェクトに移動し、`.acl` プロジェクトファイルと `.r` R スクリプトを選択します。

      > **メモ**
      >
      > Analytics プロジェクトとプロジェクトフォルダーの R ファイルを AX Server にインポートするため、これらを、**Ctrl キーを押しながらクリック**して選択します。また、t_tmp テーブルのソースデータファイルもインポートする必要があります。

   c. [**開く**]をクリックします。

インポート後の Server Explorer の外観

- *コレクション名*
  - *フォルダー名*
    - **分析アプリ**
      - *ACL プロジェクト名*
        - *アナリティクス スクリプト名*
    - **データ**
      - t_tmp
    - **関連ファイル**
      - analysis_a.r
      - analysis_b.r

アナリティクスの実行

AX Client の **Server Explorer** 内でアナリティクスを右クリックして、[**実行**]をクリックします。アナリティクスの一部として R スクリプトが実行されるので、AX Web Client から results 結果テーブルにアクセスできるようになります。

結果

アナリティクスを実行後の Server Explorer の外観

- *コレクション名*
  - *フォルダー名*
    - **分析アプリ**
      - *ACL プロジェクト名*
        - *アナリティクス スクリプト名*
    - **データ**
      - 結果
    - **関連ファイル**
      - analysis_a.r
      - analysis_b.r

結果テーブル

- **値**
- テスト値
- 値テスト

AX Server での Python スクリプトの実行

Analytics Exchange 管理者に外部 Python スクリプトを AX Server の PYTHONPATH ディレクトリにアップロードするように依頼し、次に作成したアナリティクス内からこれらのスクリプトを呼び出して、サーバーで Python プログラミング言語のオブジェクト指向機能を利用します。Python スクリプトを実行できるように AX Server 環境を準備するには、まず Python をインストールし、次に PYTHONPATH 環境変数を設定します。

前提条件

AX Server で Python スクリプトを実行するには、以下の手順を行う必要があります。

1. お使いの AX Server のコンピューターに、サポートされているバージョンの Python スクリプト言語をインストールします。
2. AX Server で PYTHONPATH 環境変数を設定します。
3. Analytics で、作業して AX Server にインポートするプロジェクトを作成します。

> **メモ**
>
> これらの前提作業を行う助けとして、Analytics Exchange 管理者に問い合わせたり以下を参照したりしてください。
>
> - AX Server の要件
> - Python の AX Server 連携用設定

Python スクリプトの作成

Analytics で Analytics プロジェクトを作成したら、アナリティクス内から呼び出すことのできる Python スクリプトを作成します。

次に、アナリティクス内からこのスクリプトを呼び出す前に、Analytics Exchange 管理者にこのスクリプト ファイルを渡して、AX Server をホストするマシンの PYTHONPATH ディレクトリにアップロードしてもらいます。AX Server でアナリティクスを実行する際、Python の実行可能ファイルが PYTHONPATH ディレクトリ内でスクリプトを探すため、スクリプトがそこに存在している必要があります。

Python ファイルの例

次の Python ファイルの例には、数値をその数値のべき乗する lambda 式を使用した小さなスクリプトが含まれています。この例は、Python でデータを分析する方法ではなく、AX Server で Python スクリプトを実行する方法を示すことが目的です。

ファイル名：lambda_example.py

```
# myFunc は value1 を平方して、その平方した値を返します
myFunc = lambda value1: value1**2
```

Analytics スクリプトを作成する

Analytics プロジェクト内で、AX Server で実行するアナリティクスとして使用する新しいスクリプトを作成します。このスクリプトで以下のことを行います。

1. 1 つのレコードから成るシンプルなテーブル、py を開きます。

   Analytics では GROUP コマンドを実行するためにテーブルを開く必要がありますが、この py テーブルはその目的のためにのみ使用されます。

2. 10 回ループします。各ループでは、インクリメント カウンターを引数として渡し、Python スクリプトを実行して、出力を結果テーブルに抽出します。

アナリティクス ヘッダーの追加

分析アプリをインポート後、AX Server で Analytics スクリプトを実行できるように、スクリプトの先頭に必要なアナリティクス ヘッダー タグを追加します。

```
COMMENT
//ANALYTIC Python統合テスト
 AX Server でのPython統合を検証
//DATA py
//DATA results
//RESULT TABLE results
END
```

スクリプト ロジックの追加

```
SET SAFETY OFF
DEL 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
```

このアナリティクス スクリプトの全容

AX Server で実行するこのアナリティクスの全容は次のとおりです。

```
COMMENT
//ANALYTIC Python統合テスト
 AX Server でのPython統合を検証
//DATA py
//DATA results
//RESULT TABLE results
END

SET SAFETY OFF
DEL 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
```

Analytics プロジェクトをインポートする

アナリティクス スクリプトを作成したら、以下の手順を実行します。

1. AX Client で、Analytics プロジェクトを格納するコレクションおよびフォルダーを作成します。
2. プロジェクトをインポートするには
   a. 作成したフォルダーを右クリックし、[**インポート**]を選択します。
   b. ローカルコンピューター上の Analytics プロジェクトに移動し、`.acl` プロジェクト ファイルを選択して [**開く**]をクリックします。

> **メモ**
>
> Analytics プロジェクトとともに py テーブルをインポートするには、ソース データ ファイルをインポートしておく必要があります。

インポート後の Server Explorer の外観

- *コレクション名*
  - *フォルダー名*
    - **分析アプリ**
      - *ACL プロジェクト名*
        - *アナリティクス スクリプト名*
    - **データ**
      - py
    - **関連ファイル**

アナリティクスの実行

AX Client の **Server Explorer** 内でアナリティクスを右クリックして、[**実行**]をクリックします。アナリティクスの一部として Python スクリプトが実行されるので、AX Web Client から results 結果テーブルにアクセスできるようになります。

> **メモ**
>
> スクリプトを実行する際、Python の実行可能ファイルは AX Server をホストするマシンの PYTHONPATH ディレクトリ内でスクリプト ファイルを探します。Analytics Exchange 管理者がこのファイルを当該ディレクトリにアップロードしていない場合には、アナリティクスは失敗します。

結果

アナリティクスを実行後の Server Explorer の外観

- *コレクション名*
  - *フォルダー名*
    - **分析アプリ**
      - *ACL プロジェクト名*
        - *アナリティクス スクリプト名*
    - **データ**
      - py
      - 結果
    - **関連ファイル**

結果テーブル

- **結果値**
- 1
- 4
- 9
- 16
- 25
- 36
- 49
- 64
- 81
- 100

Analytic Engine エラー コード

次の表は、アナリティクスを起動するときに発生する可能性のあるエラー コードを示します。

Analytic Engine スタート アップ エラー

エラー コード	説明
202	システム エラーです。
203	評価期間が終了しました。
204	評価期間が終了しました。
205	アクティブ化に失敗しました。
206	セッションの最大数に達しました。
207	メモリの初期化に問題があります。
209	不明なスクリプト エラーです。
210	データベース プロファイルのパスワードがありません。
211	サーバー接続に失敗しました。
212	サポートされていないコマンドが検出されました。
213	ダイアログ ボックスがスクリプトによって生成されました。
256	AX Engine を開始できませんでした。

Analytic Engine エラー コード

エラー コード	説明
1000	初期設定ファイルが指定されていません。新しいデフォルトの初期設定ファイルが作成されました。
1001	初期設定ファイルに問題があります。新しいデフォルトの初期設定ファイルが作成されました。
1002	プロジェクトは以前のバージョンからアップグレードされました。コピーが拡張子 .old で保存されました。

エラーコード	説明
1003	プロジェクト ファイルを処理できませんでした。前回保存したプロジェクトが使用されます。
1004	プロジェクト ファイルが指定されていません。
1005	指定されたプロジェクト ファイルは存在しません。
1006	指定されたプロジェクト ファイルは読み取り専用です。
1007	指定されたプロジェクトは現在、別のアプリケーションで使用されています。
1008	指定された .old プロジェクト ファイルは使用できません。拡張子が .ACL のプロジェクト ファイルを指定する必要があります。
1009	指定されたプロジェクト ファイルは Analytics プロジェクト ファイルではありません。
1011	指定されたプロジェクト ファイルは、以前のバージョンとして保存することはできません。
1012	ログ ファイルを書き込み用に開けません。
1013	スクリプトが指定されていません。
1014	指定されたスクリプトは存在しません。
1015	Analytics ライセンスが見つからなかったか無効です。
1016	必要なライブラリ ファイル(.dll) が見つかりませんでした。
1017	不明なエラーが発生しました。

コマンド エラー

次の表は、ACLScript コマンドが無効であることが原因でアナリティクスが失敗した場合に返されるエラー コードを示します。返されるエラー コード番号によって、失敗したコマンドが特定されます。

エラーコード	コマンド
1	SAMPLE
2	EXTRACT
3	LIST
4	TOTAL

エラーコード	コマンド
5	DEFINE
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

エラーコード	コマンド
28	DELETE
29	RANDOM
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

エラーコード	コマンド
56	EJECT
58	LET
59	ACCUMULATE
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	IMPORTIMPORT
80	REFRESH
81	NOTIFY
82	CONNECT

エラーコード	コマンド
83	RETRIEVE
84	FIELDSHIFT
85	BENFORD
86	CROSSTAB
87	(未使用)
88	ESCAPE
89	NOTES
90	FUZZY DUPLICATE
91	EXECUTE

アナリティクス ジョブ処理エラー

エラーコード	エラー メッセージ
-10	アナリティクスの実行開始後に出力先の結果フォルダーが削除されたため、アナリティクスの結果を保存できませんでした。
-11	ジョブが停止されました。
-12	サーバーがシャットダウンしたため、停止されました。
-13	結果を作成できませんでした。
-16	サーバー プロパティの構成エラーのため、実行できませんでした。
-17	一意の名前の結果ディレクトリを作成できません。
-19	ジョブがスキップされました。
-20	結果テーブルを公開する準備ができませんでした。
-21	結果を *AX* Exception に公開できませんでした。
-22	公開に失敗しました。テーブル名が無効です。

エラー コード	エラー メッセージ
-23	公開に失敗しました。テーブルの 1 つ以上の列の名前が長すぎます。
-24	公開に失敗しました。Analytics テーブルのデータ セル内の値が無効です。
-25	公開に失敗しました。テーブルのフィールド内にサポートされないデータ型があります。
-26	公開に失敗しました。*AX* Exception サーバーに接続できませんでした。
-27	ジョブは実行されませんでした。ユーザーが削除された、またはユーザーにアクセス許可がありません。
-28	ジョブは実行されませんでした。予期しないエラーです。詳細については、サーバー ログおよび Analytics ログを調べてください。
-29	データ ファイルをコピーできませんでした。必要なデータ ファイルをジョブ ディレクトリにコピーできなかったため、アナリティクスは失敗しました。
-30	ジョブは実行されませんでした。分析のリンクが機能していません。
-31	公開に失敗しました。例外マッピング ファイルが見つかりませんでした。
-32	公開に失敗しました。例外マッピング ファイルを解析できませんでした。
-34	ジョブの結果を保存できませんでした。ジョブ フォルダーを保管するドライブ上に十分なスペースがあることと、データ ファイルがロックされていないことを確認します。

Analytics コマンドによって作成される変数

Analytics のダイアログボックスで情報を入力する、またはスクリプトを実行することで、特定のコマンドが実行されると、Analytics によってシステム変数が自動的に作成されます。これらの変数とそれに含まれる値は、後続の Analytics コマンドを処理する際に利用できます。

システム変数の値は、同じコマンドを再度実行すると、更新された値に置き換わります。

"Analytics システム変数" 見開きページ Analytics によって作成されるシステム変数の一覧。

> **メモ**
>
> システム変数およびそれに含まれる値は、現在の Analytics セッションの間のみ有効です。

変数の現在の値の表示

次の方法のいずれでも、Analytics プロジェクト内のシステム定義変数およびユーザー定義変数すべての現在値を表示することができます。

- **ナビゲーター**で [**変数**] タブを選択
- コマンドラインで「DISPLAY VARIABLES」と入力
- ツールバー上で、[**変数の表示** ] をクリック(まず、ツールバーにボタンを追加する必要があります)

第 2 と第 3 の方法は、変数を格納するために利用できる残りのメモリも表示されます。

連続番号が付けられたシステム変数

"Analytics システム変数" 見開きページのうち、*n* が含まれているシステム変数名の場合、コマンドをコマンドのグループの外部で実行するときには、*n* は常に 1 になります。たとえば、TOTAL1 のようになります。

複数のコマンドを実行するのにコマンドのグループを使用する場合、結果となるシステム変数には、その変数を生成したコマンドの行番号に基づいた番号が付けられます。グループ内の最初のコマンドの行番号は 2 となります。

例:

- たとえば、Total コマンドがグループで 3 番目のコマンドの場合、その結果は TOTAL4 という変数に格納されます。
- また、別のフィールドの合計を算出するために Total コマンドを使用しており、それがグループで 5 番目のコマンドに当たる場合には、結果は TOTAL6 という変数に格納されます。

Analytics システム変数

次のテーブルは、Analytics より作成されるシステム変数、それらの変数を作成するコマンド、およびそれらの変数に含まれる値の一覧を示しています。

> **メモ**
>
> 複数のフィールドに対して同時にSTATISTICS コマンドを実行する場合、システム変数には、指定した最初のフィールドの値が含まれます。

システム変数	コマンド	値
WRITE*n*	◦ テーブルを出力するコマンド ◦ 検証 ◦ 順番検査	◦ 出力テーブル内のレコード数 ◦ データの妥当性エラーの数(検証) ◦ 順番検査エラーの数(順番検査)
OUTPUTFOLDER	◦ Analytics テーブルを出力するコマンド	出力テーブルを格納する、**ナビゲーター**内の Analytics プロジェクトフォルダーへのパス。 これは DOS 形式のパスで、「**/フォルダー名/サブフォルダー名**」の形式を用います。最初のスラッシュ(/) は [**総覧**]タブのルート階層を表します。 **ヒント** 別の出力フォルダーを指定する、または出力フォルダーを新規作成する場合は、SET FOLDER コマンドを使用します。
COUNT*n*	◦ カウント ◦ 統計	集計されたレコード数。
ACL_Ver_Major	◦ バージョン表示 (Analytics のバージョン番号は、メジャー.マイナー.パッチという形式で表されます。)	現在実行している Analytics のメジャー バージョンです。
ACL_Ver_Minor		現在実行している Analytics のマイナー バージョンです。
ACL_Ver_Patch		現在実行している Analytics のパッチ バージョンです。
ACL_Ver_Type		現在実行している Analytics のバージョンです。 ◦ 値が '0' の場合：非 Unicode 版 ◦ 値が '1' の場合： Unicode 版
MLE*n*	◦ 評価	**金額単位サンプリング** 推定誤謬額(推定による虚偽表示) **レコード サンプリング** 最大誤謬率(計算される上限逸脱率)
UEL*n*		**金額単位サンプリング** 最大誤謬額(虚偽表示上限)
RETURN_CODE	◦ Execute コマンドの結果	EXECUTE コマンドを使用した外部アプリケーションまたはプロセス実

システム変数	コマンド	値
		行によって返されるコード(リターン コード)。 リターン コードは外部アプリケーションまたはプロセスによって生成される数値であり、Analytics に返されることで外部プロセスの結果を示します。Analytics はリターン コードを生成しません。 通常、リターン コードは数値で、特定の通知やエラー メッセージが割り当てられています。たとえば、リターン コード "0" は "処理が正常終了した" ことを示します。リターン コード "2" は "指定されたファイルが見つからない" ことを示します。 特定のリターン コードとその意味は外部アプリケーションまたはプロセスによって異なります。リターン コードは "エラー コード" や "終了コード" とも呼ばれ、それらの意味は、関連する外部アプリケーションのドキュメントに記載されている可能性があります。リターン コードの一覧はインターネット上でも見ることができます。 RETURN_CODE 変数は、EXECUTE コマンドが同期的に使用されているときは作成されますが、コマンドが非同期的に使用されているときは作成されません。
GAPDUP*n*	◦ Gaps/ギャップ ◦ Duplicates/重複 ◦ あいまい重複	ギャップ、重複、またはあいまい重複グループの合計数です。
SAMPINT*n*	◦ サイズ	必要なサンプリング間隔です。
SAMPSIZE*n*		必要なサンプル数です。
ABS*n*	◦ 統計	指定した最初のフィールドの値の絶対値です。
AVERAGE*n*		指定した最初のフィールドの値の平均値です。
HIGH*n*		指定した最初のフィールドの値のうち、5 番目に高い値です。 5 番目に高い値はデフォルト設定です。この設定を変更するには、[**統計**]ダイアログ ボックスで [**高値/低値の数**]オプションを使用します。 **メモ** Analytics で最も高い値が確認される際、重複値は除外されません。たとえば、降順の値リストが 100、100、99、98 の場合、3 番目に高い値は 98 でなく 99 になります。
LOW*n*		指定した最初のフィールドの値のうち、5 番目に低い値です。 5 番目に低い値はデフォルト設定です。この設定を変更するには、[**統計**]ダイアログ ボックスで [**高値/低値の数**]オプションを使用します。 **メモ** Analytics で最も低い値が確認される際、重複値は除外されません。たとえば、昇順の値リストが 1、1、2、3 の場合、3 番目に低い値は 3 でなく 2 になります。

システム変数	コマンド	値
MAX*n*		指定した最初のフィールドの値の最大値です。
MEDIAN*n*		指定した最初のフィールドの値の中央値です。
MIN*n*		指定した最初のフィールドの値の最小値です。
MODE*n*		指定した最初のフィールドの値のうち、最もよく使用している値です。
Q25*n*		指定した最初のフィールドの値のうち、最初の四分位数値(下位四分位数値)です。
Q75*n*		指定した最初のフィールドの値のうち、3番目の四分位数値(上位四分位数値)です。
RANGE*n*		指定した最初のフィールドの値の最大値と最小値の差です。
STDDEV*n*		指定した最初のフィールドの値の標準偏差です。
TOTAL*n*	◦ 合計 ◦ 統計	指定した最初のフィールドの値の総計です。

その他のシステム変数

以下の変数は、コマンドで生成されるのではなく、システムで生成されます。

- **AXRunByUser** - AX Serverで実行されているスクリプト内で使用できます。この変数には、そのようなスクリプトでアナリティクスを実行しているユーザーのユーザー名(形式:*<ドメイン>\<ユーザー名>*)が格納されます。
- **OUTPUTFOLDER** - 現在のAnalyticsプロジェクト出力フォルダーです。

予約キーワード

Analytics では、特定の目的のために特定のキーワードが予約されています。これら予約キーワードの値をフィールド名や変数名に使用することはできません。

予約キーワードに接尾辞を付ければ、それをフィールド名や変数名として使用することができます。たとえば、"Field" という名前は許可されませんが、"Field_1" や "Field_2" は許可されます。

> **メモ**
>
> また一部の例では、"Can"(CANCEL) 、"Form" (FORMAT)、"Rec"(RECORD) などの、予約キーワードの省略形を使用することもできません。

予約キーワード	Analytics の目的
ALL	既に定義されているすべてのフィールドを表します。
AND	論理演算子の AND を表します。
AS	表示名を出力フィールドまたは式に割り当てます。
AXRunByUser	AX Server でアナリティクス スクリプトを実行しているユーザーのユーザー名 (形式 : *<ドメイン>**<ユーザー名>*) を格納するシステム変数。
CANCEL	現在のコマンドをキャンセルします。
D	直前に指定した式またはフィールド名のソート順が降順であることを表します。
END	一連の入力を終了し、空白行のように振る舞います。
EXPR	デフォルトの出力フィールド名の前に付けます。
F	論理式の *False* 値を表します。
FIELD/FIELDS	EXPORT コマンド、EXTRACT コマンド、JOIN コマンド、SAMPLE コマンドの一部として使用します。
FORMAT	Analytics テーブル レイアウトの古い名前。 Analytics テーブルの名前としては使用できません。
IF	条件を指定します。
LINE	DEFINE COLUMN コマンドで、フィールドを指定された行数に分断するかどうかを指定します。
NODUPS	Analytics レポートで、空のフィールドの重複表示値を表示しません。
NOT	論理演算子 NOT を表します。
NOZEROS	数値フィールドまたはレポートで、ゼロ値を空白として印刷または表示します。

予約キーワード	Analytics の目的
ON	フィールド リスト名の前に置きます。
OR	論理演算子 OR を表します。
OTHER	SUMMARIZE コマンドの出力に含めるが、小計しないフィールドまたは式を示します。
PAGE	REPORT コマンドを使用して、改ページを作成します。
PICTURE	数値フィールドの形式を表します。
PRIMARY	特定のタイプの結合を指定します。
RECORD	入力レコード全体を現状のままで表します。
RECORD_LENGTH	レコード長を表します。この値はレコード処理が必要な操作で使用します。
SECONDARY	特定のタイプの結合を指定します。
SUPPRESS	数値の合計を出力しないようにします。
T	論理式の *True* 値を表します。
TAPE	Analytics でデータにアクセスするための古い方法を参照します。 Analytics テーブルの名前としては使用できません。
TO	コマンドで使用する出力先ファイルを指定します。
WIDTH	特定のフィールドまたは式に対してデフォルトで設定されている印刷幅を変更します。