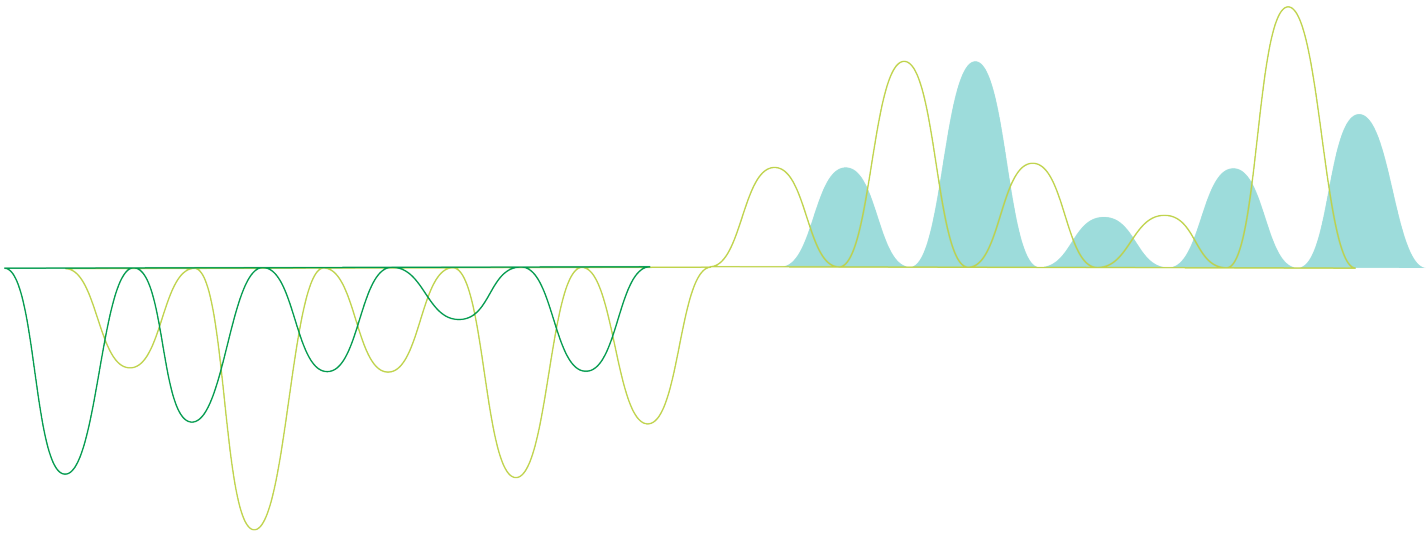
Синтаксис скрипта и функции диаграммы

Qlik Sense®
May 2022



Contents

1 Что такое Qlik Sense?

Программа Qlik Sense – это платформа для анализа данных. С помощью программы Qlik Sense можно самостоятельно анализировать и исследовать данные. Можно делиться полученными знаниями с другими людьми, анализировать данные в группах и во всей организации. Программа Qlik Sense дает возможность задавать себе вопросы и отвечать на них, самостоятельно идти по пути познания. Программа Qlik Sense позволяет вам с коллегами принимать решения в совместной работе.

1.1 Что можно сделать с помощью программы Qlik Sense?

Большинство продуктов бизнес-анализа (BI) могут помочь ответить на вопросы, изученные заранее. Но как ответить на вопросы, возникающие впоследствии? Вопросы, которые возникают после прочтения отчета или просмотра визуализации? Благодаря ассоциативной работе программы Qlik Sense вы сможете отвечать на вопрос за вопросом, двигаясь по собственному пути познания. С помощью программы Qlik Sense вы сможете легко, просто по щелчку, проводить свои исследования, на каждом шаге узнавая что-то новое, двигаясь дальше в изучении на основе полученных знаний.

1.2 Как работает программа Qlik Sense?

Программа Qlik Sense оперативно создает виды информации. Программе Qlik Sense не требуется заданных и статических отчетов, и вы не зависите от других пользователей – вы просто щелкаете кнопкой мыши и получаете информацию. При каждом щелчке кнопкой мыши программа Qlik Sense немедленно реагирует, обновляя каждую визуализацию Qlik Sense и вид в приложении новым рассчитанным набором данных и визуализаций, зависящим от выборок пользователя.

Модель приложения

Вместо развертывания и управления огромными бизнес-приложениями можно создать свои собственные приложения Qlik Sense, которые можно многократно использовать, изменять и совместно использовать с другими людьми. Модель приложения позволяет пользователю самому задавать себе вопросы и отвечать на них, нет необходимости обращаться к эксперту за отчетом или визуализацией.

Ассоциативная работа

Qlik Sense автоматически управляет всеми связями данных и представляет информацию пользователю с помощью схемы **green/white/gray**. Выборки подсвечиваются зеленым цветом, связанные данные представляются белым, а исключенные (несвязанные) данные отображаются серым цветом. Мгновенный ответ позволяет пользователям обдумывать новые вопросы и продолжать свое исследование.

Совместная работа и мобильность

Программа Qlik Sense позволяет пользователю осуществлять совместную работу с коллегами независимо от времени и места их нахождения. Также все функции Qlik Sense включая ассоциативную и совместную работу, доступны на мобильных устройствах. Программа Qlik Sense позволяет пользователям задавать вопросы и отвечать на них, а также рассматривать последующие вопросы, привлекая коллег, независимо от местоположения пользователя.

1.3 Как развернуть программу Qlik Sense?

Существует две версии Qlik Sense для развертывания: Qlik Sense Desktop и Qlik Sense Enterprise.

Qlik Sense Desktop

Это простая в установке версия для пользователя, которая обычно устанавливается на локальном компьютере.

Qlik Sense Enterprise

Эта версия используется для развертывания сайтов Qlik Sense. Сайт – это один или несколько сетевых компьютеров, подсоединенных к обычному логическому репозиторию или центральному узлу.

1.4 Как осуществлять контроль и управление сайтом Qlik Sense

С помощью консоли Qlik Management Console вы можете легко настраивать, контролировать сайты Qlik Sense и управлять ими. Можно управлять лицензиями, доступом и правилами безопасности, конфигурировать узлы и подключения к источникам данных, синхронизировать содержимое и пользователей и выполнять еще много различных действий.

1.5 Расширение возможностей Qlik Sense и адаптация под ваши требования

Приложение Qlik Sense располагает широким рядом средств API и SDK, которые позволяют расширять и настраивать приложение Qlik Sense для различных целей, таких как следующие.

Построение расширений и гибридных веб-приложений

Здесь можно выполнять веб-разработку с помощью JavaScript, чтобы построить расширения, которые являются пользовательскими визуализациями в приложениях Qlik Sense, использовать API гибридных веб-приложений для построения веб-сайтов с содержимым приложения Qlik Sense.

Построение клиентов

Можно построить клиенты в объектах .NET и встроить объекты Qlik Sense в собственные приложения. Также можно построить собственные клиенты на любом языке программирования, который поддерживает связь с WebSocket с помощью протокола клиента приложения Qlik Sense.

Построение инструментов сервера

С помощью API служебного и пользовательского каталога можно построить свой собственный инструмент для контроля и управления сайтами Qlik Sense.

Подключение к другим источникам данных

Создайте коннекторы программы Qlik Sense для получения данных из пользовательских источников данных.

2 Обзор синтаксиса скрипта

2.1 Введение в синтаксис скрипта

В скрипте определяются имя источника данных, имена таблиц и полей, входящих в логику. Более того, в нем указывают поля в определении прав доступа. Скрипт состоит из ряда последовательно выполняемых операторов.

Синтаксис командной строки Qlik Sense и синтаксис скриптов описываются в нотации, называемой формой Backus-Naur или кодом BNF.

Первые строки кода автоматически генерируются при создании нового файла Qlik Sense. Значения по умолчанию для этих переменных интерпретации чисел выводятся из региональных настроек ОС.

Скрипт состоит из ряда последовательно выполняемых операторов и ключевых слов. Все операторы скрипта должны заканчиваться точкой с запятой: «;».

Для преобразования загруженных данных можно использовать выражения и функции в операторах **LOAD**.

Табличный файл, в котором применяется разделитель в виде запятой, символа табуляции или точки с запятой, допускает использование оператора **LOAD**. По умолчанию оператор **LOAD** загружает все поля файла.

Доступ к общим базам данных можно получить с помощью коннекторов баз данных ODBC или OLE DB. Здесь используются стандартные операторы SQL. Принятый в операторе SQL синтаксис отличается в разных драйверах ODBC.

Кроме того, доступ к другим источникам данных можно получить с помощью пользовательских коннекторов.

2.2 Что такое форма Backus-Naur?

Синтаксис командной строки Qlik Sense и синтаксис скриптов описываются в нотации, называемой формой Backus-Naur, известной также как код BNF.

В следующей таблице представлен список символов, используемых в коде BNF, с описанием их интерпретации:

Символы

Символ	Описание
|	Логическая операция OR: символ можно использовать с любой стороны.
( )	Скобки очередности выполнения: используются для структурирования синтаксиса BNF.

Символ	Описание
[ ]	Квадратные скобки: заключенные в них элементы являются необязательными.
{ }	Фигурные скобки: заключенные в них элементы могут повторяться ноль и более раз.
Символ	Нетерминальная синтаксическая категория: может быть разделена на другие символы. Например на составляющие вышеуказанного, другие нетерминальные символы, текстовые строки и т. д.
::=	Отметка начала блока, определяющего символ.
LOAD	Терминальный символ, состоящий из текстовой строки. Записывается как есть в скрипт.

Все терминальные символы напечатаны шрифтом **bold face**. Например, «(» следует интерпретировать как скобки, определяющие порядок выполнения, а «**(**» следует интерпретировать как символ скрипта.

Пример:

Описание оператора alias:

```
alias fieldname as aliasname { , fieldname as aliasname}
```

Это следует интерпретировать как текстовую строку «alias», за которой следует произвольное имя поля, а потом текстовая строка «as» и произвольное имя псевдонима. Можно задать любое число дополнительных комбинаций «fieldname as alias», используя запятую в качестве разделителя.

Например, верными являются следующие операторы:

```
alias a as first;
alias a as first, b as second;
alias a as first, b as second, c as third;
```

Следующие операторы являются неверными:

```
alias a as first b as second;
alias a as first { , b as second };
```

2 Операторы и ключевые слова скрипта

Скрипт Qlik Sense состоит из ряда операторов. В качестве оператора может выступать обычный оператор скрипта или оператор управления скрипта. Перед некоторыми операторами могут стоять префиксы.

Как правило, обычные операторы используются для управления данными тем или иным образом. Эти операторы могут быть перезаписаны любым числом линий в скрипте и всегда должны заканчиваться точкой с запятой, «;».

Как правило, операторы управления используются для контроля хода выполнения скрипта. Каждое предложение оператора управления должно находиться внутри одной строки скрипта и может заканчиваться на точку с запятой или знак конца строки.

Префиксы можно использовать с соответствующими обычными операторами, но не с операторами управления. Тем не менее префиксы **when** и **unless** можно использовать в качестве суффиксов с некоторыми выражениями определенных операторов управления.

В следующем подразделе перечислены все существующие операторы скрипта, операторы управления и префиксы в алфавитном порядке.

Все ключевые слова скрипта можно вводить в любой комбинации символов в нижнем и верхнем регистре. В именах полей и переменных, используемых в операторах, учитывается регистр.

2.3 Операторы управления скриптом

Скрипт Qlik Sense состоит из ряда операторов. В качестве оператора может выступать обычный оператор скрипта или оператор управления скрипта.

Как правило, операторы управления используются для контроля хода выполнения скрипта. Каждое предложение оператора управления должно находиться внутри одной строки скрипта и может заканчиваться на точку с запятой или знак конца строки.

Операторы управления никогда не применяются с префиксами, за исключением префиксов **when** и **unless**, использование которых допускается с несколькими особыми операторами управления.

Все ключевые слова скрипта можно вводить в любой комбинации символов в нижнем и верхнем регистре.

Обзор операторов управления скриптом

Каждая функция подробно описана после обзора. Также можно щелкнуть имя функции в синтаксисе, чтобы получить немедленный доступ к подробной информации об этой конкретной функции.

Call

Оператор управления **call** вызывает подпрограмму, которую необходимо задать с помощью предыдущего оператора **sub**.

```
Call name ( [ paramlist ])
```

Do..loop

Оператор управления **do..loop** является компонентом итерации скрипта, который выполняет один или несколько операторов до выполнения логического условия.

```
Do..loop [ ( while | until ) condition ] [statements]
[exit do [ ( when | unless ) condition ] [statements]
loop [ ( while | until ) condition ]
```

Exit script

Этот оператор управления останавливает выполнение скрипта. Его можно вставить в любое место скрипта.

```
Exit script[ (when | unless) condition ]
```

For each ..next

Оператор управления **for each..next** является компонентом итерации скрипта, который выполняет один или несколько операторов для каждого значения в списке, разделенном запятой. Операторы внутри цикла, заключенного с помощью **for** и **next**, выполняются для каждого значения списка.

```
For each..next var in list
[statements]
[exit for [ ( when | unless ) condition ]
[statements]
next [var]
```

For..next

Оператор управления **for..next** представляет собой компонент итерации скрипта со счетчиком. Операторы внутри цикла, которые находятся между разделами **for** и **next**, будут выполняться для каждого значения переменной счетчика в пределах указанных минимального и максимального значений.

```
For..next counter = expr1 to expr2 [ stepexpr3 ]
[statements]
[exit for [ ( when | unless ) condition ]
[statements]
Next [counter]
```

If..then

Оператор управления **if..then** является компонентом выбора скрипта, который позволяет выполнять скрипт по различным путям в зависимости от одного или нескольких логических условий.

*Поскольку оператор **if..then** является оператором управления и заканчивается точкой с запятой или знаком конца строки, каждое из четырех его возможных предложений (**if..then**, **elseif..then**, **else** и **end if**) не должно выходить за границу строки.*

```
If..then..elseif..else..end if condition then
```

```
  [ statements ]
{ elseif condition then
  [ statements ] }
[ else
  [ statements ] ]
end if
```

Sub

Оператор управления **sub..end sub** определяет подпрограмму, которая должна вызываться оператором **call**.

```
Sub..end sub name [ ( paramlist )] statements end sub
```

Switch

Оператор управления **switch** является компонентом выбора скрипта, который позволяет выполнять скрипт по различным путям в зависимости от значения выражения.

```
Switch..case..default..end switch expression {case valuelist [ statements ]}
[default statements] end switch
```

Call

Оператор управления **call** вызывает подпрограмму, которую необходимо задать с помощью предыдущего оператора **sub**.

Синтаксис:

```
Call name ( [ paramlist ])
```

Аргументы:

Аргументы

Аргумент	Описание
name	Имя подпрограммы.
paramlist	Список фактических параметров, отправляемых в подпрограмму и перечисленных через запятую. Элементы списка могут быть именами полей, переменными или произвольными выражениями.

Подпрограмма, вызываемая оператором **call**, должна быть задана оператором **sub** ранее при выполнении скрипта.

Параметры копируются в подпрограмму и, если параметр оператора **call** является переменной, а не выражением, снова копируются назад при выходе из подпрограммы.

Ограничения:

- Поскольку оператор **call** является оператором управления и заканчивается точкой с запятой или знаком конца строки, он не должен выходить за границу строки.

- Когда определяется подпрограмма с использованием `Sub..end sub` внутри оператора управления, например `if..then`, подпрограмму можно вызвать только в пределах этого оператора управления.

Пример:

В данном примере все файлы, связанные с Qlik, показаны в папке и подпапках, данные о файлах приведены в таблице. Предполагается, что вы создали подключение к данным папки с именем Apps.

При вызове подпрограммы DoDir в качестве параметра используется ссылка на папку 'lib://Apps'. В рамках самой подпрограммы осуществляется рекурсивный вызов `Call DoDir (Dir)`, который запускает рекурсивный поиск функцией файлов в подпапках.

```
sub DoDir (Root)    For Each Ext in 'qvw', 'qvo', 'qvs', 'qvt', 'qvd', 'qvc', 'qvf'       For
Each File in filelist (Root&'\*.' &Ext)           LOAD              '$(File)' as Name,
      FileSize( '$(File)' ) as Size,               FileTime( '$(File)' ) as FileTime
autogenerate 1;       Next File    Next Ext       For Each Dir in dirlist (Root&'\*' )
Call DoDir (Dir)    Next Dir End Sub  Call DoDir ('lib://Apps')
```

Do..loop

Оператор управления **do..loop** является компонентом итерации скрипта, который выполняет один или несколько операторов до выполнения логического условия.

Синтаксис:

```
Do [ ( while | until ) condition ] [statements]
[exit do [ ( when | unless ) condition ] [statements]
loop[ ( while | until ) condition ]
```

*Поскольку оператор **do..loop** является оператором управления и заканчивается точкой с запятой или знаком конца строки, каждое из трех его возможных предложений (**do**, **exit do** и **loop**) не должно выходить за границу строки.*

Аргументы:

Аргументы

Аргумент	Описание
condition	Логическое выражение, имеющее значение True или False.
statements	Любая группа, состоящая из одного или нескольких операторов скрипта Qlik Sense.

Аргумент	Описание
while / until	Условное предложение **while** или **until** должно появиться только один раз в любом операторе **do..loop**, то есть после **do** или после **loop**. Каждое условие интерпретируется только при первом появлении, однако вычисляется при каждом появлении в цикле.
exit do	Если в цикле появляется предложение **exit do**, выполнение скрипта будет передано первому оператору после предложения **loop**, указывающего на конец цикла. Предложение **exit do** можно сделать условным с помощью дополнительного использования суффикса **when** или **unless**.

Пример:

```
// LOAD files file1.csv..file9.csv
Set a=1;
Do while a<10
LOAD * from file$(a).csv;
Let a=a+1;
Loop
```

End

Ключевое слово скрипта **End** используется, чтобы закрыть предложения **If**, **Sub** и **Switch**.

Exit

Ключевое слово скрипта **Exit** является частью оператора **Exit Script**, но также может использоваться для выхода из выражений **Do**, **For** или **Sub**.

Exit script

Этот оператор управления останавливает выполнение скрипта. Его можно вставить в любое место скрипта.

Синтаксис:

```
Exit Script [ (when | unless) condition ]
```

Поскольку оператор **exit script** является оператором управления и заканчивается точкой с запятой или знаком конца строки, он не должен выходить за границу строки.

Аргументы:

Аргументы

Аргумент	Описание
condition	Логическое выражение, имеющее значение True или False.

Аргумент	Описание
when / unless	Оператор **exit script** можно сделать условным с помощью дополнительного использования предложения **when** или **unless**.

Примеры:

```
//Exit script
Exit Script;

//Exit script when a condition is fulfilled
Exit Script when a=1
```

For..next

Оператор управления **for..next** представляет собой компонент итерации скрипта со счетчиком. Операторы внутри цикла, которые находятся между разделами **for** и **next**, будут выполняться для каждого значения переменной счетчика в пределах указанных минимального и максимального значений.

Синтаксис:

```
For counter = expr1 to expr2 [ step expr3 ]
[statements]
[exit for [ ( when | unless ) condition ]
[statements]
Next [counter]
```

Выражения *expr1*, *expr2* и *expr3* рассчитываются только при первом входе в цикл. Значение переменной counter может быть изменено операторами внутри цикла, однако это делать не рекомендуется.

Если в цикле появляется предложение **exit for**, выполнение скрипта будет передано первому оператору после предложения **next**, указывающего на конец цикла. Предложение **exit for** можно сделать условным с помощью дополнительного использования суффикса **when** или **unless**.

*Поскольку оператор **for..next** является оператором управления и заканчивается точкой с запятой или знаком конца строки, каждое из трех его возможных предложений (**for..to..step**, **exit for** и **next**) не должно выходить за границу строки.*

Аргументы:

Аргументы

Аргумент	Описание
counter	Имя переменной. Если переменная *counter* задана после **next**, она должна иметь такое же имя переменной, как указано после соответствующего предложения **for**.
expr1	Выражение, определяющее первое значение переменной *counter*, для которой должен выполняться цикл.
expr2	Выражение, определяющее последнее значение переменной *counter*, для которой должен выполняться цикл.
expr3	Выражение, которое определяет значение приращения переменной *counter* при каждом выполнении цикла.
condition	логическое выражение, имеющее значение True или False.
statements	Любая группа, состоящая из одного или нескольких операторов скрипта Qlik Sense.

Example 1: Загрузка последовательности файлов

```
// LOAD files file1.csv..file9.csv
for a=1 to 9

      LOAD * from file$(a).csv;

next
```

Example 2: Загрузка случайного числа файлов

В этом примере используются следующие файлы с данными: *x1.csv, x3.csv, x5.csv, x7.csv* и *x9.csv*. Загрузка остановлена в случайной точке с помощью условия `if rand( )<0.5 then`.

```
for counter=1 to 9 step 2

      set filename=x$(counter).csv;
      if rand( )<0.5 then

            exit for unless counter=1

      end if
      LOAD a,b from $(filename);

next
```

For each..next

Оператор управления **for each..next** является компонентом итерации скрипта, который выполняет один или несколько операторов для каждого значения в списке, разделенном запятой. Операторы внутри цикла, заключенного с помощью **for** и **next**, выполняются для каждого значения списка.

Синтаксис:
С помощью специального синтаксиса можно создавать списки с именами файлов и каталогов в текущем каталоге.

```
for each var in list
[statements]
[exit for [ ( when | unless ) condition ]
[statements]
next [var]
```

Аргументы:

Аргументы

Аргумент	Описание
var	Имя переменной скрипта, которое получает новое значение из списка для каждого выполнения цикла. Если переменная **var** задана после **next**, она должна иметь такое же имя переменной, как указано после соответствующего предложения **for each**.

Значение переменной **var** может быть изменено операторами внутри цикла, однако это делать не рекомендуется.

Если в цикле появляется предложение **exit for**, выполнение скрипта будет передано первому оператору после предложения **next**, указывающего на конец цикла. Предложение **exit for** можно сделать условным с помощью дополнительного использования суффикса **when** или **unless**.

*Поскольку оператор **for each..next** является оператором управления и заканчивается точкой с запятой или знаком конца строки, каждое из трех его возможных предложений (**for each**, **exit for** и **next**) не должно выходить за границу строки.*

Синтаксис:

```
list := item { , item }
item := constant | (expression) | filelist mask | dirlist mask |
fieldvaluelist mask
```

Аргументы

Аргумент	Описание
constant	Любое число или строка. Обратите внимание на то, что строка, непосредственно записываемая в скрипте, должна быть заключена в одинарные кавычки. Строка без одинарных кавычек будет интерпретироваться как переменная с использованием значения переменной. Числа не нужно заключать в одинарные кавычки.

Аргумент	Описание
expression	Произвольное выражение.
mask	Маска имени файла или папки, которая может включать в себя любые допустимые в имени файла символы и стандартные знаки подстановки, * и ?. Можно использовать абсолютные пути к файлу или пути lib://.
condition	Логическое выражение, имеющее значение True или False.
statements	Любая группа, состоящая из одного или нескольких операторов скрипта Qlik Sense.
filelist mask	Такой синтаксис создает разделенный запятыми список всех файлов в текущем каталоге, соответствующих маске имени файла. *Этот аргумент поддерживает только подключения к библиотеке в стандартном режиме.*
dirlist mask	Такой синтаксис создает разделенный запятыми список всех папок в текущей папке, соответствующей маске имени папки. *Этот аргумент поддерживает только подключения к библиотеке в стандартном режиме.*
fieldvaluelist mask	Этот синтаксис повторяется в значениях поля, которое уже загружено в Qlik Sense.

Маски фильтров, в которых используются знаки подстановки (и ?), не поддерживаются Qlik Коннекторы поставщиков веб-хранилищ и другими подключениями DataFiles.*

Example 1: Загрузка списка файлов

```
// LOAD the files 1.csv, 3.csv, 7.csv and xyz.csv for each a in 1,3,7,'xyz'    LOAD * from
file$(a).csv; next
```

Example 2: Создание списка файлов на диске

В этом примере показана загрузка всех файлов в папке, относящихся к программе Qlik Sense.

```
 sub DoDir (Root)    for each Ext in 'qvw', 'qva', 'qvo', 'qvs', 'qvc', 'qvf', 'qvd'
for each File in filelist (Root&'/*.' &Ext)           LOAD              '$(File)' as Name,
         FileSize( '$(File)' ) as Size,              FileTime( '$(File)' ) as FileTime
 autogenerate 1;        next File     next Ext    for each Dir in dirlist (Root&'/*' )
call DoDir (Dir)     next Dir  end sub  call DoDir ('lib://DataFiles')
```

Example 3: Повторяясь в значениях поля

Этот пример повторяется в списке загруженных значений элемента FIELD и создает новое поле NEWFIELD. Для каждого значения элемента FIELD необходимо создать две записи NEWFIELD.

```
 load * inline [ FIELD one two three ];  FOR Each a in FieldValueList('FIELD') LOAD '$(a)' &'-
'&RecNo() as NEWFIELD AutoGenerate 2; NEXT a
```

Полученная таблица выглядит следующим образом:

Example table

NEWFIELD
one-1
one-2
two-1
two-2
three-1
three-2

If..then..elseif..else..end if

Оператор управления **if..then** является компонентом выбора скрипта, который позволяет выполнять скрипт по различным путям в зависимости от одного или нескольких логических условий.

Как правило, операторы управления используются для контроля хода выполнения скрипта. Вместо этого в выражении диаграммы используйте условную функцию **if**.

Синтаксис:

```
If  condition then
  [ statements ]
{ elseif condition then
  [ statements ] }
[ else
  [ statements ] ]
end if
```

Поскольку оператор **if..then** является оператором управления и заканчивается точкой с запятой или знаком конца строки, каждое из четырех его возможных предложений (**if..then**, **elseif..then**, **else** и **end if**) не должно выходить за границу строки.

Аргументы:

Аргументы

Аргумент	Описание
condition	Логическое выражение, которое может иметь значение True или False.
statements	Любая группа, состоящая из одного или нескольких операторов скрипта Qlik Sense.

Example 1:

```
if a=1 then
      LOAD * from abc.csv;
      SQL SELECT e, f, g from tab1;
end if
```

Example 2:

```
if a=1 then; drop table xyz; end if;
```

Example 3:

```
if x>0 then
      LOAD * from pos.csv;
elseif x<0 then
      LOAD * from neg.csv;
else
      LOAD * from zero.txt;
end if
```

Next

Ключевое слово скрипта **Next** используется, чтобы закрыть циклы **For**.

Sub..end sub

Оператор управления **sub..end sub** определяет подпрограмму, которая должна вызываться оператором **call**.

Синтаксис:

```
Sub name [ ( paramlist )] statements end sub
```

Аргументы копируются в подпрограмму и снова копируются обратно при выходе из подпрограммы, если соответствующий фактический параметр в операторе **call** представляет собой имя переменной.

Если в подпрограмме присутствует больше формальных параметров, чем фактических параметров, передаваемых оператором **call**, то дополнительные параметры инициализируются со значением NULL, и их можно использовать в качестве локальных переменных в подпрограмме.

Аргументы:

Аргументы

Аргумент	Описание
name	Имя подпрограммы.
paramlist	Список имен переменных, разделенных запятой, для формальных параметров подпрограммы. Они могут использоваться как любая другая переменная в подпрограмме.
statements	Любая группа, состоящая из одного или нескольких операторов скрипта Qlik Sense.

Ограничения:

- Поскольку оператор **sub** является оператором управления и заканчивается точкой с запятой или знаком конца строки, каждое из двух его возможных предложений (**sub** и **end sub**) не должно выходить за границу строки.
- Когда определяется подпрограмма с использованием `Sub..end sub` внутри оператора управления, например `if..then`, подпрограмму можно вызвать только в пределах этого оператора управления.

Example 1:

```
Sub INCR (I,J)
I = I + 1
Exit Sub when I < 10
J = J + 1
End Sub
Call INCR (X,Y)
```

Example 2: – передача параметра

```
Sub ParTrans (A,B,C)
A=A+1
B=B+1
C=C+1
End Sub
A=1
X=1
C=1
Call ParTrans (A, (X+1)*2)
```

В результате этого локально внутри подпрограммы A будет инициализировано как 1, B как 4 и C как NULL.

При выходе из подпрограммы глобальная переменная A получает значение 2 (скопированное из подпрограммы). Второй фактический параметр (X+1)*2 не будет копироваться, поскольку не является переменной. Наконец, глобальная переменная C не будет изменена вследствие вызова подпрограммы.

Switch..case..default..end switch

Оператор управления **switch** является компонентом выбора скрипта, который позволяет выполнять скрипт по различным путям в зависимости от значения выражения.

Синтаксис:

```
Switch expression {case valuelist [ statements ]} [default statements] end switch
```

*Поскольку оператор **switch** является оператором управления и заканчивается точкой с запятой или знаком конца строки, каждое из четырех его возможных предложений (**switch**, **case**, **default** и **end switch**) не должно выходить за границу строки.*

Аргументы:

Аргументы

Аргумент	Описание
expression	Произвольное выражение.
valuelist	Список значений, разделенных запятой, с которыми будет сравниваться значение выражения. Выполнение скрипта продолжится с операторов в первой группе, в которой значение valuelist будет равно значению expression. Каждое значение valuelist может быть произвольным выражением. Если совпадение не найдено ни в одном из предложений **case**, то будут выполнены операторы в выражении **default** при их наличии.
statements	Любая группа, состоящая из одного или нескольких операторов скрипта Qlik Sense.

Пример:

```
Switch I
Case 1
LOAD '$(I): CASE 1' as case autogenerate 1;
Case 2
LOAD '$(I): CASE 2' as case autogenerate 1;
Default
LOAD '$(I): DEFAULT' as case autogenerate 1;
End Switch
```

To

Ключевое слово скрипта **To** используется в нескольких операторах скрипта.

2.4 Префиксы скрипта

Префиксы можно использовать с соответствующими обычными операторами, но не с операторами управления. Тем не менее префиксы **when** и **unless** можно использовать в качестве суффиксов с некоторыми выражениями определенных операторов управления.

Все ключевые слова скрипта можно вводить в любой комбинации символов в нижнем и верхнем регистре. В именах полей и переменных, используемых в операторах, учитывается регистр.

Обзор префиксов скрипта

Каждая функция подробно описана после обзора. Также можно щелкнуть имя функции в синтаксисе, чтобы получить немедленный доступ к подробной информации об этой конкретной функции.

Add

Префикс **Add** может быть добавлен к любому оператору **LOAD** или **SELECT** в скрипте для указания, что он должен добавлять записи в другую таблицу. Он также указывает, что этот оператор следует выполнять в частичной перезагрузке. Префикс **Add** может также использоваться в операторе **Map**.

```
Add [only] [Concatenate[(tablename )]] (loadstatement | selectstatement)
Add [ Only ] mapstatement
```

Buffer

Файлы QVD могут создаваться и обслуживаться автоматически посредством префикса **buffer**. Этот префикс может использоваться на большинстве операторов **LOAD** и **SELECT** в скрипте. Он указывает на то, что файлы QVD используются для кэширования/буферизации результата оператора.

```
Buffer[(option [ , option])] ( loadstatement | selectstatement )
option::= incremental | stale [after] amount [(days | hours)]
```

Concatenate

Если для двух таблиц необходимо выполнить объединение, и они имеют разные наборы полей, объединение двух таблиц может быть выполнено принудительно с помощью префикса **Concatenate**.

```
Concatenate[ (tablename ) ] ( loadstatement | selectstatement )
```

Crosstable

Префикс **crosstable** используется для преобразования перекрестной таблицы в прямую таблицу, что приводит к преобразованию широкой таблицы с множеством столбцов в длинную таблицу, в которой заголовки столбцов помещены в один столбец атрибутов.

```
Crosstable (attribute field name, data field name [ , n ] ) ( loadstatement |
selectstatement )
```

First

Префикс First операторов LOAD или SELECT (SQL) используется для загрузки заданного максимального числа записей из таблицы источника данных.

```
First n( loadstatement | selectstatement )
```

Generic

Распаковка и загрузка универсальной базы данных может выполняться с помощью префикса generic.

```
Generic ( loadstatement | selectstatement )
```

Hierarchy

Префикс hierarchy используется для преобразования иерархической таблицы в полезную таблицу модели данных Qlik Sense. Его можно поставить перед оператором LOAD или SELECT. Он будет использовать результат оператора загрузки в качестве ввода для преобразования таблицы.

```
Hierarchy (NodeID, ParentID, NodeName, [ParentName], [PathSource],
[PathName], [PathDelimiter], [Depth])(loadstatement | selectstatement)
```

HierarchBelongsTo

Префикс используется для преобразования иерархической таблицы в полезную таблицу модели данных Qlik Sense. Его можно поставить перед оператором LOAD или SELECT. Он будет использовать результат оператора загрузки в качестве ввода для преобразования таблицы.

```
HierarchyBelongsTo (NodeID, ParentID, NodeName, AncestorID, AncestorName,
[DepthDiff])(loadstatement | selectstatement)
```

Inner

Перед префиксами join и keep может стоять префикс inner. Если этот префикс используется перед join, то он указывает, что необходимо выполнить внутреннее объединение. Результирующая таблица, таким образом, будет содержать только комбинации значений полей из таблиц исходных данных с представлением связанных значений полей в обеих таблицах. Если этот префикс используется перед keep, он указывает, что обе таблицы с исходными данными следует уменьшить до области взаимного пересечения, прежде чем они смогут быть сохранены в программе Qlik Sense. .

```
Inner ( Join | Keep) [ (tablename) ](loadstatement |selectstatement )
```

IntervalMatch

Префикс IntervalMatch используется для создания таблиц сравнения дискретных числовых значений с одним или несколькими числовыми интервалами, а также сравнения значений с одним или несколькими дополнительными ключами.

```
IntervalMatch (matchfield)(loadstatement | selectstatement )
IntervalMatch (matchfield,keyfield1 [ , keyfield2, ... keyfield5 ] )
(loadstatement | selectstatement )
```

Join

Префикс **join** объединяет загруженную таблицу с существующей таблицей, для которой задано имя, или с последней созданной таблицей данных.

```
[Inner | Outer | Left | Right ] Join [ (tablename ) ]( loadstatement |
selectstatement )
```

Keep

Префикс **keep** подобен префиксу **join**. Также как префикс **join**, этот префикс сравнивает загруженную таблицу с существующей таблицей, для которой задано имя, или с последней созданной таблицей данных, но вместо объединения загруженной таблицы с существующей он позволяет сократить одну или обе таблицы до сохранения в программе Qlik Sense путем пересечения данных таблиц. Выполняемое сравнение аналогично натуральному объединению по всем общим полям, т. е. выполняется так же, как и при соответствующем объединении. Однако две таблицы не соединяются и сохраняются в программе Qlik Sense в виде двух отдельных таблиц с заданными именами.

```
(Inner | Left | Right) Keep [(tablename ) ]( loadstatement | selectstatement
)
```

Left

Перед префиксами **Join** и **Keep** может стоять префикс **left**.

Если этот префикс используется перед **join**, то он указывает, что необходимо выполнить левое объединение. Результирующая таблица будет содержать только комбинации значений полей из таблиц исходных данных с представлением связанных значений полей в первой таблице. Если этот префикс используется перед префиксом **keep**, он указывает, что вторую таблицу с исходными данными следует уменьшить до области взаимного пересечения с первой таблицей, прежде чем они смогут быть сохранены в программе Qlik Sense.

```
Left ( Join | Keep) [ (tablename) ](loadstatement |selectstatement )
```

Mapping

Префикс **mapping** используется для создания таблицы сопоставления, которую можно использовать, например, для замены значений полей и имен полей в ходе выполнения скрипта.

```
Сопоставление  ( loadstatement | selectstatement )
```

Merge

Префикс **Merge** может быть добавлен к любому оператору **LOAD** или **SELECT** в скрипте для указания, что загруженная таблица должна быть объединена с другой таблицей. Он также указывает, что этот оператор следует выполнять в частичной перезагрузке.

```
Merge [only] [(SequenceNoField [, SequenceNoVar])] On ListOfKeys [Concatenate
[(TableName)]] (loadstatement | selectstatement)
```

NoConcatenate

Префикс **NoConcatenate** определяет, что две загруженные таблицы с идентичными наборами полей будут обрабатываться как две отдельные внутренние таблицы вместо автоматического объединения.

```
NoConcatenate( loadstatement | selectstatement )
```

Outer

Для указания внешнего объединения перед явным префиксом **Join** может стоять префикс **Outer**. При внешнем объединении создаются все возможные комбинации двух таблиц. Результирующая таблица, таким образом, будет содержать комбинации значений полей из таблиц исходных данных с представлением связанных значений полей в одной или обеих таблицах. Ключевое слово **Outer** является дополнительным. Это тип объединения по умолчанию, которое используется, когда не указан префикс join.

```
Outer Join [ (tablename) ](loadstatement |selectstatement )
```

Partial reload

Полная перезагрузка всегда начинается с удаления всех таблиц в существующей модели данных, после чего выполняется скрипт загрузки. *Частичная перезагрузка (page 65)* этого не делает. Вместо этого все таблицы в модели данных сохраняются, и затем выполняются только операторы **Load** и **Select**, которым предшествует префикс **Add**, **Merge** или **Replace**. Другие таблицы данных не затрагиваются командой. Аргумент **only** обозначает, что оператор должен быть выполнен только во время частичных перезагрузок и должен быть проигнорирован во время полных перезагрузок. В следующей таблице подводится итог выполнения оператора для частичных или полных перезагрузок.

Replace

Префикс **Replace** может быть добавлен к любому оператору **LOAD** или **SELECT** в скрипте для указания, что загруженная таблица должна заменить другую таблицу. Он также указывает, что этот оператор следует выполнять в частичной перезагрузке. Префикс **Replace** может также использоваться в операторе **Map**.

```
Replace [only] [Concatenate[(tablename) ]] (loadstatement | selectstatement)
Replace [only] mapstatement
```

Right

Перед префиксами **Join** и **Keep** может стоять префикс **right**.

Если этот префикс используется перед **join**, то он указывает, что необходимо выполнить правое объединение. Результирующая таблица будет содержать только комбинации значений полей из таблиц исходных данных с представлением связанных значений полей во второй таблице. Если этот префикс используется перед префиксом **keep**, он указывает, что первую таблицу с исходными данными следует уменьшить до области взаимного пересечения со второй таблицей, прежде чем они смогут быть сохранены в программе Qlik Sense.

```
Right (Join | Keep) [(tablename)](loadstatement |selectstatement )
```

Sample

Префикс **sample** операторов **LOAD** или **SELECT** используется для загрузки произвольного образца записей из источника данных.

```
Sample p ( loadstatement | selectstatement )
```

Semantic

Таблицы, содержащие связи между записями, можно загрузить с помощью префикса **semantic**. Это могут быть, например, рекурсивные ссылки в пределах таблицы, где одна запись указывает на другую, такую как родительская, та, которой она принадлежит, или предшествующая.

```
Semantic ( loadstatement | selectstatement)
```

Unless

Префикс и суффикс **unless** используется для создания условного предложения, определяющего вычисление или невычисление оператора либо условия «exit». Это короткое утверждение можно использовать вместо полного оператора **if..end if**.

```
(Unless condition statement | exitstatement Unless condition )
```

When

Префикс и суффикс **when** используется для создания условного предложения, определяющего исполнение или неисполнение оператора либо условия «exit». Это короткое утверждение можно использовать вместо полного оператора **if..end if**.

```
( When condition statement | exitstatement when condition )
```

Add

Префикс **Add** может быть добавлен к любому оператору **LOAD** или **SELECT** в скрипте для указания, что он должен добавлять записи в другую таблицу. Он также указывает, что этот оператор следует выполнять в частичной перезагрузке. Префикс **Add** может также использоваться в операторе **Map**.

Чтобы частичная перезагрузка работала правильно, приложение должно быть открыто с данными до ее запуска.

Выполните частичную перезагрузку с помощью кнопки **Перезагрузить**. Можно также использовать Qlik Engine JSON API.

Синтаксис:

```
Add [only] [Concatenate[(tablename)]] (loadstatement | selectstatement)
```

```
Add [only] mapstatement
```

Во время обычной (не частичной) перезагрузки конструкция **Add LOAD** будет работать как обычный оператор **LOAD**. Записи будут создаваться и сохраняться в таблице.

Если будет использоваться префикс **Concatenate** или там будет существовать таблица с тем же набором полей, записи будут добавляться к соответствующей существующей таблице. Иначе конструкция **Add LOAD** создаст новую таблицу.

Частичная перезагрузка сделает то же самое. Единственная разница заключается в том, что конструкция **Add LOAD** никогда не будет создавать новую таблицу. Там всегда существует соответствующая таблица из предыдущего выполнения скрипта, в которую записи должны быть добавлены.

Проверка дубликатов не выполняется. Таким образом, оператор, использующий префикс **Add**, будет, как правило, включать в себя квалификатор distinct или защитные дубликаты предложения where.

Оператор **Add Map...Using** запускает сопоставление данных также и во время частичного выполнения скрипта.

Аргументы:

Аргументы

Аргумент	Описание
only	Дополнительный классификатор, указывающий на то, что оператор следует выполнять только во время частичных перезагрузок. Его следует игнорировать во время обычных (не частичных) перезагрузок.

Примеры и результаты:

Пример	Результат

`Tab1:`
 	
`LOAD Name, Number FROM Persons.csv;`
 	
`Add LOAD Name, Number FROM newPersons.csv;`
	Во время обычной перезагрузки данные загружаются из файла *Persons.csv* и сохраняются в таблице Qlik Sense Tab1. Затем данные из файла *NewPersons.csv* объединяются с той же таблицей Qlik Sense.
	
	Во время частичной перезагрузки данные загружаются из файла *NewPersons.csv* и добавляются в таблицу Qlik Sense Tab1. Проверка дубликатов не выполняется.

Пример	Результат
Tab1: SQL SELECT Name, Number FROM Persons.csv; Add LOAD Name, Number FROM NewPersons.csv where not exists(Name);	Проверка дубликатов выполняется путем проверки наличия Name в ранее загруженных табличных данных. Во время обычной перезагрузки данные загружаются из файла *Persons.csv* и сохраняются в таблице Qlik Sense Tab1. Затем данные из файла *NewPersons.csv* объединяются с той же таблицей Qlik Sense. Во время частичной перезагрузки данные загружаются из файла *NewPersons.csv*, который добавляется к таблице Qlik Sense Tab1. Проверка дубликатов выполняется путем проверки наличия Name в ранее загруженных табличных данных.
Tab1: LOAD Name, Number FROM Persons.csv; Add Only LOAD Name, Number FROM NewPersons.csv where not exists(Name);	Во время обычной перезагрузки данные загружаются из файла *Persons.csv* и сохраняются в таблице Qlik Sense Tab1. Оператор загрузки *NewPersons.csv* игнорируется. Во время частичной перезагрузки данные загружаются из файла *NewPersons.csv*, который добавляется к таблице Qlik Sense Tab1. Проверка дубликатов выполняется путем проверки наличия Name в ранее загруженных табличных данных.

Buffer

Файлы QVD могут создаваться и обслуживаться автоматически посредством префикса **buffer**. Этот префикс может использоваться на большинстве операторов **LOAD** и **SELECT** в скрипте. Он указывает на то, что файлы QVD используются для кэширования/буферизации результата оператора.

Синтаксис:

```
Buffer [(option [ , option])] ( loadstatement | selectstatement )
option::= incremental | stale [after] amount [(days | hours)]
```

Если не используется ни один параметр, буфер QVD, созданный при первом выполнении скрипта, будет использоваться в течение неопределенного времени.

Файл буфера находится в подпапке *Буферы*, обычно это *C:\ProgramData\Qlik\Sense\Engine\Buffers* (установка сервера) или *C:\Users\{user}\Documents\Qlik\Sense\Buffers* (Qlik Sense Desktop).

Имя файла QVD является вычисляемым именем (160-разрядными шестнадцатеричными случайными данными всего следующего оператора **LOAD** или **SELECT** и другой специфической информацией). Это означает, что буфер QVD будет недействительным при любых изменениях в следующем операторе **LOAD** или **SELECT**.

Обычно буферы QVD удаляются, если к ним больше не обращаются ни на каком этапе выполнения всего скрипта в приложении, его создавшем, либо в том случае, если приложение, его создавшее, уже не существует.

Аргументы:

Аргументы

Аргумент	Описание
incremental	Параметр incremental дает возможность прочитать только часть базового файла. Данные о предыдущем размере файла находятся в заголовке XML файла QVD. Это особенно полезно при работе с файлами журнала. Все записи, загруженные в предыдущий раз, считываются из файла QVD, в то время как последующие новые записи считываются из оригинального источника, в результате чего создается обновленный файл QVD. Параметр incremental можно использовать только вместе с предложениями **LOAD** и текстовыми файлами. Инкрементную нагрузку нельзя использовать в случае изменения или удаления старых данных.
stale [after] amount [(days | hours)]	amount – число, обозначающее период времени. Могут использоваться десятичные числа. Единицей измерения являются дни, если не указано. Параметр stale after обычно используется с источниками баз данных, если нет простой метки времени на оригинальных данных. Вместо этого можно указать, насколько старым может быть описание используемого снимка QVD. Предложение stale after просто указывает период времени с момента создания буфера QVD, после которого он будет считаться недействительным. До этого времени в качестве источника данных будет использоваться буфер QVD, а после этого – оригинальный источник данных. Файл буфера QVD будет автоматически обновлен, и начнется новый период.

Ограничения:

Среди многочисленных ограничений необходимо отметить наиболее важное, которое заключается в том, что в центре любого составного оператора должен быть оператор для файла **LOAD** либо **SELECT**.

Example 1:

```
Buffer SELECT * from MyTable;
```

Example 2:

```
Buffer (stale after 7 days) SELECT * from MyTable;
```

Example 3:

```
Buffer (incremental) LOAD * from MyLog.log;
```

Concatenate

Если для двух таблиц необходимо выполнить объединение, и они имеют разные наборы полей, объединение двух таблиц может быть выполнено принудительно с помощью префикса **Concatenate**. Этот оператор выполняет принудительное объединение с существующей именованной таблицей или последней созданной логической таблицей.

Синтаксис:

```
Concatenate[ (tablename ) ] ( loadstatement | selectstatement )
```

Объединение, по сути, совпадает с оператором **SQL UNION**, но с двумя отличиями:

- Префикс **Concatenate** может использоваться независимо от того, имеют ли таблицы идентичные имена полей.
- Идентичные записи при наличии префикса **Concatenate** не удаляются.

Аргументы:

Аргументы

Аргумент	Описание
tablename	Имя существующей таблицы.

Пример:

```
Concatenate LOAD * From file2.csv;
Concatenate SELECT * From table3;
tab1:
LOAD * From file1.csv;
tab2:
LOAD * From file2.csv;
.. .. ..
Concatenate (tab1) LOAD * From file3.csv;
```

Crosstable

Префикс **crosstable** используется для преобразования перекрестной таблицы в прямую таблицу, что приводит к преобразованию широкой таблицы с множеством столбцов в длинную таблицу, в которой заголовки столбцов помещены в один столбец атрибутов.

Синтаксис:

```
crosstable (attribute field name, data field name [ , n ] ) ( loadstatement |
selectstatement )
```

Аргументы:

Аргументы

Аргумент	Описание
attribute field name	Поле, которое содержит значения атрибутов.
data field name	Поле, которое содержит значения данных.
n	Число полей описателя перед таблицей, которые следует преобразовать в общий формат. По умолчанию задается 1.

Перекрестная таблица – это распространенный тип таблиц, включающих матрицу значений, расположенную между двумя и более ортогональными списками данных в заголовках, один из которых используется в качестве заголовков столбцов. Типичный пример – один столбец для каждого месяца. В результате использования префикса **crosstable** заголовки столбцов (например, названия месяцев) будут сохранены в одном поле (поле атрибутов), а данные столбцов (номера месяцев) будут сохранены во втором поле (поле данных).

Пример

Скрипт загрузки

Добавьте образец скрипта в свое приложение и запустите. Чтобы увидеть результаты, добавьте поля, перечисленные в столбце результатов, в лист приложения.

```
tmpData: //Crosstable (MonthText, Sales) Load * inline [ Product, Jan 2021, Feb 2021, Mar 2021, Apr 2021, May 2021, Jun 2021 A, 100, 98, 103, 63, 108, 82 B, 284, 279, 297, 305, 294, 292 C, 50, 53, 50, 54, 49, 51];   //Final: //Load Product, //Date(Date#(MonthText,'MMM YYYY'),'MMM YYYY') as Month, //Sales  //Resident tmpData;  //Drop Table tmpData;
```

Результат

Результирующая таблица

Продукт	Январь 2021 г.	Февраль 2021 г.	Март 2021 г.	Апрель 2021 г.	Май 2021 г.	Июнь 2021 г.
A	100	98	103	63	108	82
B	284	279	297	305	294	292
C	50	53	50	54	49	51

Объяснение

Этот пример демонстрирует перекрестную таблицу с одним столбцом для каждого месяца и с одной строкой для каждого продукта. Текущий формат усложняет анализ данных. Будет намного

лучше, если все числа будут в одном поле, а все месяцы в другом, то есть в таблице с тремя столбцами. А теперь давайте посмотрим, как преобразовать перекрестную таблицу таким образом.

Преобразование перекрестной таблицы

Раскомметрируйте скрипт и выполните его.

```
tmpData: Crosstable (MonthText, Sales) Load * inline [ Product, Jan 2021, Feb 2021, Mar 2021, Apr 2021, May 2021, Jun 2021 A, 100, 98, 103, 63, 108, 82 B, 284, 279, 297, 305, 294, 292 C, 50, 53, 50, 54, 49, 51];   Final: Load Product, Date(Date#(MonthText,'MMM YYYY'),'MMM YYYY') as Month, Sales  Resident tmpData;  Drop Table tmpData;
```

Результат

Результирующая таблица

Продукт	Месяц	Sales
A	Jan 2021	100
A	Feb 2021	98
A	Mar 2021	103
A	Apr 2021	63
A	May 2021	108
A	Jun 2021	82
B	Jan 2021	284
B	Feb 2021	279
B	Mar 2021	297
B	Apr 2021	305
B	May 2021	294
B	Jun 2021	292
C	Jan 2021	50
C	Feb 2021	53
C	Mar 2021	50
C	Apr 2021	54
C	May 2021	49
C	Jun 2021	51

Объяснение

Перекрестная таблица трансформируется в прямую таблицу, в которой один столбец будет для месяцев, а другой для объема продаж.

First

Префикс **First** операторов **LOAD** или **SELECT (SQL)** используется для загрузки заданного максимального числа записей из таблицы источника данных.

Синтаксис:

```
First n ( loadstatement | selectstatement )
```

Аргументы:

Аргументы

Аргумент	Описание
n	Произвольное выражение, результатом которого является целое число, обозначающее максимальное число считываемых записей. Элемент *n* может быть заключен в скобки, например *(n)*, но это необязательно.

Примеры:

```
First 10 LOAD * from abc.csv;
First (1) SQL SELECT * from Orders;
```

Generic

Распаковка и загрузка универсальной базы данных может выполняться с помощью префикса **generic**.

Универсальные базы данных/источники данных содержат структурированные повторяющиеся данные, например список адресов или спецификация продукта, где встречаются повторные описания объектов и используются подобные атрибуты.

Синтаксис:

```
Generic( loadstatement | selectstatement )
```

Примеры:

```
Generic LOAD * from abc.csv;
Generic SQL SELECT * from table1;
```

Таблицы, загружаемые с помощью оператора **generic**, автоматически не объединяются.

Пример

Пример 1.

Скрипт загрузки

Добавьте образец скрипта в свое приложение и запустите. Чтобы увидеть результаты, добавьте поля, перечисленные в столбце результатов, в лист приложения.

```
GenericDB:
Generic Load *;
Load * inline [
Region, Attribute, Value

US, Name, AAA
US, Address, A123
US, Phone, 001-123

US, Name, BBB
US, Address, B456
US, Phone, 002-456

SWE, Name, CCC
SWE, Address, C7789
SWE, Phone, 003-789 ];
```

Результат

Результирующая таблица

Регион	Имя	Address	Phone
SWE	CCC	C7789	003-789
US	AAA	A123	001-123
US	AAA	A123	002-456
US	AAA	B456	001-123
US	AAA	B456	002-456
US	BBB	A123	001-123
US	BBB	A123	002-456
US	BBB	B456	001-123
US	BBB	B456	002-456

Пример 2.

Скрипт загрузки

Добавьте образец скрипта в свое приложение и запустите. Чтобы увидеть результаты, добавьте поля, перечисленные в столбце результатов, в лист приложения.

```
Sheet1:
Generic Load * inline [
object, attribute, value
ball, color, red
ball, diameter, 10 cm
ball, weight, 100 g
box, color, black
box, height, 16 cm
box, length, 20 cm
```

```
box, weight, 500 g
box, width, 10 cm ];
```

Результат

Результирующая таблица

object	color	diameter	length	height	width	weight
ball	red	10 cm	-	-	-	100 g
box	black	-	20 cm	16 cm	10 cm	500 g

Hierarchy

Префикс **hierarchy** используется для преобразования иерархической таблицы в полезную таблицу модели данных Qlik Sense. Его можно поставить перед оператором **LOAD** или **SELECT**. Он будет использовать результат оператора загрузки в качестве ввода для преобразования таблицы.

Префикс создает таблицу развернутых узлов, которая, как правило, включает то же количество записей, что и входная таблица, но при этом каждый уровень иерархии сохраняется в отдельном поле. В иерархической структуре можно использовать поле пути.

Синтаксис:

```
Hierarchy (NodeID, ParentID, NodeName, [ParentName, [PathSource, [PathName,
[PathDelimiter, Depth]]]])(loadstatement | selectstatement)
```

В качестве входной таблицы должна использоваться таблица со смежными узлами. Таблицы со смежными узлами – таблицы, где каждая запись соответствует узлу и имеет поле, содержащее ссылку на родительский узел. В таких таблицах узел хранится в одной записи, но может иметь любое число дочерних узлов. В таблице могут содержаться дополнительные поля, описывающие атрибуты для узлов.

Префикс создает таблицу развернутых узлов, которая, как правило, включает то же количество записей, что и входная таблица, но при этом каждый уровень иерархии сохраняется в отдельном поле. В иерархической структуре можно использовать поле пути.

Обычно входная таблица имеет точно одну запись на узел, и в таком случае выходная таблица будет содержать такое же число записей. Однако иногда существуют узлы с несколькими родительским узлами, то есть один узел представлен несколькими записями во входной таблице. В таком случае в выходной таблице может содержаться больше записей, чем во входной.

Все узлы с родительским идентификатором, не найденные в столбце идентификаторов узлов (включая узлы с отсутствующими родительскими идентификаторами), будут расцениваться как корневые. К тому же загружаться будут только узлы с соединением с корневым узлом, прямым или косвенным, что тем самым позволит избежать циклических ссылок.

Можно создать дополнительные поля, содержащие имя родительского узла, путь узла и глубину узла.

Аргументы:

Аргументы

Аргумент	Описание
NodeID	Имя поля, содержащего идентификатор узла. Это поле должно существовать во входной таблице.
ParentID	Имя поля, содержащего идентификатор родительского узла. Это поле должно существовать во входной таблице.
NodeName	Имя поля, содержащего имя узла. Это поле должно существовать во входной таблице.
ParentName	Строка, которая используется для наименования нового поля **ParentName**. При его отсутствии это поле не создается.
ParentSource	Имя поля, которое содержит имя узла, используемого для создания пути к узлу. Дополнительный параметр. Если не указано, используется **NodeName**.
PathName	Строка, которая используется для наименования нового поля **Path**, содержащего путь от корневого каталога к узлу. Дополнительный параметр. При его отсутствии это поле не создается.
PathDelimiter	Строка, которая используется в качестве разделителя в новом поле **Path**. Дополнительный параметр. При его отсутствии используется «/».
Depth	Строка, которая используется для наименования нового поля **Depth**, содержащего глубину узла в иерархии. Дополнительный параметр. При его отсутствии это поле не создается.

Пример:

```
Hierarchy(NodeID, ParentID, NodeName, ParentName, NodeName, PathName, '\', Depth) LOAD *
inline [
NodeID, ParentID, NodeName
1, 4, London
2, 3, Munich
3, 5, Germany
4, 5, UK
5, , Europe
];
```

NodeID	ParentID	NodeName	NodeName1	NodeName2	NodeName3	ParentName	PathName	Depth
1	4	London	Europe	UK	London	UK	Europe\UK\London	3
2	3	Munich	Europe	Germany	Munich	Germany	Europe\Germany\Munich	3
3	5	German	Europe	Germany	-	Europe	Europe\Germany	2

		y						
4	5	UK	Europe	UK	-	Europe	Europe\UK	2
5		Europe	Europe	-	-	-	Europe	1

HierarchyBelongsTo

Префикс используется для преобразования иерархической таблицы в полезную таблицу модели данных Qlik Sense. Его можно поставить перед оператором **LOAD** или **SELECT**. Он будет использовать результат оператора загрузки в качестве ввода для преобразования таблицы.

Префикс позволяет создавать таблицу, которая содержит все связи родительский-дочерний элемент иерархии. Родительские поля затем могут использоваться для выбора целых деревьев в иерархии. Выходная таблица, как правило, включает несколько записей на каждый узел.

Синтаксис:

```
HierarchyBelongsTo (NodeID, ParentID, NodeName, AncestorID, AncestorName,
[DepthDiff])(loadstatement | selectstatement)
```

В качестве входной таблицы должна использоваться таблица со смежными узлами. Таблицы со смежными узлами – таблицы, где каждая запись соответствует узлу и имеет поле, содержащее ссылку на родительский узел. В таких таблицах узел хранится в одной записи, но может иметь любое число дочерних узлов. В таблице могут содержаться дополнительные поля, описывающие атрибуты для узлов.

Префикс позволяет создавать таблицу, которая содержит все связи родительский-дочерний элемент иерархии. Родительские поля затем могут использоваться для выбора целых деревьев в иерархии. Выходная таблица, как правило, включает несколько записей на каждый узел.

Можно создать дополнительные поля, содержащие разницу глубины узлов.

Аргументы:

Аргументы

Аргумент	Описание
NodeID	Имя поля, содержащего идентификатор узла. Это поле должно существовать во входной таблице.
ParentID	Имя поля, содержащего идентификатор родительского узла. Это поле должно существовать во входной таблице.
NodeName	Имя поля, содержащего имя узла. Это поле должно существовать во входной таблице.
AncestorID	Строка имени нового поля идентификатора родительского узла, которое содержит идентификатор родительского узла.

Аргумент	Описание
AncestorName	Строка имени нового поля родительского узла, которое содержит имя родительского узла.
DepthDiff	Строка имени нового поля **DepthDiff**, содержащего глубину узла в иерархии по отношению к родительскому узлу. Дополнительный параметр. При его отсутствии это поле не создается.

Пример:

```
HierarchyBelongsTo (NodeID, AncestorID, NodeName, AncestorID, AncestorName, DepthDiff) LOAD *
inline [
NodeID, AncestorID, NodeName
1, 4, London
2, 3, Munich
3, 5, Germany
4, 5, UK
5, , Europe
];
```

Results

NodeID	AncestorID	NodeName	AncestorName	DepthDiff
1	1	London	London	0
1	4	London	UK	1
1	5	London	Europe	2
2	2	Munich	Munich	0
2	3	Munich	Germany	1
2	5	Munich	Europe	2
3	3	Germany	Germany	0
3	5	Germany	Europe	1
4	4	UK	UK	0
4	5	UK	Europe	1
5	5	Europe	Europe	0

Inner

Перед префиксами **join** и **keep** может стоять префикс **inner**. Если этот префикс используется перед **join**, то он указывает, что необходимо выполнить внутреннее объединение. Результирующая таблица, таким образом, будет содержать только комбинации значений полей из таблиц исходных данных с представлением связанных значений полей в обеих таблицах. Если этот префикс используется перед **keep**, он

указывает, что обе таблицы с исходными данными следует уменьшить до области взаимного пересечения, прежде чем они смогут быть сохранены в программе Qlik Sense.

Синтаксис:

```
Inner ( Join | Keep) [ (tablename) ](loadstatement |selectstatement )
```

Аргументы:

Аргументы

Аргумент	Описание
tablename	Будет выполнено сравнение именованной таблицы с загруженной таблицей.
loadstatementили selectstatement	Оператор **LOAD** или **SELECT** для загруженной таблицы.

Пример

Скрипт загрузки

Добавьте образец скрипта в свое приложение и запустите. Чтобы увидеть результаты, добавьте поля, перечисленные в столбце результатов, в лист приложения.

```
Table1: Load * inline [ Column1, Column2 A, B 1, aa 2, cc 3, ee ];  Table2: Inner Join Load * inline [ Column1, Column3 A, C 1, xx 4, yy ];
```

Результат

Результирующая таблица

Column1	Column2	Column3
A	B	C
1	aa	xx

Объяснение

Этот пример демонстрирует вывод оператора внутреннего соединения Inner Join, который объединяет только значения, присутствующие и в первой (левой) и во второй (правой) таблицах.

IntervalMatch

Префикс **IntervalMatch** используется для создания таблиц сравнения дискретных числовых значений с одним или несколькими числовыми интервалами, а также сравнения значений с одним или несколькими дополнительными ключами.

Синтаксис:

```
IntervalMatch (matchfield)(loadstatement | selectstatement )
IntervalMatch (matchfield,keyfield1 [ , keyfield2, ... keyfield5 ] )
(loadstatement | selectstatement )
```

Префикс **IntervalMatch** устанавливается перед оператором **LOAD** или **SELECT**, который загружает интервалы. До оператора с префиксом **IntervalMatch** поле, которое содержит дискретные точки диаграммы (Time в приведенном ниже примере) и дополнительные ключи, уже должно быть загружено в Qlik Sense. Данный префикс не считывает это поле из таблицы базы данных сам по себе. Он преобразует загруженную таблицу интервалов и ключей в таблицу, содержащую дополнительный столбец дискретных числовых точек диаграммы. Здесь также разворачиваются различные записи, что позволяет включать в новую таблицу одну запись на возможную комбинацию дискретных точек диаграммы, интервалов и значений ключевых полей.

Интервалы могут накладываться друг на друга, а дискретные значения будут связаны со всеми соответствующими интервалами.

После расширения префикса IntervalMatch с помощью ключевых полей он используется для создания таблиц связывания дискретных числовых значений с одним или несколькими числовыми интервалами, а также связывания значений с одним или несколькими дополнительными ключами.

Во избежание игнорирования неопределенных границ интервалов может потребоваться разрешить сопоставление значений NULL с другими полями, которые образуют нижнюю или верхнюю границы интервала. Это выполняется с помощью оператора **NullAsValue** или явного теста, который заменяет значения NULL числовыми значениями, расположенными на достаточном расстоянии перед или после дискретных числовых точек диаграммы.

Аргументы:

Аргументы

Аргумент	Описание
matchfield	Поле, содержащее дискретные числовые значения, которые нужно связать с интервалами.
keyfield	Поля, содержащие дополнительные атрибуты, сопоставляемые при преобразовании.
loadstatement orselectstatement	Должна быть получена таблица, где первое поле содержит нижнюю границу каждого интервала, второе поле содержит верхнюю границу каждого интервала, а в случае использования сопоставления ключей третье и все последующие поля содержат ключевые поля, присутствующие в операторе **IntervalMatch**. Интервалы всегда закрытые, т. е. конечные точки включены в интервал. Нечисловые границы приводят к тому, что интервал игнорируется (неопределенный).

Example 1:

Рассмотрим две таблицы. В первой таблице приведено количество дискретных событий, а во второй – время начала и конца выполнения различных заказов. С помощью префикса **IntervalMatch** возможно логически связать две таблицы для того, чтобы узнать, например, на какие заказы повлияли нарушения в работе, а также какие заказы были обработаны в какие смены.

```
EventLog:
LOAD * Inline [
Time, Event, Comment
00:00, 0, Start of shift 1
01:18, 1, Line stop
02:23, 2, Line restart 50%
04:15, 3, Line speed 100%
08:00, 4, Start of shift 2
11:43, 5, End of production
];

OrderLog:
LOAD * INLINE [
Start, End, Order
01:00, 03:35, A
02:30, 07:58, B
03:04, 10:27, C
07:23, 11:43, D
];

//Link the field Time to the time intervals defined by the fields Start and End.
Inner Join IntervalMatch ( Time )
LOAD Start, End
Resident OrderLog;
```

Теперь таблица **OrderLog** содержит дополнительный столбец: *Time*. Число записей также увеличивается.

Table with additional column

Time	Start	End	Order
00:00	-	-	-
01:18	01:00	03:35	A
02:23	01:00	03:35	A
04:15	02:30	07:58	B
04:15	03:04	10:27	C
08:00	03:04	10:27	C
08:00	07:23	11:43	D
11:43	07:23	11:43	D

Example 2: (с помощью префикса keyfield)

Пример аналогичен приведенному выше: в качестве ключевого поля добавляется *ProductionLine* .

```
EventLog:
LOAD * Inline [
Time, Event, Comment, ProductionLine
00:00, 0, Start of shift 1, P1
01:00, 0, Start of shift 1, P2
```

```
01:18, 1, Line stop, P1
02:23, 2, Line restart 50%, P1
04:15, 3, Line speed 100%, P1
08:00, 4, Start of shift 2, P1
09:00, 4, Start of shift 2, P2
11:43, 5, End of production, P1
11:43, 5, End of production, P2
];

OrderLog:
LOAD * INLINE [
Start, End, Order, ProductionLine
01:00, 03:35, A, P1
02:30, 07:58, B, P1
03:04, 10:27, C, P1
07:23, 11:43, D, P2
];

//Link the field Time to the time intervals defined by the fields Start and End and match the
values
// to the key ProductionLine.
Inner Join
IntervalMatch ( Time, ProductionLine )
LOAD Start, End, ProductionLine
Resident OrderLog;
```

Теперь простую таблицу можно создать следующим образом:

Tablebox example

ProductionLine	Time	Event	Comment	Order	Start	End
P1	00:00	0	Start of shift 1	-	-	-
P2	01:00	0	Start of shift 1	-	-	-
P1	01:18	1	Line stop	A	01:00	03:35
P1	02:23	2	Line restart 50%	A	01:00	03:35
P1	04:15	3	Line speed 100%	B	02:30	07:58
P1	04:15	3	Line speed 100%	C	03:04	10:27
P1	08:00	4	Start of shift 2	C	03:04	10:27
P2	09:00	4	Start of shift 2	D	07:23	11:43
P1	11:43	5	End of production	-	-	-
P2	11:43	5	End of production	D	07:23	11:43

Join

Префикс **join** объединяет загруженную таблицу с существующей таблицей, для которой задано имя, или с последней созданной таблицей данных.

Синтаксис:

```
[inner | outer | left | right ]Join [ (tablename ) ]( loadstatement | selectstatement )
```

Join – это естественное объединение, образуемое по всем общим полям. Перед оператором join можно задать один из префиксов **inner**, **outer**, **left** или **right**.

Аргументы:

Аргументы

Аргумент	Описание
tablename	Будет выполнено сравнение именованной таблицы с загруженной таблицей.
loadstatementили selectstatement	Оператор **LOAD** или **SELECT** для загруженной таблицы.

Пример:

```
Join SELECT * from table1;



tab1:

LOAD * from file1.csv;

tab2:

LOAD * from file2.csv;

.. .. ..

join (tab1) LOAD * from file3.csv;
```

Пример

Скрипт загрузки

Добавьте образец скрипта в свое приложение и запустите. Чтобы увидеть результаты, добавьте поля, перечисленные в столбце результатов, в лист приложения.

```
Table1: Load * inline [ Column1, Column2 A, B 1, aa 2, cc 3, ee ];  Table2: Join Load * inline [ Column1, Column3 A, C 1, xx 4, yy ];
```

Результирующая таблица

Column1	Column2	Column3
A	B	C
1	aa	xx

Column1	Column2	Column3
2	cc	-
3	ee	-
4	-	yy

Объяснение

В этом примере две таблицы (Table1 и Table2) объединяются в одну таблицу Table1. В подобных случаях часто используется префикс **join** для объединения нескольких таблиц в одну, чтобы выполнять агрегирование значений одной таблицы.

Keep

Префикс **keep** подобен префиксу **join**. Также как префикс **join**, этот префикс сравнивает загруженную таблицу с существующей таблицей, для которой задано имя, или с последней созданной таблицей данных, но вместо объединения загруженной таблицы с существующей он позволяет сократить одну или обе таблицы до сохранения в программе Qlik Sense путем пересечения данных таблиц. Выполняемое сравнение аналогично натуральному объединению по всем общим полям, т. е. выполняется так же, как и при соответствующем объединении. Однако две таблицы не соединяются и сохраняются в программе Qlik Sense в виде двух отдельных таблиц с заданными именами.

Синтаксис:

```
(inner | left | right) keep [(tablename ) ]( loadstatement | selectstatement
)
```

Перед префиксом **keep** следует задать один из префиксов **inner**, **left** или **right**.

Явный префикс **join** в языке скриптов в программе Qlik Sense выполняет полное объединение двух таблиц. В результате получается одна таблица. Во многих случаях такое объединение приводит к созданию очень больших таблиц. Одной из основных функций Qlik Sense является возможность связывания нескольких таблиц вместо их объединения, что позволяет значительно сократить использование памяти, повысить скорость обработки и гибкость. По этой причине явных объединений в скриптах Qlik Sense следует, как правило, избегать. Функция keep предназначена для сокращения числа случаев необходимого использования явных объединений.

Аргументы:

Аргументы

Аргумент	Описание
tablename	Будет выполнено сравнение именованной таблицы с загруженной таблицей.
loadstatementили selectstatement	Оператор **LOAD** или **SELECT** для загруженной таблицы.

Пример:

```
Inner Keep LOAD * from abc.csv;
Left Keep SELECT * from table1;
tab1:
LOAD * from file1.csv;
tab2:
LOAD * from file2.csv;
.. .. ..
Left Keep (tab1) LOAD * from file3.csv;
```

Left

Перед префиксами **Join** и **Keep** может стоять префикс **left**.

Если этот префикс используется перед **join**, то он указывает, что необходимо выполнить левое объединение. Результирующая таблица будет содержать только комбинации значений полей из таблиц исходных данных с представлением связанных значений полей в первой таблице. Если этот префикс используется перед префиксом **keep**, он указывает, что вторую таблицу с исходными данными следует уменьшить до области взаимного пересечения с первой таблицей, прежде чем они смогут быть сохранены в программе Qlik Sense.

Вы искали строковую функцию по этому же имени? См.: Left (page 842)

Синтаксис:

```
Left ( Join | Keep) [ (tablename) ](loadstatement | selectstatement)
```

Аргументы:

Аргументы

Аргумент	Описание
tablename	Будет выполнено сравнение именованной таблицы с загруженной таблицей.
loadstatementили selectstatement	Оператор **LOAD** или **SELECT** для загруженной таблицы.

Пример

Скрипт загрузки

Добавьте образец скрипта в свое приложение и запустите. Чтобы увидеть результаты, добавьте поля, перечисленные в столбце результатов, в лист приложения.

```
Table1: Load * inline [ Column1, Column2 A, B 1, aa 2, cc 3, ee ];  Table2: Left Join Load *
inline [ Column1, Column3 A, C 1, xx 4, yy ];
```

Результат

Результирующая таблица

Column1	Column2	Column3
A	B	C
1	aa	xx
2	cc	-
3	ee	-

Объяснение

Этот пример демонстрирует вывод оператора соединения левой части Left Join, который объединяет только значения, присутствующие в первой (левой) таблице.

Сопоставление

Префикс **mapping** используется для создания таблицы сопоставления, которую можно использовать, например, для замены значений полей и имен полей в ходе выполнения скрипта.

Синтаксис:

```
Mapping( loadstatement | selectstatement )
```

Префикс **mapping** можно поставить перед оператором **LOAD** или **SELECT**. Он будет сохранять результат оператора загрузки в качестве таблицы сопоставления. Сопоставление представляет собой эффективный способ замены значений полей во время выполнения скрипта, например замены значений «СШ», «С.Ш.» или «Америка» значением «США». Таблица сопоставления состоит из двух столбцов, первый из которых содержит значения, используемые для сравнения, а второй – желаемые значения для сопоставления. Таблицы сопоставления временно хранятся в памяти и автоматически удаляются после выполнения скрипта.

К содержанию таблицы сопоставления доступ осуществляется с помощью, например, оператора **Map … Using**, **Rename Field**, функции **Applymap()** или **Mapsubstring()**.

Пример:

В этом примере мы загружаем список продавцов с кодом страны, представляющим их страну проживания. Мы используем таблицу, соответствующую коду страны, для той страны, код которой будет заменен ее названием. В таблице сопоставления указаны только три страны, коды других стран указаны в параметре 'Rest of the world'.

```
// Load mapping table of country codes:
map1:
mapping LOAD *
Inline [
CCode, Country
Sw, Sweden
```

```
Dk, Denmark
No, Norway
] ;
// Load list of salesmen, mapping country code to country
// If the country code is not in the mapping table, put Rest of the world
Salespersons:
LOAD *,
ApplyMap('map1', CCode,'Rest of the world') As Country
Inline [
CCode, Salesperson
Sw, John
Sw, Mary
Sw, Per
Dk, Preben
Dk, Olle
No, Ole
Sf, Risttu] ;
// We don't need the CCode anymore
Drop Field 'CCode';
```

Полученная таблица выглядит следующим образом:

Mapping table

Salesperson	Country
John	Sweden
Mary	Sweden
Per	Sweden
Preben	Denmark
Olle	Denmark
Ole	Norway
Risttu	Rest of the world

Merge

Префикс **Merge** может быть добавлен к любому оператору **LOAD** или **SELECT** в скрипте для указания, что загруженная таблица должна быть объединена с другой таблицей. Он также указывает, что этот оператор следует выполнять в частичной перезагрузке.

Типичный вариант использования – вы загружаете журнал изменений и хотите использовать его для применения inserts, updates и deletes к существующей таблице.

Чтобы частичная перезагрузка работала правильно, приложение должно быть открыто с данными до ее запуска.

Выполните частичную перезагрузку с помощью кнопки **Перезагрузить**. Можно также использовать Qlik Engine JSON API.

Синтаксис:

```
Merge [only] [(SequenceNoField [, SequenceNoVar])] On ListOfKeys [Concatenate [(TableName)]] (loadstatement | selectstatement)
```

Аргументы:

Аргументы

Аргумент	Описание
only	Дополнительный классификатор, указывающий на то, что оператор следует выполнять только во время частичных перезагрузок. Этот оператор следует игнорировать во время обычных (не частичных) перезагрузок.
SequenceNoField	Имя поля, содержащего метку времени или порядковый номер, который определяет порядок операций.
SequenceNoVar	Имя переменной, которой присваивается максимальное значение для SequenceNoField в объединяемой таблице.
ListOfKeys	Список разделенных запятой имен полей, указывающих первичный ключ.
Operation	Первое поле оператора Load должно содержать операцию в виде текстовой строки: Insert, Update или Delete. Также принимаются i, u и d.

Общая функциональность

Во время обычной (не частичной) перезагрузки конструкция **Merge LOAD** работает как обычный оператор **Load**, но с дополнительной функциональностью для удаления устаревших записей и записей, отмеченных на удаление. Первое поле оператора **Load** должно содержать информацию об операции: Insert, Update или Delete.

Идентификатор каждой загруженной записи сравнивается с ранее загруженными записями, и только последняя запись (согласно порядковому номеру) будет сохранена. Если последняя запись отмечена как Delete, она не будет сохраняться.

Целевая таблица

Набор полей определяет, в какую таблицу будут вноситься изменения. Если таблица с таким же набором полей (кроме первого поля; операции) уже существует, это будет релевантная таблица для изменения. Также можно использовать префикс **Concatenate**, чтобы задать таблицу. Если целевая таблица не определена, результат конструкции **Merge LOAD** сохраняется в новой таблице.

Если используется префикс Concatenate, результирующая таблица содержит группу полей, соответствующую объединению существующей таблицы и входных данных для слияния. Таким образом, целевая таблица может получить больше полей, чем журнал изменений, используемый в качестве входных данных для слияния.

При частичной перезагрузке происходит то же самое, что и при полной. Единственное отличие заключается в том, что при частичной перезагрузке редко создается новая таблица. Если не использовано предложение **Only**, всегда существует целевая таблица с таким же набором полей от предыдущего выполнения скрипта.

Порядковый номер

Если загруженный журнал изменений является накапливаемым, то есть уже содержит ранее загруженные изменения, параметр SequenceNoVar может использоваться в предложении **Where** для ограничения объема входных данных. Тогда конструкция **Merge LOAD** может загружать только записи, в которых поле SequenceNoField больше, чем SequenceNoVar. После завершения **Merge LOAD** присваивает новое значение полю SequenceNoVar с максимальным значением в поле SequenceNoField.

Операции

Merge LOAD может иметь меньше полей, чем целевая таблица. При выполнении различных операций отсутствующие поля обрабатываются по-разному:

Insert (Вставка): поля, отсутствующие в **Merge LOAD**, но существующие в целевой таблице, получают значение NULL в целевой таблице.

Удалить: Отсутствующие поля не влияют на результат. релевантные записи удаляются в любом случае.

Update (Обновление): поля, перечисленные в **Merge LOAD**, обновляются в целевой таблице. Отсутствующие поля не изменяются. Это означает, что два следующих оператора не являются идентичными:

- Merge on Key Concatenate Load 'U' as Operation, Key, F1, Null() as F2 From ...;
- Merge on Key Concatenate Load 'U' as Operation, Key, F1, Null() as F2 From ...;

Первый оператор обновляет перечисленные записи и изменяет F2 на NULL. Второй оператор не изменяет F2, но оставляет прежние значения в целевой таблице.

Merge LOAD нельзя использовать в записях с подстановочными знаками, например в таблице Section Access, в которой все значения обозначены звездочкой.

Примеры

Пример 1. Простое слияние с указанной таблицей

В этом примере во встроенную таблицу с именем Persons загружаются три строки. Затем **Merge** изменяет таблицу следующим образом:

- Добавляет строку *Mary, 4*.
- Удаляет строку *Steven, 3*.
- Присваивает число *5* строке *Jake*.

Для переменной *LastChangeDate* устанавливается максимальное значение в столбце *ChangeDate* после выполнения **Merge**.

Скрипт загрузки

Добавьте образец скрипта в свое приложение и запустите. Чтобы увидеть результаты, добавьте поля, перечисленные в столбце результатов, в лист приложения.

```
Set DateFormat='D/M/YYYY'; Persons: load * inline [ Name, Number Jake, 3 Jill, 2 Steven, 3 ];
Merge (ChangeDate, LastChangeDate) on Name Concatenate(Persons) LOAD * inline [ Operation,
ChangeDate,   Name,     Number Insert,    1/1/2021,     Mary,     4 Delete,    1/1/2021,
Steven,  Update,    2/1/2021,     Jake,     5 ];
```

Результат

До загрузки **Merge Load** результирующая таблица выглядит следующим образом.

Resulting table

Name	Number
Jake	3
Jill	2
Steven	3

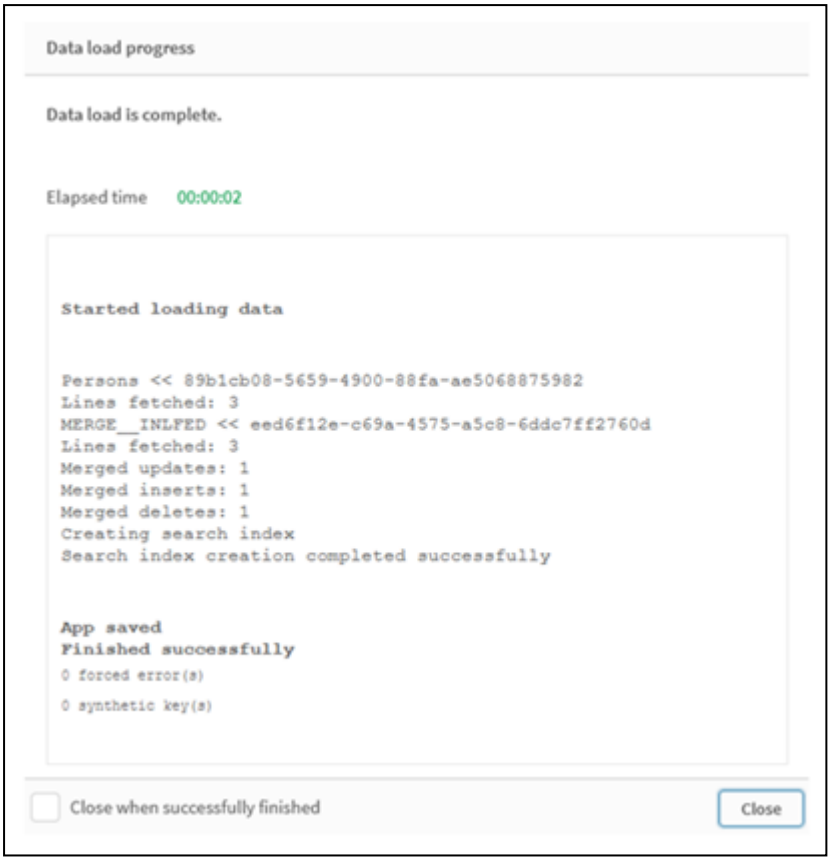
После выполнения **Merge Load** таблица выглядит следующим образом.

Resulting table

ChangeDate	Name	Number
2/1/2021	Jake	5
-	Jill	2
1/1/2021	Mary	4

Когда данные загружены, диалоговое окно **Выполнение загрузки данных** показывает выполняемые операции.

Диалоговое окно «Выполнение загрузки данных»



Пример 2. Скрипт загрузки данных с отсутствующими полями

В этом примере загружаются те же данные, что выше, но теперь с идентификатором для каждого человека.

Merge изменяет таблицу следующим образом:

- Добавляет строку *Mary, 4*.
- Удаляет строку *Steven, 3*.
- Присваивает число *5* строке *Jake*.
- Присваивает число *6* строке *Jill*.

Скрипт загрузки

Здесь используются два оператора **Merge Load**: один для операций Insert (Вставка) и Delete (Удаление), а второй для операции Update (Обновление).

Добавьте образец скрипта в свое приложение и запустите. Чтобы увидеть результаты, добавьте поля, перечисленные в столбце результатов, в лист приложения.

```
Set DateFormat='D/M/YYYY'; Persons: Load * Inline [ PersonID, Name, Number 1, Jake, 3 2, Jill,
2 3, Steven, 3 ];  Merge (ChangeDate, LastChangeDate) on PersonID Concatenate(Persons) Load *
Inline [ Operation, ChangeDate,   PersonID, Name,     Number Insert,    1/1/2021,     4,
 Mary,     4 Delete,    1/1/2021,     3,        Steven,  ];  Merge (ChangeDate,
LastChangeDate) on PersonID Concatenate(Persons) Load * Inline [ Operation, ChangeDate,
PersonID, Number Update,    2/1/2021,     1,        5 Update,    3/1/2021,     2,        6 ];
```

Результат

После выполнения операторов **Merge Load** таблица выглядит следующим образом.

Resulting table

PersonID	ChangeDate	Name	Number
1	2/1/2021	Jake	5
2	3/1/2021	Jill	6
4	1/1/2021	Mary	4

Обратите внимание, что второй оператор **Merge** не включает поле **Name** и, как следствие, имена остались без изменений.

Пример 3: Скрипт загрузки данных – частичная перезагрузка с использованием предложения Where с ChangeDate

В следующем примере аргумент **Only** указывает, что команда **Merge** выполняется только во время частичной перезагрузки. Обновления фильтруются на основе ранее полученной переменной LastChangeDate. После завершения выполнения **Merge** переменной LastChangeDate присваивается максимальное значение столбца ChangeDate, обработанное во время слияния

Скрипт загрузки

```
Merge Only (ChangeDate, LastChangeDate) on Name Concatenate(Persons) LOAD Operation,
ChangeDate,  Name, Number  from [lib://ChangeFilesFolder/BulkChangesInPersonsTable.csv] (txt)
where ChangeDate >= $(LastChangeDate);
```

NoConcatenate

Префикс **NoConcatenate** определяет, что две загруженные таблицы с идентичными наборами полей будут обрабатываться как две отдельные внутренние таблицы вместо автоматического объединения.

Синтаксис:

```
NoConcatenate( loadstatement | selectstatement )
```

Пример:

```
LOAD A,B from file1.csv;
NoConcatenate LOAD A,B from file2.csv;
```

Only

Ключевое слово скрипта **Only** используется в качестве функции агрегирования или как часть синтаксиса в префиксах частичной перезагрузки **Add**, **Replace** и **Merge**.

Outer

Для указания внешнего объединения перед явным префиксом **Join** может стоять префикс **Outer**. При внешнем объединении создаются все возможные комбинации двух таблиц. Результирующая таблица, таким образом, будет содержать комбинации значений полей из таблиц исходных данных с представлением связанных значений полей в одной или обеих таблицах. Ключевое слово **Outer** является дополнительным. Это тип объединения по умолчанию, которое используется, когда не указан префикс join.

Синтаксис:

```
Outer Join [ (tablename) ](loadstatement |selectstatement )
```

Аргументы:

Аргументы

Аргумент	Описание
tablename	Будет выполнено сравнение именованной таблицы с загруженной таблицей.
loadstatementили selectstatement	Оператор **LOAD** или **SELECT** для загруженной таблицы.

Пример

Скрипт загрузки

Добавьте образец скрипта в свое приложение и запустите. Чтобы увидеть результаты, добавьте поля, перечисленные в столбце результатов, в лист приложения.

```
Table1: Load * inline [ Column1, Column2 A, B 1, aa 2, cc 3, ee ];  Table2: Outer Join Load *
inline [ Column1, Column3 A, C 1, xx 4, yy ];
```

Результирующая таблица

Column1	Column2	Column3
A	B	C
1	aa	xx
2	cc	-
3	ee	-
4	-	yy

Объяснение

В этом примере две таблицы (Table1 и Table2) объединяются в одну таблицу Table1. В подобных случаях часто используется префикс **outer** для объединения нескольких таблиц в одну, чтобы выполнять агрегирование значений одной таблицы.

Частичная перезагрузка

Полная перезагрузка всегда начинается с удаления всех таблиц в существующей модели данных, после чего выполняется скрипт загрузки.

При частичной перезагрузке это не делается. Вместо этого все таблицы в модели данных сохраняются, и затем выполняются только операторы **Load** и **Select**, которым предшествует префикс **Add**, **Merge** или **Replace**. Другие таблицы данных не затрагиваются командой. Аргумент **only** обозначает, что оператор должен быть выполнен только во время частичных перезагрузок и должен быть проигнорирован во время полных перезагрузок. В следующей таблице подводится итог выполнения оператора для частичных или полных перезагрузок.

Оператор	**Полная перезагрузка**	**Частичная перезагрузка**
Load ...	Оператор будет выполнен	Оператор не будет выполнен
Add/Replace/Merge Load ...	Оператор будет выполнен	Оператор будет выполнен
Add/Replace/Merge Only Load ...	Оператор не будет выполнен	Оператор будет выполнен

Частичная перезагрузка обеспечивает несколько преимуществ по сравнению с полной перезагрузкой:

- Она быстрее, так как необходимо загружать только недавно измененные данные. При работе с большими наборами данных эта разница существенна.
- Потребляется меньше памяти, так как загружается меньше данных.
- Она более надежна, так как запросы исходных данных выполняются быстрее, сокращая риск возникновения проблем с сетью.

Чтобы частичная перезагрузка работала правильно, приложение должно быть открыто с данными до ее запуска.

Выполните частичную перезагрузку с помощью кнопки **Перезагрузить**. Можно также использовать Qlik Engine JSON API.

Пример

Пример 1.

Скрипт загрузки

Добавьте образец скрипта в свое приложение и выполните частичную перезагрузку. Чтобы увидеть результаты, добавьте поля, перечисленные в столбце результатов, в лист приложения.

```
T1: Add only Load distinct recno()+10 as Num autogenerate 10;
```

Результат

Resulting table

Num	Count(Num)
11	1
12	1
13	1
14	1
15	1
16	1
17	1
18	1
19	1
20	1

Объяснение

Оператор выполняется только при частичной перезагрузке. Если префикс distinct пропущен, количество в поле **Num** будет увеличиваться с каждой последующей частичной перезагрузкой.

Пример 2.

Скрипт загрузки

Добавьте образец скрипта в свое приложение. Выполните полную перезагрузку и посмотрите, что произойдет. Затем выполните частичную перезагрузку и оцените результат. Чтобы увидеть результаты, добавьте поля, перечисленные в столбце результатов, на лист приложения.

```
 T1: Load recno() as ID, recno() as Value autogenerate 10;  T1: Replace only Load recno() as ID, repeat(recno(),3) as Value autogenerate 10;
```

Результат

Output table after full reload

ID	Value
1	1
2	2
3	3
4	4
5	5

ID	Value
6	6
7	7
8	8
9	9
10	10

Output table after partial reload

ID	Value
1	111
2	222
3	333
4	444
5	555
6	666
7	777
8	888
9	999
10	101010

Объяснение

Первая таблица загружается при полной перезагрузке, а вторая таблица просто заменяет первую в процессе частичной перезагрузки.

Replace

Ключевое слово скрипта **Replace** используется в качестве строковой функции или как префикс в частичной перезагрузке.

Replace

Префикс **Replace** может быть добавлен к любому оператору **LOAD** или **SELECT** в скрипте для указания, что загруженная таблица должна заменить другую таблицу. Он также указывает, что этот оператор следует выполнять в частичной перезагрузке. Префикс **Replace** может также использоваться в операторе **Map**.

Чтобы частичная перезагрузка работала правильно, приложение должно быть открыто с данными до ее запуска.

Выполните частичную перезагрузку с помощью кнопки **Перезагрузить**. Можно также использовать Qlik Engine JSON API.

Синтаксис:

```
Replace [only] [Concatenate[(tablename)]] (loadstatement | selectstatement)
```

```
Replace [only] mapstatement
```

Во время обычной (не частичной) перезагрузки конструкция **Replace LOAD** будет работать как обычный оператор **LOAD**, но ей будет предшествовать **Drop Table**. Сначала старая таблица будет отброшена, затем записи будут созданы и сохранены как новая таблица.

Если используется префикс **Concatenate** или там будет существовать таблица с тем же набором полей, соответствующая таблица будет отброшена. Иначе никакая таблица не будет отброшена, и конструкция **Replace LOAD** будет идентична обычному **LOAD**.

Частичная перезагрузка сделает то же самое. Единственная разница – всегда есть таблица из предыдущего выполнения скрипта, которая будет отброшена. Конструкция **Replace LOAD** будет всегда сначала отбрасывать старую таблицу, а затем создавать новую.

Оператор **Replace Map...Using** запускает сопоставление данных также и во время частичного выполнения скрипта.

Аргументы:

Аргументы

Аргумент	Описание
only	Дополнительный классификатор, указывающий на то, что оператор следует выполнять только во время частичных перезагрузок. Его следует игнорировать во время обычных (не частичных) перезагрузок.

Примеры и результаты:

Пример	Результат
`Tab1: Replace LOAD * from File1.csv;`	Во время обычной и частичной перезагрузки изначально отбрасывается таблица Qlik Sense Tab1. После этого из файла File1.csv загружаются новые данные, которые сохраняются в таблице Tab1.

Пример	Результат
`Tab1: Replace only LOAD * from File1.csv;`	Во время обычной перезагрузки этот оператор игнорируется. Во время частичной перезагрузки изначально отбрасывается любая таблица Qlik Sense, которая раньше называлась Tab1. После этого из файла File1.csv загружаются новые данные, которые сохраняются в таблице Tab1.
`Tab1: LOAD a,b,c from File1.csv; Replace LOAD a,b,c from File2.csv;`	Во время обычной перезагрузки сначала считывается файл File1.csv в таблицу Qlik Sense Tab1, однако затем она сразу отбрасывается и заменяется новыми данными, загруженными из файла File2.csv. Все данные из файла File1.csv теряются. Во время частичной перезагрузки изначально отбрасывается вся таблица Qlik Sense Tab1. После этого она заменяется новыми данными, загруженными из файла File2.csv.
`Tab1: LOAD a,b,c from File1.csv; Replace only LOAD a,b,c from File2.csv;`	Во время обычной перезагрузки данные загружаются из файла File1.csv и сохраняются в таблице Qlik Sense Tab1. Файл File2.csv игнорируется. Во время частичной перезагрузки изначально отбрасывается вся таблица Qlik Sense Tab1. После этого она заменяется новыми данными, загруженными из файла File2.csv. Все данные из файла File1.csv теряются.

Right

Перед префиксами **Join** и **Keep** может стоять префикс **right**.

Если этот префикс используется перед **join**, то он указывает, что необходимо выполнить правое объединение. Результирующая таблица будет содержать только комбинации значений полей из таблиц исходных данных с представлением связанных значений полей во второй таблице. Если этот префикс используется перед префиксом **keep**, он указывает, что первую таблицу с исходными данными следует уменьшить до области взаимного пересечения со второй таблицей, прежде чем они смогут быть сохранены в программе Qlik Sense.

Вы искали строковую функцию по этому же имени? См.: Right (page 851)

Синтаксис:

```
Right (Join | Keep) [(tablename)](loadstatement |selectstatement )
```

Аргументы:

Аргументы

Аргумент	Описание
tablename	Будет выполнено сравнение именованной таблицы с загруженной таблицей.

Аргумент	Описание
loadstatementили selectstatement	Оператор **LOAD** или **SELECT** для загруженной таблицы.

Пример

Скрипт загрузки

Добавьте образец скрипта в свое приложение и запустите. Чтобы увидеть результаты, добавьте поля, перечисленные в столбце результатов, в лист приложения.

```
Table1: Load * inline [ Column1, Column2 A, B 1, aa 2, cc 3, ee ];  Table2: Right Join Load * inline [ Column1, Column3 A, C 1, xx 4, yy ];
```

Результат

Результирующая таблица

Column1	Column2	Column3
A	B	C
1	aa	xx
4	-	yy

Объяснение

Этот пример демонстрирует вывод оператора соединения правой части Right Join, который объединяет только значения, присутствующие во второй (правой) таблице.

Sample

Префикс **sample** операторов **LOAD** или **SELECT** используется для загрузки произвольного образца записей из источника данных.

Синтаксис:

```
Sample  p ( loadstatement | selectstatement )
```

Аргументы:

Аргументы

Аргумент	Описание
p	Произвольное выражение, которое определяет число больше 0 и меньше или равное 1. Число обозначает вероятность считывания определенной записи. Все записи будут считаны, но только некоторые из них будут загружены в программу Qlik Sense.

Пример:

```
Sample 0.15 SQL SELECT * from Longtable;
Sample(0.15) LOAD * from Longtab.csv;
```

Скобки допускаются, но необязательны.

Semantic

Таблицы, содержащие связи между записями, можно загрузить с помощью префикса **semantic**. Это могут быть, например, рекурсивные ссылки в пределах таблицы, где одна запись указывает на другую, такую как родительская, та, которой она принадлежит, или предшествующая.

Синтаксис:

```
Semantic( loadstatement | selectstatement)
```

При семантической загрузке создаются семантические поля, которые могут отображаться в фильтрах для использования при навигации в данных.

Таблицы, загруженные посредством оператора **semantic**, не могут быть объединены.

Пример:

```
Semantic LOAD * from abc.csv;
Semantic SELECT Object1, Relation, Object2, InverseRelation from table1;
```

Unless

Префикс и суффикс **unless** используется для создания условного предложения, определяющего вычисление или невычисление оператора либо условия «exit». Это короткое утверждение можно использовать вместо полного оператора **if..end if**.

Синтаксис:

```
(Unless condition statement | exitstatement Unless condition )
```

Действия **statement** или **exitstatement** выполняются, только если элемент **condition** имеет значение False.

Префикс **unless** можно использовать в операторах, включающих в себя один или несколько других операторов, в том числе дополнительные префиксы **when** или **unless**.

Аргументы:

Аргументы

Аргумент	Описание
condition	Логическое выражение, имеющее значение True или False.
statement	Любой оператор скрипта Qlik Sense, за исключением операторов управления.
exitstatement	Предложение **exit for**, **exit do** или **exit sub** или оператор **exit script**.

Примеры:

```
exit script unless A=1;
unless A=1 LOAD * from myfile.csv;
unless A=1 when B=2 drop table Tab1;
```

When

Префикс и суффикс **when** используется для создания условного предложения, определяющего исполнение или неисполнение оператора либо условия «exit». Это короткое утверждение можно использовать вместо полного оператора **if..end if**.

Синтаксис:

```
(when condition statement | exitstatement when condition )
```

Действия **statement** или **exitstatement** выполняются, только если условие имеет значение True.

Префикс **when** можно использовать в операторах, включающих в себя один или несколько других операторов, в том числе дополнительные префиксы **when** или **unless**.

Синтаксис:

Аргументы

Аргумент	Описание
condition	Логическое выражение, имеющее значение True или False.
statement	Любой оператор скрипта Qlik Sense, за исключением операторов управления.
exitstatement	Предложение **exit for**, **exit do** или **exit sub** или оператор **exit script**.

Example 1:

```
exit script when A=1;
```

Example 2:

```
when A=1 LOAD * from myfile.csv;
```

Example 3:

```
when A=1 unless B=2 drop table Tab1;
```

2.5 Обычные операторы скриптов

Как правило, обычные операторы используются для управления данными тем или иным образом. Эти операторы могут быть перезаписаны любым числом линий в скрипте и всегда должны заканчиваться точкой с запятой, «;».

Все ключевые слова скрипта можно вводить в любой комбинации символов в нижнем и верхнем регистре. В именах полей и переменных, используемых в операторах, учитывается регистр.

Обзор обычных операторов скриптов

Каждая функция подробно описана после обзора. Также можно щелкнуть имя функции в синтаксисе, чтобы получить немедленный доступ к подробной информации об этой конкретной функции.

Alias

Оператор **alias** используется для установки псевдонима, по которому будет переименовано поле при включении в следующий скрипт.

```
Alias fieldname as aliasname {,fieldname as aliasname}
```

Autonumber

Этот оператор создает уникальное значение целого для каждого определенного оцененного значения поля, возникающего в процессе выполнения скрипта.

```
AutoNumber fields [Using namespace] ]
```

Binary

Оператор **binary** используется для загрузки данных из другого документа QlikView, включая данные Section Access.

```
Binary  [path] filename
```

comment

Позволяет отображать комментарии поля (метаданные) из баз данных и электронных таблиц. Имена полей, отсутствующие в приложении, будут игнорироваться. Если имя поля встречается несколько раз, используется последнее значение.

```
Comment field *fieldlist using mapname
Comment field fieldname with comment
```

comment table

Позволяет отображать комментарии таблицы (метаданные) из баз данных или электронных таблиц.

```
Comment table tablelist using mapname
Comment table tablename with comment
```

Connect

Данная функция недоступна в Qlik Sense SaaS.

Оператор **CONNECT** используется для определения доступа программы Qlik Sense к общей базе данных с помощью интерфейса OLE DB/ODBC. Для интерфейса ODBC необходимо сначала задать источник данных с помощью администратора ODBC.

```
ODBC Connect TO connect-string [ ( access_info ) ]
OLEDB CONNECT TO connect-string [ ( access_info ) ]
CUSTOM CONNECT TO connect-string [ ( access_info ) ]
LIB CONNECT TO connection
```

Declare

Оператор **Declare** используется для создания определений полей, где можно определить отношения между полями или функциями. Ряд определений полей можно использовать для автоматического создания производных полей, которые можно использовать как измерения. Например можно создать определение календаря и использовать его для создания соответствующих измерений, таких как год, месяц, неделя и день, на основе поля даты.

```
definition_name:
Declare [Field[s]] Definition [Tagged tag_list ]
[Parameters parameter_list ]
Fields field_list
[Groups group_list ]

<definition name>:
Declare [Field][s] Definition
Using <existing_definition>
[With <parameter_assignment> ]
```

Derive

Оператор **Derive** используется для создания производных полей на основе определения поля, созданного с помощью оператора **Declare**. Можно указать, для каких полей данных необходимо извлечь поля, или извлечь их явно или неявно на основе тегов полей.

```
Derive [Field[s]] From [Field[s]] field_list Using definition
Derive [Field[s]] From Explicit [Tag[s]] (tag_list) Using definition
Derive [Field[s]] From Implicit [Tag[s]] Using definition
```

Direct Query

Оператор **DIRECT QUERY** обеспечивает доступ к таблицам через подключение ODBC или OLE DB с помощью функции Direct Discovery.

```
Direct Query [path]
```

Directory

Оператор **Directory** задает каталог, в котором будет выполняться поиск файлов данных в последующих операторах **LOAD** до создания нового оператора **Directory**.

```
Directory [path]
```

Disconnect

Оператор **Disconnect** разрывает текущее соединение ODBC/OLE DB/Custom. Этот оператор является дополнительным.

```
Disconnect
```

drop field

Одно или несколько полей Qlik Sense можно удалить из модели данных, а, значит, и из памяти в любой момент выполнения скрипта с помощью оператора **drop field**.

> *Допустимыми являются оба оператора **drop field** и **drop fields**, причем оба они выполняют одно и то же действие. Если таблица не задана, поле удаляется из всех таблиц, в которых оно встречается.*

```
Drop field  fieldname [ , fieldname2 ...] [from tablename1 [ , tablename2
...]]
drop fields fieldname [ , fieldname2 ...] [from tablename1 [ , tablename2
...]]
```

drop table

Одну или несколько внутренних таблиц Qlik Sense можно удалить из модели данных, а значит и из памяти в любой момент выполнения скрипта с помощью оператора **drop table**.

> *Допустимыми являются оба оператора: **drop table** и **drop tables**.*

```
Drop table  tablename [, tablename2 ...]
drop tables[ tablename [, tablename2 ...]
```

Execute

Оператор **Execute** используется для запуска других программ в ходе загрузки данных Qlik Sense. Например, для выполнения необходимых преобразований.

```
Execute commandline
```

FlushLog

Оператор **FlushLog** инициирует запись содержимого буфера скрипта в файл журнала скрипта Qlik Sense.

```
FlushLog
```

Force

Оператор **force** инициирует интерпретацию программой Qlik Sense имен и значений полей последующих операторов **LOAD** и **SELECT** как записанных только символами верхнего регистра, только символами нижнего регистра, всегда приписными буквами или как есть (смешанными). Этот оператор позволяет ассоциировать значения полей в таблицах, выполненных в соответствии с различными условными обозначениями.

```
Force ( capitalization | case upper | case lower | case mixed )
```

LOAD

Оператор **LOAD** загружает поля из файла, из определенных в скрипте данных, из ранее загруженной таблицы, из веб-страницы, из результата последующего оператора **SELECT** или путем создания данных. Также можно загружать данные из аналитических подключений.

```
 Load [ distinct ] *fieldlist
[( from file [ format-spec ] |
from_field fieldassource [format-spec]
inline data [ format-spec ] |
resident table-label |
autogenerate size )]
[ where criterion | while criterion ]
[ group_by groupbyfieldlist ]
[order_by orderbyfieldlist ]
[extension pluginname.functionname(tabledescription)]
```

Let

Оператор **let** создан как дополнение к оператору **set**, используемому для определения переменных скрипта. Оператор **let**, в отличие от оператора **set**, вычисляет выражение, расположенное справа от знака «=» во время выполнения скрипта до присваивания его переменной.

```
Let variablename=expression
```

Loosen Table

Одну или несколько внутренних таблиц данных в программе Qlik Sense можно явно объявить слабосвязанными в ходе выполнения скрипта с помощью оператора **Loosen Table**. При преобразовании таблицы в слабосвязанную все связи между значениями полей в таблице удаляются. Похожего эффекта можно добиться, загрузив каждое поле слабосвязанной таблицы в качестве независимой несвязанной таблицы. Слабосвязанная таблица может применяться в ходе проверки для временной изоляции различных частей структуры данных. Слабосвязанная таблица обозначена в обозревателе таблиц пунктирной линией. Использование одного или нескольких операторов **Loosen Table** в скрипте приведет к тому, что программа Qlik Sense будет игнорировать параметры таблиц, считая их ставшими слабосвязанными до выполнения скрипта.

```
tablename [ , tablename2 ...]
Loosen Tables tablename [ , tablename2 ...]
```

Map ... using

Оператор **map ... using** используется для сопоставления определенных значений полей или выражений со значениями в определенной таблице сопоставления. Таблицу сопоставления можно создать с помощью оператора **Mapping**.

```
Map  *fieldlist Using  mapname
```

NullAsNull

Оператор **NullAsNull** отключает преобразование значений NULL в строчные значения, ранее заданные с помощью оператора **NullAsValue**.

```
NullAsNull *fieldlist
```

NullAsValue

Оператор **NullAsValue** указывает, для каких из полей обнаруженные значения NULL должны быть преобразованы в значения.

```
NullAsValue *fieldlist
```

Qualify

Оператор **Qualify** используется для включения квалификации имен полей, т. е. имена полей получат имя таблицы в качестве префикса.

```
Qualify *fieldlist
```

Rem

Оператор **rem** служит для вставки замечаний или комментариев в скрипт или для временного отключения операторов скрипта без их удаления.

```
Rem string
```

Rename Field

Эта функция скрипта переименовывает одно или несколько существующих полей в программе Qlik Sense после их загрузки.

```
Rename field (using mapname|oldname to newname{ , oldname to newname })
```

```
Rename Fields (using mapname | oldname to newname{ , oldname to newname })
```

Rename Table

Эта функция скрипта переименовывает одну или несколько существующих внутренних таблиц в программе Qlik Sense после их загрузки.

```
Rename table  (using mapname | oldname to newname{ , oldname to newname })
Rename Tables (using mapname | oldname to newname{ , oldname to newname })
```

Section

Оператор **section** позволяет определить, следует ли рассматривать последующие операторы **LOAD** и **SELECT** в качестве данных или определения прав доступа.

```
Section (access | application)
```

Select

Выбор полей из источника данных ODBC или поставщика OLE DB осуществляется с помощью стандартных операторов SQL **SELECT**. Однако то, принимаются операторы **SELECT** или нет, зависит в основном от используемого драйвера ODBC или поставщика OLE DB.

```
Select [all | distinct | distinctrow | top n [percent] ] *fieldlist

From tablelist

[Where criterion ]

[Group by fieldlist [having criterion ] ]

[Order by fieldlist [asc | desc] ]

[ (Inner | Left | Right | Full)Join tablename on fieldref = fieldref ]
```

Set

Оператор **set** используется для определения переменных скрипта. Эти переменные можно использовать для подстановки строк, путей, драйверов и т. д.

```
Set variablename=string
```

Sleep

Оператор **sleep** приостанавливает выполнение скрипта на указанное время.

```
Sleep n
```

SQL

Оператор **SQL** позволяет отправлять произвольную команду SQL посредством подключения ODBC или OLE DB.

```
SQL sql_command
```

SQLColumns

Оператор **sqlcolumns** возвращает набор полей с описанием столбцов источника данных ODBC или OLE DB, с которыми выполнена операция **connect**.

```
SQLColumns
```

SQLTables

Оператор **sqltables** возвращает набор полей с описанием таблиц источника данных ODBC или OLE DB, с которыми выполнена операция **connect**.

```
SQLTables
```

SQLTypes

Оператор **sqltypes** возвращает набор полей с описанием типов источника данных ODBC или OLE DB, с которыми выполнена операция **connect**.

```
SQLTypes
```

Star

Строку, которая представляет набор всех значений поля в базе данных, можно определить с помощью оператора **star**. Она влияет на последующие операторы **LOAD** и **SELECT**.

```
Star is [ string ]
```

Store

Оператор **Store** создает файл QVD, CSV или text.

```
Store [ *fieldlist from] table into filename [ format-spec ];
```

Tag

Этот оператор скрипта позволяет присваивать теги одному или нескольким полям или таблицам. Если делается попытка присвоить тег полю или таблице, отсутствующим в приложении, то эта операция будет проигнорирована. Если обнаружены конфликты между именами полей или тегов, то используется последнее значение.

```
Tag[field|fields] fieldlist with tagname
Tag [field|fields] fieldlist using mapname
Tag table tablelist with tagname
```

Trace

Оператор **trace** записывает строку в окно **Ход выполнения скрипта** и в файл журнала скрипта, если тот используется. Он очень полезен для отладки. Расширение $, добавляемое к переменным, вычисляемым до оператора **trace**, позволяет настроить сообщение.

```
Trace string
```

Unmap

Оператор **Unmap** деактивирует значение поля mapping, заданное предыдущим оператором **Map … Using** для последующих загружаемых полей.

```
Unmap *fieldlist
```

Unqualify

Оператор **Unqualify** используется для снятия уточнения имен полей, которое ранее было включено оператором **Qualify**.

```
Unqualify *fieldlist
```

Untag

Этот оператор скрипта позволяет удалять теги из полей или таблиц. Если делается попытка удалить тег из поля или таблицы, отсутствующим в приложении, то эта операция будет проигнорирована.

```
Untag[field|fields] fieldlist with tagname
Tag [field|fields] fieldlist using mapname
Tag table tablelist with tagname
```

Alias

Оператор **alias** используется для установки псевдонима, по которому будет переименовано поле при включении в следующий скрипт.

Синтаксис:

```
alias fieldname as aliasname {,fieldname as aliasname}
```

Аргументы:

Аргументы

Аргумент	Описание
fieldname	Имя поля в исходных данных
aliasname	Имя псевдонима, которое требуется использовать взамен

Примеры и результаты:

Пример	Результат
`Alias ID_N as NameID;`	
`Alias A as Name, B as Number, C as Date;`	Изменения имени, определенные данным оператором, применяются ко всем последующим операторам **SELECT** и **LOAD**. Новый псевдоним для имени поля может быть задан с помощью нового оператора **alias** в любой последующей точке скрипта.

AutoNumber

Этот оператор создает уникальное значение целого для каждого определенного оцененного значения поля, возникающего в процессе выполнения скрипта.

Также можно использовать функцию *autonumber (page 468)* внутри оператора **LOAD**, однако такой подход сопряжен с ограничениями при использовании оптимизированной загрузки. Для создания оптимизированной загрузки можно сначала загрузить данные из файла **QVD**, а затем преобразовать значения в ключи символов с помощью оператора **AutoNumber**.

Синтаксис:

```
AutoNumber *fieldlist [Using namespace] ]
```

Аргументы:

Аргументы

Аргумент	Описание
*fieldlist	Список разделенных запятыми полей, значения которого подлежат замене уникальными целыми числами. В именах полей можно использовать знаки подстановки ? и *, чтобы включить все поля с совпадающими именами. Также для включения всех полей можно использовать символ *. При использовании знаков подстановки следует заключить имена полей в кавычки.
namespace	**Параметр** `namespace` является дополнительным. Его можно использовать для создания пространства имен, где одинаковые значения в разных полях будут использовать один и тот же ключ. Если этот параметр не используется, все поля будут иметь отдельные индексы ключей.

Ограничения:

При наличии нескольких операторов **LOAD** в скрипте необходимо поместить оператор **AutoNumber** за последним оператором **LOAD**.

Пример – скрипт с использованием AutoNumber

Пример скрипта

В этом примере данные сначала загружаются без оператора **AutoNumber**. После этого добавляется оператор **AutoNumber**, чтобы продемонстрировать результат его применения.

Данные, используемые в примере

Загрузите следующие данные через встроенную загрузку в редакторе загрузки данных, чтобы создать пример с выражениями скрипта, показанный ниже. Пока оставьте оператор **AutoNumber** закомментированным.

```
RegionSales: LOAD *,  Region &'|'& Year &'|'& Month as KeyToOtherTable INLINE [ Region, Year,
Month, Sales North,     2014,   May,    245 North,      2014,   May,    347 North,      2014,   June,   127 S
June,   645 South,      2013,   May,    367 South,      2013,   May,    221 ];
&'|'& Year &'|'& Month as KeyToOtherTable INLINE [Region, Year, Month, Budget North,  2014,
May,    200 North,  2014,   May,    350 North,  2014,   June,   150 South,  2014,   June,
500 South,  2013,   May,    300 South,  2013,   May,    200 ];  //AutoNumber KeyToOtherTable;
```

Создание визуализаций

Создайте две визуализации таблиц на листе Qlik Sense. В первую таблицу добавьте измерения **KeyToOtherTable**, **Region**, **Year**, **Month** и **Sales**. Во вторую таблицу добавьте измерения **KeyToOtherTable**, **Region**, **Year**, **Month** и **Budget** .

Результат

Таблица RegionSales

KeyToOtherTable	Region	Year	Month	Sales
North|2014|June	North	2014	June	127
North|2014|May	North	2014	May	245
North|2014|May	North	2014	May	347
South|2013|May	South	2013	May	221
South|2013|May	South	2013	May	367
South|2014|June	South	2014	June	645

Таблица Budget

KeyToOtherTable	Region	Year	Month	Budget
North|2014|June	North	2014	June	150
North|2014|May	North	2014	May	200
North|2014|May	North	2014	May	350
South|2013|May	South	2013	May	200

KeyToOtherTable	Region	Year	Month	Budget
South|2013|May	South	2013	May	300
South|2014|June	South	2014	June	500

Объяснение

В примере отображается составное поле **KeyToOtherTable**, которое связывает две таблицы. Оператор **AutoNumber** не используется. Обратите внимание на длину значений **KeyToOtherTable**.

Добавление оператора AutoNumber

Раскомментируйте оператор **AutoNumber** в скрипте загрузки.

```
AutoNumber KeyToOtherTable;
```

Результат

Таблица RegionSales

KeyToOtherTable	Region	Year	Month	Sales
1	North	2014	June	127
1	North	2014	May	245
2	North	2014	May	347
3	South	2013	May	221
4	South	2013	May	367
4	South	2014	June	645

Таблица Budget

KeyToOtherTable	Region	Year	Month	Budget
1	North	2014	June	150
1	North	2014	May	200
2	North	2014	May	350
3	South	2013	May	200
4	South	2013	May	300
4	South	2014	June	500

Объяснение

Значения поля **KeyToOtherTable** заменены уникальными целыми числами, в результате чего длина значений поля сократилась, а память освободилась. Оператор **AutoNumber** затрагивает

ключевые поля в обеих таблицах, таблицы остаются связанными. Этот пример представлен в целях демонстрации, поэтому в данном случае информация краткая, но при использовании таблицы, содержащей большое количество строк, информация будет более содержательной.

Binary

Оператор **binary** используется для загрузки данных из другого приложения Qlik Sense или документа QlikView, включая данные доступа к секции. Другие элементы приложения не включены, например, листы, истории, визуализации, основные элементы или переменные.

В скрипте допускается не более одного оператора **binary**. Оператор **binary** должен быть первым оператором скрипта, даже перед операторами SET, которые обычно расположены в начале скрипта.

Синтаксис:

```
binary [path] filename
```

Аргументы:

Аргументы

Аргумент	Описание
path	Путь к файлу, который должен быть ссылкой на подключение к данным папки. Это необходимо, если файл расположен не в рабочем каталоге Qlik Sense. **Пример:** *'lib://Table Files/'* В прежней версии режима написания скриптов следующие форматы пути тоже поддерживаются: • абсолютный **Пример:** *c:\data* • относительно приложения, содержащего эту строку скрипта. **Пример:** *data*
filename	Имя файла, включая расширение файла .qvw или .qvf

Ограничения:

Оператор **binary** нельзя использовать для загрузки данных из приложения в одном развертывании Qlik Sense Enterprise, указав ссылку на идентификатор приложения. Загрузку можно выполнять только из файла *.qvf*.

Примеры

Строка	Описание
`Binary lib://DataFolder/customer.qvw;`	В этом примере файл должен быть расположен в подключении к данным **Папка**. Это может быть, например, папка, созданная администратором на сервере Qlik Sense. Щелкните **Создать новое подключение** в редакторе загрузки данных и выберите **Папка** в разделе **Расположения файлов**.
`Binary customer.qvf;`	В этом примере файл должен быть расположен в рабочем каталоге Qlik Sense.
`Binary c:\qv\customer.qvw;`	Пример с использованием абсолютного пути файла работает только в прежней версии режима написания скриптов.

Comment field

Позволяет отображать комментарии поля (метаданные) из баз данных и электронных таблиц. Имена полей, отсутствующие в приложении, будут игнорироваться. Если имя поля встречается несколько раз, используется последнее значение.

Синтаксис:

```
comment [fields] *fieldlist using mapname
comment [field] fieldname with comment
```

Таблица сопоставления должна включать в себя два столбца: в первом содержатся имена полей, а во втором – комментарии.

Аргументы:

Аргументы

Аргумент	Описание
**fieldlist*	Список разделенных запятыми полей, подлежащих комментированию. Символ * в качестве списка полей обозначает все поля. В именах полей разрешается использовать знаки подстановки * и ?. При использовании знаков подстановки, возможно, понадобится заключать имена полей в кавычки.
mapname	Имя таблицы сопоставления, считанной ранее в операторе сопоставления **LOAD** или **SELECT**.
fieldname	Имя поля, для которого необходимо добавить комментарий.
comment	Комментарий, который следует добавить к полю.

Example 1:

```
commentmap:
```

```
mapping LOAD * inline [
a,b
Alpha,This field contains text values
Num,This field contains numeric values
];
comment fields using commentmap;
```

Example 2:

```
comment field Alpha with AFieldContainingCharacters;
comment field Num with '*A field containing numbers';
comment Gamma with 'Mickey Mouse field';
```

Comment table

Позволяет отображать комментарии таблицы (метаданные) из баз данных или электронных таблиц.

Имена таблиц, отсутствующие в приложении, будут игнорироваться. Если имя таблицы встречается несколько раз, используется последнее значение. Для чтения комментариев из источника данных может использоваться ключевое слово.

Синтаксис:

```
comment [tables] tablelist using mapname
comment [table] tablename with comment
```

Аргументы:

Аргументы

Аргумент	Описание
tablelist	(table{,table})
mapname	Имя таблицы сопоставления, считанной ранее в операторе сопоставления **LOAD** или **SELECT**.
tablename	Имя таблицы, для которой необходимо добавить комментарий.
comment	Комментарий, который следует добавить в таблицу.

Example 1:

```
Commentmap:
mapping LOAD * inline [
a,b
Main,This is the fact table
Currencies, Currency helper table
];
comment tables using Commentmap;
```

Example 2:

```
comment table Main with 'Main fact table';
```

Connect

Оператор **CONNECT** используется для определения доступа программы Qlik Sense к общей базе данных с помощью интерфейса OLE DB/ODBC. Для интерфейса ODBC необходимо сначала задать источник данных с помощью администратора ODBC.

Данная функция недоступна в Qlik Sense SaaS.

Этот оператор поддерживает только подключения к данным из папки в стандартном режиме.

Синтаксис:

```
ODBC CONNECT TO connect-string
OLEDB CONNECT TO connect-string
CUSTOM CONNECT TO connect-string
LIB CONNECT TO connection
```

Аргументы:

Аргументы

Аргумент	Описание

connect-string	`connect-string ::= datasourcename { ; conn-spec-item }`
	Строка подключения содержит имя источника данных и может включать в себя один или несколько дополнительных элементов спецификаций подключения. Если имя источника данных содержит пробелы, либо присутствуют какие-либо элементы спецификаций подключения, строка подключения должна быть заключена в кавычки.
	
	datasourcename должен являться определенным источником данных ODBC или строкой, которая определяет поставщика OLE DB.
	
	`conn-spec-item ::=`**`DBQ=`**`database_specifier |`**`DriverID=`**`driver_specifier |`**`UID=`**`userid |`**`PWD=`**`password`
	
	Возможные элементы спецификаций подключения могут различаться в зависимости от базы данных. Для некоторых баз данных возможно использование других элементов, отличных от вышеупомянутых. Для баз данных OLE DB некоторые элементы, относящиеся к подключению, являются обязательными, а не дополнительными.

connection	Имя подключения данных, сохраненное в редакторе загрузки данных.

Если интерфейс **ODBC** помещен перед оператором **CONNECT**, будет использоваться интерфейс ODBC; в остальных случаях будет использоваться OLE DB.

Оператор **LIB CONNECT TO** использует для подключения к базе данных сохраненное подключение, созданное в редакторе загрузки данных.

Example 1:

```
ODBC CONNECT TO 'Sales
DBQ=C:\Program Files\Access\Samples\Sales.mdb';
```

Источник данных, определенный посредством этого оператора, используется последующими операторами **Select (SQL)** до тех пор, пока не будет создан новый оператор **CONNECT**.

Example 2:

```
LIB CONNECT TO 'DataConnection';
```

Connect32

Этот оператор используется так же, как оператор **CONNECT**, однако вынуждает 64-разрядную систему использовать 32-разрядного поставщика ODBC/OLE DB. Не применим для пользовательского подключения.

Connect64

Этот оператор используется так же, как оператор **CONNECT**, однако требует использования 64-разрядного поставщика. Не применим для пользовательского подключения.

Declare

Оператор **Declare** используется для создания определений полей, где можно определить отношения между полями или функциями. Ряд определений полей можно использовать для автоматического создания производных полей, которые можно использовать как измерения. Например можно создать определение календаря и использовать его для создания соответствующих измерений, таких как год, месяц, неделя и день, на основе поля даты.

Можно использовать **Declare**, чтобы установить новое определение поля или создать определение поля на основе уже существующего определения.

Установка нового определения поля

Синтаксис:

```
definition_name:
Declare [Field[s]] Definition [Tagged tag_list ]
[Parameters parameter_list ]
Fields field_list
```

Аргументы:

Аргумент	Описание
definition_name	Имя определения поля с двоеточием в конце.  *Не используйте autoCalendar в качестве имени определения поля, так как это имя зарезервировано для автоматически созданных шаблонов календаря.* **Пример:** `Calendar:`
tag_list	Список тегов, разделенных запятыми, которые будут применяться к полям, извлеченным из определения поля. Применять теги не обязательно, но если не применить теги, которые используются для определения порядка сортировки, такие как $date, $numeric или $text, сортировка производных полей будет выполняться по порядку загрузки, как указано по умолчанию. **Пример:** `'$date'Thank you for bringing this to our attention, and apologies for the inconvenience.`
parameter_list	Список параметров, разделенных запятыми. Параметр определяется в виде `name=value` и назначается в качестве начального значения, которое можно переписать при повторном использовании определения поля. Дополнительно. **Пример:** `first_month_of_year = 1`
field_list	Список полей, разделенных запятыми, которые будут созданы при использовании определения поля. Поле определяется в виде `<expression> As field_name tagged tag`. Используйте `$1` для ссылки на поле данных, из которого должны быть созданы производные поля. **Пример:** `Year($1) As Year tagged ('$numeric')`

Пример:

```
Calendar:
DECLARE FIELD DEFINITION TAGGED '$date'
   Parameters
      first_month_of_year = 1
   Fields
```

```
      Year($1) As Year Tagged ('$numeric'),
      Month($1) as Month Tagged ('$numeric'),
      Date($1) as Date Tagged ('$date'),
      Week($1) as Week Tagged ('$numeric'),
      Weekday($1) as Weekday Tagged ('$numeric'),
      DayNumberOfYear($1, first_month_of_year) as DayNumberOfYear Tagged ('$numeric')
;
```

Календарь теперь определен. Можно применить его к загруженным полям с датами, в данном случае OrderDate и ShippingDate, с помощью предложения **Derive**.

Повторное использование существующего определения поля

Синтаксис:

```
<definition name>:
Declare [Field][s] Definition
Using <existing_definition>
[With <parameter_assignment> ]
```

Аргументы:

Аргумент	Описание
definition_name	Имя определения поля с двоеточием в конце. **Пример:** `MyCalendar:`
existing_definition	Определение поля для повторного использования при создании нового определения поля. Новое определение поля будет работать таким же образом, как определение, на котором оно основано, за исключением случая, когда используется parameter_assignment для изменения значения, используемого в выражениях поля. **Пример:** `Using Calendar`
parameter_assignment	Список назначений параметров, разделенных запятыми. Назначение параметра определяется в виде `name=value`, оно переопределяет значение параметра, заданное в базовом определении поля. Дополнительно. **Пример:** `first_month_of_year = 4`

Пример:

В этом примере мы повторно используем определение календаря, созданное в предыдущем примере. В этом случае мы хотим использовать финансовый год, начинающийся в апреле. Это достигается путем назначения значения 4 параметру first_month_of_year, который повлияет на определяемое поле DayNumberOfYear.

В этом примере допускается, что вы используете данные образца и определение поля из предыдущего примера.

```
MyCalendar:
DECLARE FIELD DEFINITION USING Calendar WITH first_month_of_year=4;

DERIVE FIELDS FROM FIELDS OrderDate,ShippingDate USING MyCalendar;
```

После повторной загрузки скрипта данных созданные поля будут доступны в редакторе листа с именами OrderDate.MyCalendar.* и ShippingDate.MyCalendar.*.

Derive

Оператор **Derive** используется для создания производных полей на основе определения поля, созданного с помощью оператора **Declare**. Можно указать, для каких полей данных необходимо извлечь поля, или извлечь их явно или неявно на основе тегов полей.

Синтаксис:

```
Derive [Field[s]] From [Field[s]] field_list Using definition
Derive [Field[s]] From Explicit [Tag[s]] tag_list Using definition
Derive [Field[s]] From Implicit [Tag[s]] Using definition
```

Аргументы:

Аргументы

Аргумент	Описание
definition	Имя определения поля для использования при извлечении полей. **Пример:** `Calendar`
field_list	Список полей данных, разделенных запятыми, из которых будут созданы производные поля на основе определения поля. Поля данных должны быть полями, уже загруженными в скрипт. **Пример:** `OrderDate, ShippingDate`
tag_list	Список тегов, разделенных запятыми. Производные поля будут созданы для всех полей данных с любым из перечисленных тегов. Список тегов должен быть заключен в круглые скобки. **Пример:** `('$date','$timestamp')`

Примеры:

- Извлечь поля для определенных полей данных.
  В этом случае мы указываем поля OrderDate и ShippingDate.
  `DERIVE FIELDS FROM FIELDS OrderDate,ShippingDate USING Calendar;`
- Извлечь поля для всех полей с определенным тегом.
  В этом случае мы извлекаем поля на основе Calendar для всех полей с тегом `$date`.
  `DERIVE FIELDS FROM EXPLICIT TAGS ('$date') USING Calendar;`
- Извлечь поля для всех полей с тегом определения поля.
  В этом случае мы извлекаем поля для всех полей данных с тем же тегом, что существует в определении поля Calendar, который в данном случае является `$date`.
  `DERIVE FIELDS FROM IMPLICIT TAG USING Calendar;`

Direct Query

Оператор **DIRECT QUERY** обеспечивает доступ к таблицам через подключение ODBC или OLE DB с помощью функции Direct Discovery.

Синтаксис:

```
DIRECT QUERY DIMENSION fieldlist [MEASURE fieldlist] [DETAIL fieldlist] FROM
tablelist
[WHERE where_clause]
```

Ключевые слова **DIMENSION**, **MEASURE** и **DETAIL** можно использовать в любом порядке.

Предложения ключевых слов **DIMENSION** и **FROM** требуются во всех операторах **DIRECT QUERY**. Ключевое слово **FROM** должно стоять после ключевого слова **DIMENSION**.

Поля, указанные сразу после ключевого слова **DIMENSION**, загружаются в память и могут использоваться для создания связей между данными в памяти и данными Direct Discovery.

*Оператор **DIRECT QUERY** не может содержать предложения **DISTINCT** или **GROUP BY**.*

С помощью ключевого слова **MEASURE** можно определить поля, которые Qlik Sense будет распознавать на «уровне метаданных». Фактические данные поля measure находятся только в базе данных во время процесса загрузки данных. Они извлекаются через прямое подключение с помощью выражений диаграммы, используемых в визуализации.

Обычно поля с дискретными значениями, которые используются в качестве измерений, загружаются с ключевым словом **DIMENSION**, тогда как числа, используемые только при агрегировании, должны быть выбраны с ключевым словом **MEASURE**.

Поля **DETAIL** обеспечивают информацию или подробности, такие как поля с комментариями, которые пользователь может отобразить в простой таблице, которую можно развернуть и просмотреть подробности. Поля **DETAIL** не могут использоваться в выражениях диаграммы.

Оператор **DIRECT QUERY** не зависит от источника данных для источников, поддерживающих SQL. Поэтому один и тот же оператор **DIRECT QUERY** можно использовать для разных баз данных SQL без внесения изменений. Direct Discovery создает запросы для конкретных баз данных, если необходимо.

Исходный синтаксис источника данных можно использовать, когда пользователь знает, какая база данных запрашивается, и хочет использовать специальные расширения для базы данных SQL. Исходный синтаксис источника данных поддерживается:

- В качестве выражения поля в предложениях **DIMENSION** и **MEASURE**
- В качестве содержимого предложения **WHERE**

Примеры:

```
DIRECT QUERY
      DIMENSION Dim1, Dim2

      MEASURE
            NATIVE ('X % Y') AS X_MOD_Y
FROM TableName

DIRECT QUERY
      DIMENSION Dim1, Dim2

      MEASURE X, Y

      FROM TableName

      WHERE NATIVE ('EMAIL MATCHES "\*.EDU"')
```

Следующие термины используются в качестве ключевых слов и поэтому не могут использоваться в качестве имени столбца или поля без кавычек: and, as, detach, detail, dimension, distinct, from, in, is, like, measure, native, not, or, where

Аргументы:

Аргумент	Описание
fieldlist	Список спецификаций поля, разделенных запятыми, *fieldname {, fieldname}*. Спецификация поля может быть именем поля. В этом случае такое же имя используется для имени столбца базы данных и имени поля Qlik Sense. Также спецификация поля может быть «полем alias». В этом случае выражению базы данных или имени столбца задается имя поля Qlik Sense.
tablelist	Список имен таблиц или представлений в базе данных, из которой загружаются данные. Как правило, это представления, содержащие оператор JOIN, выполненный в базе данных.

Аргумент	Описание

where_ clause	Здесь не приведено полное описание синтаксиса предложений базы данных **WHERE**, но большинство «реляционных выражений» SQL разрешено использовать, включая вызовы функций, оператор **LIKE** для строк, **IS NULL**, **IS NOT NULL**, а оператор **IN. BETWEEN** не включен.
	
	NOT – это унарный оператор, в отличие от модификатора на определенные ключевые слова.
	
	Примеры:
	`WHERE x > 100 AND "Region Code" IN ('south', 'west')`
	
	`WHERE Code IS NOT NULL and Code LIKE '%prospect'`
	
	`WHERE NOT X in (1,2,3)`
	Последний пример не может быть записан как:
	`WHERE X NOT in (1,2,3)`

Пример:

В этом примере используется таблица базы данных с именем TableName, содержащая поля Dim1, Dim2, Num1, Num2 и Num3.Поля Dim1 и Dim2 будут загружены в набор данных Qlik Sense.

```
DIRECT QUERY DIMENSTION Dim1, Dim2 MEASURE Num1, Num2, Num3 FROM TableName ;
```

Поля Dim1 и Dim2 будут доступны для использования в качестве измерений. Поля Num1, Num2 и Num3 будут доступны для агрегирований. Поля Dim1 и Dim2 также доступны для агрегирований. Тип агрегирований, для которого могут использоваться поля Dim1 и Dim2, зависит от их типов данных. Например, во многих случаях поля **DIMENSION** содержат строковые данные, такие как имена или номера счетов. Эти поля нельзя суммировать, но их можно посчитать: `count(Dim1)`.

*Операторы **DIRECT QUERY** записываются непосредственно в редактор скриптов. Чтобы упростить конструкцию операторов **DIRECT QUERY**, можно создать оператор **SELECT** из подключения к данным, а затем редактировать созданный скрипт, чтобы переделать его в оператор **DIRECT QUERY**.*
*Например, оператор **SELECT***

```
SQL SELECT
 SalesOrderID,
 RevisionNumber,
 OrderDate,
 SubTotal,
 TaxAmt
FROM MyDB.Sales.SalesOrderHeader;
```

*можно заменить следующим оператором **DIRECT QUERY**:*

```
DIRECT QUERY
 DIMENSION
 SalesOrderID,
 RevisionNumber

MEASURE
 SubTotal,
 TaxAmt

DETAIL
OrderDate

FROM MyDB.Sales.SalesOrderHeader;
```

Списки полей Direct Discovery

Список полей – это список спецификаций поля, разделенных запятыми: *fieldname {, fieldname}*. Спецификация поля может быть именем поля. В этом случае такое же имя используется для имени столбца базы данных и имени поля. Также спецификация поля может быть «полем alias». В этом случае выражению базы данных или имени столбца задается имя поля Qlik Sense.

Имена полей могут быть простыми именами или заключенными в кавычки. Простое имя начинается с буквенного символа Юникода и состоит из комбинации букв, цифр и знаков подчеркивания. Имена в кавычках начинаются с двойной кавычки и содержат любую последовательность символов. Если имя, заключенное в кавычки, содержит двойные кавычки, эти кавычки представляются в виде двух смежных двойных кавычек.

Имена полей Qlik Sense используются с учетом регистра. Имена полей базы данных могут учитывать или не учитывать регистр, в зависимости от базы данных. Запрос Direct Discovery сохраняет регистр всех идентификаторов полей и псевдонимов. В следующем примере псевдоним "MyState" используется для внутренних целей для сохранения данных из столбца базы данных "STATEID".

```
DIRECT QUERY Dimension STATEID as MyState Measure AMOUNT from SALES_TABLE;
```

Это отличается от результата использования оператора **SQL Select** с псевдонимом. Если псевдоним не заключен в кавычки, результат будет содержать регистр по умолчанию столбца, возвращенного целевой базой данных. В следующем примере оператор **SQL Select** для базы данных Oracle создает "MYSTATE," со всеми буквами в верхнем регистре, как и внутренний псевдоним Qlik Sense, даже если в псевдониме используются символы в разном регистре. Оператор **SQL Select** использует имя столбца, возвращенное базой данных, которое в случае Oracle состоит из всех символов в верхнем регистре.

```
SQL Select STATEID as MyState, STATENAME from STATE_TABLE;
```

Чтобы избежать такого поведения, для указания псевдонима используйте оператор LOAD.

```
Load STATEID as MyState, STATENAME;
SQL Select STATEID, STATEMENT from STATE_TABLE;
```

В данном примере столбец "STATEID" сохраняется Qlik Sense для внутренних целей в качестве "MyState".

Большинство скалярных выражений базы данных разрешено использовать в качестве спецификаций поля. Вызовы функций также можно использовать в качестве спецификаций поля. Выражения могут содержать константы: булевы, числовые или строки, заключенные в одиночные кавычки (встроенные одинарные кавычки представляются в виде двух смежных одинарных кавычек.).

Примеры:

```
DIRECT QUERY

   DIMENSION

      SalesOrderID, RevisionNumber

   MEASURE

      SubTotal AS "Sub Total"

FROM AdventureWorks.Sales.SalesOrderHeader;


DIRECT QUERY

   DIMENSION

      "SalesOrderID" AS "Sales Order ID"
```

```
      MEASURE
         SubTotal,TaxAmt,(SubTotal-TaxAmt) AS "Net Total"
   FROM AdventureWorks.Sales.SalesOrderHeader;

   DIRECT QUERY
      DIMENSION
         (2*Radius*3.14159) AS Circumference,
         Molecules/6.02e23 AS Moles
      MEASURE
         Num1 AS numA
   FROM TableName;
```

```
DIRECT QUERY
   DIMENSION
      concat(region, 'code') AS region_code
   MEASURE
      Num1 AS NumA
FROM TableName;
```

Direct Discovery не поддерживает использование агрегирования в операторах **LOAD**. При использовании агрегирования результат может быть непредсказуемым. Оператор **LOAD** не следует использовать следующим образом:

```
DIRECT QUERY DIMENSION stateid, SUM(amount*7) AS MultiFirst MEASURE amount FROM sales_table;
```

SUM не следует использовать в операторе **LOAD**.

Direct Discovery также не поддерживает функции Qlik Sense в операторах **Direct Query**. Например, использование следующей спецификации для поля **DIMENSION** приведет к возникновению ошибки, когда поле "Mth" будет использоваться в качестве измерения в визуализации:

```
month(ModifiedDate) as Mth
```

Directory

Оператор **Directory** задает каталог, в котором будет выполняться поиск файлов данных в последующих операторах **LOAD** до создания нового оператора **Directory**.

Синтаксис:

```
Directory[path]
```

Если оператор **Directory** задается без параметра **path** или вообще опускается, программа Qlik Sense будет искать в рабочем каталоге Qlik Sense.

Аргументы:

Аргументы

Аргумент	Описание
path	Текст может интерпретироваться как путь к файлу data. Path – путь к файлу: • абсолютный **Пример:** *c:\data* • относительно рабочего каталога приложения Qlik Sense. **Пример:** *data* • URL-адрес (HTTP или FTP), указывающий на местоположение в Интернете или интрасети. **Пример:** *http://www.qlik.com*

Примеры:

```
DIRECTORY C:\userfiles\data; // OR -> DIRECTORY data\

LOAD * FROM
[data1.csv] // ONLY THE FILE NAME CAN BE SPECIFIED HERE (WITHOUT THE FULL PATH)
(ansi, txt, delimiter is ',', embedded labels);

LOAD * FROM
[data2.txt] // ONLY THE FILE NAME CAN BE SPECIFIED HERE UNTIL A NEW DIRECTORY STATEMENT IS
MADE
(ansi, txt, delimiter is '\t', embedded labels);
```

Disconnect

Оператор **Disconnect** разрывает текущее соединение ODBC/OLE DB/Custom. Этот оператор является дополнительным.

Синтаксис:

```
Disconnect
```

Подключение будет разорвано автоматически при выполнении нового оператора **connect** или после завершения выполнения скрипта.

Пример:

```
Disconnect;
```

Drop

Ключевое слово скрипта **Drop** можно использовать для удаления таблиц или полей из базы данных.

Drop field

Одно или несколько полей Qlik Sense можно удалить из модели данных, а, значит, и из памяти в любой момент выполнения скрипта с помощью оператора **drop field**.

*Допустимыми являются оба оператора **drop field** и **drop fields**, причем оба они выполняют одно и то же действие. Если таблица не задана, поле удаляется из всех таблиц, в которых оно встречается.*

Синтаксис:

```
Drop field fieldname { , fieldname2 ...} [from tablename1 { , tablename2
...}]
Drop fields fieldname { , fieldname2 ...} [from tablename1 { , tablename2
...}]
```

Примеры:

```
Drop field A;
Drop fields A,B;
Drop field A from X;
Drop fields A,B from X,Y;
```

Drop table

Одну или несколько внутренних таблиц Qlik Sense можно удалить из модели данных, а значит и из памяти в любой момент выполнения скрипта с помощью оператора **drop table**.

Синтаксис:

```
drop table  tablename {, tablename2 ...}
drop tables tablename {, tablename2 ...}
```

*Допустимыми являются оба оператора: **drop table** и **drop tables**.*

В результате выполнения этого действия произойдет удаление следующих элементов:

- Реальной таблицы.
- Всех полей, которые не относятся к остальным таблицам.
- Значений полей в остальных полях, относящихся только к отброшенным таблицам.

Примеры и результаты:

Пример	Результат
`drop table Orders, Salesmen, T456a;`	Эта строка предписывает удаление из памяти трех таблиц.

`Tab1:`
`Load * Inline [`
`Customer, Items, UnitPrice`
`Bob, 5, 1.50`
`];`
 	
`Tab2:`
`LOAD Customer, Sum( Items * UnitPrice ) as Sales`
`resident Tab1`
`group by Customer;`
 	
`drop table Tab1;`
	После создания таблицы *Tab2* таблица *Tab1* удаляется.

Drop table

Одну или несколько внутренних таблиц Qlik Sense можно удалить из модели данных, а значит и из памяти в любой момент выполнения скрипта с помощью оператора **drop table**.

Синтаксис:

```
drop table  tablename {, tablename2 ...}
drop tables tablename {, tablename2 ...}
```

Допустимыми являются оба оператора: ***drop table*** *и* ***drop tables****.*

В результате выполнения этого действия произойдет удаление следующих элементов:

- Реальной таблицы.
- Всех полей, которые не относятся к остальным таблицам.
- Значений полей в остальных полях, относящихся только к отброшенным таблицам.

Примеры и результаты:

Пример	Результат
`drop table Orders, Salesmen, T456a;`	Эта строка предписывает удаление из памяти трех таблиц.

Пример	Результат
Tab1: Load * Inline [Customer, Items, UnitPrice Bob, 5, 1.50]; Tab2: LOAD Customer, Sum(Items * UnitPrice) as Sales resident Tab1 group by Customer; drop table Tab1;	После создания таблицы *Tab2* таблица *Tab1* удаляется.

Execute

Оператор **Execute** используется для запуска других программ в ходе загрузки данных Qlik Sense. Например, для выполнения необходимых преобразований.

Данная функция недоступна в Qlik Sense SaaS.

Этот оператор не поддерживается в стандартном режиме.

Синтаксис:

```
execute commandline
```

Аргументы:

Аргументы

Аргумент	Описание
commandline	Текст, который может интерпретироваться операционной системой как командная строка. Можно обратиться к абсолютному пути файла или пути папки lib://.

Для использования **Execute** должны быть выполнены следующие условия:

- Необходимо запустить устаревший режим (применимо для Qlik Sense и Qlik Sense Desktop).
- Для параметра OverrideScriptSecurity необходимо установить значение 1 в файле *Settings.ini* (применимо для Qlik Sense).
  Файл *Settings.ini* расположен в папке *C:\ProgramData\Qlik\Sense\Engine* и обычно он пуст.

*Если для OverrideScriptSecurity установлено включение **Execute**, любой пользователь может выполнить файлы на сервере. Например, пользователь может прикрепить исполняемый файл к приложению, а затем выполнить файл в скрипте загрузки данных.*

Выполните следующие действия.

1. Создайте копию *Settings.ini* и откройте ее в текстовом редакторе.
2. Убедитесь, что в первой строке файла указано *[Параметры 7]*.
3. Вставьте новую строку и введите *OverrideScriptSecurity=1*.
4. Вставьте пустую строку в конце файла.
5. Сохраните файл.
6. Замените *Settings.ini* отредактированным файлом.
7. Перезапустите Qlik Sense Engine Service (QES).

Если программа Qlik Sense запущена в качестве службы, некоторые команды могут работать не так, как ожидается.

Пример:

```
Execute C:\Program Files\Office12\Excel.exe;

Execute lib://win\notepad.exe // win is a folder connection referring to c:\windows
```

Field/Fields

Ключевые слова скрипта **Field** и **Fields** используются в операторах **Declare**, **Derive**, **Drop**, **Comment**, **Rename** и **Tag/Untag**.

FlushLog

Оператор **FlushLog** инициирует запись содержимого буфера скрипта в файл журнала скрипта Qlik Sense.

Синтаксис:

```
FlushLog
```

Содержимое буфера записывается в файл журнала. Эта команда может быть полезна для целей отладки, так как вы получите данные, которые в противном случае могли быть потеряны в случае ошибки при выполнении скрипта.

Пример:

```
FlushLog;
```

Force

Оператор **force** инициирует интерпретацию программой Qlik Sense имен и значений полей последующих операторов **LOAD** и **SELECT** как записанных только символами верхнего регистра, только символами нижнего регистра, всегда приписными буквами

или как есть (смешанными). Этот оператор позволяет ассоциировать значения полей в таблицах, выполненных в соответствии с различными условными обозначениями.

Синтаксис:

```
Force ( capitalization | case upper | case lower | case mixed )
```

Если не указан ни один параметр, применяется force case mixed. Оператор force действует до создания следующего оператора force.

Оператор **force** не влияет на секцию доступа: регистр во всех загруженных значениях полей не учитывается.

Примеры и результаты

Пример	Результат

В данном примере показано принудительное использование прописных букв.
`FORCE Capitalization;`
`Capitalization:`
`LOAD * Inline [`
`ab`
`Cd`
`eF`
`GH`
`];`
	Таблица **Capitalization** содержит следующие значения:
	`Ab`
	`Cd`
	`Ef`
	`Gh`
	Все значения записываются прописными буквами.

В данном примере показано принудительное использование верхнего регистра.
`FORCE Case Upper;`
`CaseUpper:`
`LOAD * Inline [`
`ab`
`Cd`
`eF`
`GH`
`];`
	Таблица **CaseUpper** содержит следующие значения:
	`AB`
	`CD`
	`EF`
	`GH`
	Все значения записываются в верхнем регистре.

В данном примере показано принудительное использование нижнего регистра.
`FORCE Case Lower;`
`CaseLower:`
`LOAD * Inline [`
`ab`
`Cd`
`eF`
`GH`
`];`
	Таблица **CaseLower** содержит следующие значения:
	`ab`
	`cd`
	`ef`
	`gh`
	Все значения записываются в нижнем регистре.

Пример	Результат

В данном примере показано принудительное использование смешанного регистра.
 	
`FORCE Case Mixed;`
`CaseMixed:`
`LOAD * Inline [`
`ab`
`Cd`
`eF`
`GH`
`];`
	Таблица **CaseMixed** содержит следующие значения:
	
	`ab`
	`Cd`
	`eF`
	`GH`
	Все значения отображаются в том же виде, что и в скрипте.

См. также:

From

Ключевое слово скрипта **From** используется в операторах **Load** для связи с файлом, а также в операторах **Select** для связи с видом или таблицей базы данных.

Load

Оператор **LOAD** загружает поля из файла, из определенных в скрипте данных, из ранее загруженной таблицы, из веб-страницы, из результата последующего оператора **SELECT** или путем создания данных. Также можно загружать данные из аналитических подключений.

Синтаксис:

```
LOAD [ distinct ] fieldlist
[( from file [ format-spec ] |
from_field fieldassource [format-spec]|
inline data [ format-spec ] |
resident table-label |
autogenerate size ) |extension pluginname.functionname([script]
tabledescription)]
[ where criterion | while criterion ]
[ group by groupbyfieldlist ]
[order by orderbyfieldlist ]
```

Аргументы:

Аргументы

Аргумент	Описание
distinct	Используйте **distinct** в качестве предиката, если необходимо загрузить только уникальные записи. При наличии повторяющихся записей будет загружен первый экземпляр. При использовании предыдущих загрузок поместите **distinct** в первый оператор LOAD, так как **distinct** затрагивает только таблицу назначения.

Аргумент	Описание
fieldlist	*fieldlist ::= (* | field {, * | field })* Список полей, которые необходимо загрузить. Символ * в качестве списка полей обозначает все поля таблицы. *field ::= (fieldref | expression)* [**as** *aliasname]* Определение поля должно всегда содержать литерал, ссылку на существующее поле или выражение. *fieldref ::= (fieldname* |*@fieldnumber* |*@startpos:endpos [I | U | R | B* | **T**]) *fieldname* – это текст, идентичный имени поля в таблице. Обратите внимание, что для указания имени поля необходимо заключить его в прямые двойные кавычки или квадратные скобки, если имя содержит пробелы. Иногда имена полей явно недоступны. В таких случаях используется другая нотация: *@fieldnumber* представляет номер поля в табличном файле с разделителями. Он должен быть положительным целым числом с предшествующим символом «@». Нумерация всегда начинается с 1 и идет до числа полей. *@startpos*:*endpos* представляет начальную и конечную позиции поля в файле с записями фиксированной длины. Позиции должны быть положительными целыми числами. Двум числам должен предшествовать символ «@», и они должны быть разделены двоеточием. Нумерация всегда начинается с 1 и содержит число позиций. В последнем поле элемент **n** используется как конечная позиция. • Если после *@startpos*:*endpos* указаны символы **I** или **U**, прочитанные байты будут интерпретированы как двоичное целое число со знаком (**I**) или без знака (**U**) (порядок байтов Intel). Прочитанное число позиций должно быть 1, 2 или 4. • Если после *@startpos*:*endpos* указан символ **R**, прочитанные байты будут интерпретированы как двоичное действительное число (32-разрядное IEEE или 64-разрядное с плавающей запятой). Прочитанное число позиций должно быть 4 или 8. • Если после *@startpos*:*endpos* указан символ **B**, прочитанные байты будут интерпретироваться как числа в двоичной кодировке BCD (Binary Coded Decimal) в соответствии со стандартом COMP-3. Может быть указано любое число байтов. *expression* может быть числовой или строковой функцией на основе одного или нескольких других полей в этой же таблице. Дополнительные сведения см. в справке по синтаксису выражений. **as** используется для назначения полю нового имени.

Аргумент	Описание
from	Элемент **from** используется, если данные должны быть загружены из файла с помощью папки или подключения к данным веб-файла. *file ::= [path] filename* **Пример:** ***'lib://Table Files/'*** Если путь отсутствует, программа Qlik Sense выполняет поиск файла в каталоге, указанном оператором **Directory**. Если оператора **Directory** нет, программа Qlik Sense выполняет поиск в рабочем каталоге *C:\Users\{user}\Documents\Qlik\Sense\Apps*. *При установке на сервере Qlik Sense рабочий каталог указывается в программе Qlik Sense Repository Service, по умолчанию это C:\ProgramData\Qlik\Sense\Apps.* Элемент *filename* может содержать стандартные знаки подстановки DOS (* и ?). В результате будут загружены все файлы в указанном каталоге, удовлетворяющие критериям. *format-spec ::= (fspec-item { , fspec-item })* Спецификация формата состоит из списка нескольких элементов спецификации формата, заключенных в скобки. **Устаревший режим написания скриптов** В прежней версии режима написания скриптов следующие форматы пути тоже поддерживаются: • абсолютный **Пример:** ***c:\data*** • относительно рабочего каталога приложения Qlik Sense. **Пример:** ***data*** • URL-адрес (HTTP или FTP), указывающий на местоположение в Интернете или интрасети. **Пример:** ***http://www.qlik.com***

Аргумент	Описание
from_field	**from_field** используется в случае, если данные должны быть загружены из ранее загруженного поля. *fieldassource::=*(*tablename, fieldname)* Поле – это имя ранее загруженных *tablename* и *fieldname*. *format-spec ::= (fspec-item {, fspec-item })* Спецификация формата состоит из списка нескольких элементов спецификации формата, заключенных в скобки.
inline	**inline** используется в случае, если данные должны быть введены в скрипте, а не загружены из файла. *data ::= [text]* Данные, введенные с использованием предложения **inline**, должны быть заключены в двойные или в квадратные скобки. Текст между ними интерпретируется так же, как и содержимое файла. Поэтому при вставке новой строки в текстовый файл ее также необходимо вставить в текст предложения **inline**, например, нажав клавишу Enter при вводе скрипта. Количество столбцов определяется по первой строке. *format-spec ::= (fspec-item {, fspec-item })* Спецификация формата состоит из списка нескольких элементов спецификации формата, заключенных в скобки.
resident	Элемент **resident** используется в случае, если данные должны быть загружены из ранее загруженной таблицы. *table label* – это метка, предшествующая оператору(-ам) **LOAD** или **SELECT**, используемым для создания исходной таблицы. В конце метки должно быть указано двоеточие.
autogenerate	**autogenerate** используется в случае, если данные должны быть автоматически созданы программой Qlik Sense. *size ::= number* *Number* – это целое число, обозначающее число создаваемых записей. В списке полей не должны присутствовать выражения, требующие данные из внешнего источника данных или ранее загруженной таблицы, пока вы не обратитесь к отдельному значению поля в ранее загруженной таблице с помощью функции **Peek**.

Аргумент	Описание
extension	Можно загружать данные из аналитических подключений. Предложение **extension** можно использовать для вызова функции, определенной в подключаемом модуле серверного расширения (SSE), либо для оценки скрипта. Если отправить одну таблицу в подключаемый модуль SSE, будет возвращена одна таблица данных. Если подключаемый модуль не указал имена возвращенных полей, полям будут присвоены имена начиная с Field1, Field2. `Extension pluginname.functionname( tabledescription );` • Загрузка данных при помощи функции подключаемого модуля SSE *tabledescription ::= (table { ,tablefield})* Если порядок полей таблицы не указан, поля будут использоваться в порядке загрузки. • Загрузка данных при помощи оценки скрипта в подключаемом модуле SSE *tabledescription ::= (script, table { ,tablefield})* **Обработка типов данных в определении поля таблицы** Типы данных в аналитических подключениях определяются автоматически. Если в составе данных отсутствуют числовые значения, однако содержится по крайней мере одна текстовая строка, отличная от NULL, поле считается текстовым. В противном случае поле считается числовым. Чтобы принудительно использовать тип данных, заключите имя поля в **String()** или **Mixed()**. • **String()** принудительно задает поле как текстовое. Если поле является числовым, текстовая часть двойного значения извлекается без выполнения преобразования. • **Mixed()** принудительно задает поле как двойное. **String()** или **Mixed()** нельзя использовать вне определений поля таблицы **extension** не поддерживается, как не поддерживается использование других функций Qlik Sense в определении поля таблицы. **Подробные сведения об аналитических подключениях** Перед использованием аналитических подключений их необходимо настроить.

Аргумент	Описание
where	**where** – предложение, которое используется для указания того, нужно ли включить запись в выборку или нет. Выборка включается, если элемент *criterion* имеет значение True. *criterion* – это логическое выражение.
while	**while** – это предложение, используемое для указания необходимости повторного чтения записи. Эта же запись читается, если для элемента *criterion* указано значение True. Чтобы быть полезным, предложение **while** обычно должно содержать функцию **IterNo( )**. *criterion* – это логическое выражение.
group by	**group by** – это выражение, используемое для определения полей данных для агрегирования (группировки). Поля агрегирования должны быть включены таким же образом в загруженные выражения. Вне функций агрегирования в загруженных выражениях могут использоваться только поля агрегирования. *groupbyfieldlist ::= (fieldname { ,fieldname })*
order by	**order by** – это предложение, используемое для сортировки записей резидентной таблицы до их обработки оператором **load**. Резидентная таблица может быть отсортирована по одному или нескольким полям в возрастающем или убывающем порядке. Сортировка осуществляется первично по числовому значению и дополнительно в порядке соответствия национальных параметров. Это предложение может использоваться, только если источником данных является резидентная таблица. Поля упорядочения указывают поле для сортировки резидентной таблицы. Поле может быть указано по имени или по числу в резидентной таблице (первое поле имеет номер 1). orderbyfieldlist ::= fieldname [sortorder] { , fieldname [sortorder] } *sortorder* имеет значение *asc* для сортировки по возрастанию или *desc* для сортировки по убыванию. Если *sortorder* не указан, используется *asc*. *fieldname*, *path*, *filename* и *aliasname* – это текстовые строки, представляющие подразумеваемые соответствующие имена. Любое поле в исходной таблице может использоваться в качестве *fieldname*. Однако поля, созданные с помощью предложения (*aliasname*), не рассматриваются и не могут использоваться внутри одного оператора **load**.

Если источник данных не указан с помощью выражений **from**, **inline**, **resident**, **from_field, extension** или **autogenerate**, данные будут загружены из результата сразу после выполнения оператора **SELECT** или **LOAD**. Последующий оператор не должен иметь префикса.

Примеры:

Загрузка различных форматов файлов
Загрузка файла данных с разделителями с параметрами по умолчанию:

```
LOAD * from data1.csv;
```

Загрузка файла данных с разделителями из подключения к библиотеке (DataFiles):

```
LOAD * from 'lib://DataFiles/data1.csv';
```

Загрузка всех файлов данных с разделителями из подключения к библиотеке (DataFiles):

```
LOAD * from 'lib://DataFiles/*.csv';
```

Загрузка файла с разделителями с точкой в качестве разделителя и со встроенными метками:

```
LOAD * from 'c:\userfiles\data1.csv' (ansi, txt, delimiter is ',', embedded labels);
```

Загрузка файла с разделителями с табуляцией в качестве разделителя и со встроенными метками:

```
LOAD * from 'c:\userfiles\data2.txt' (ansi, txt, delimiter is '\t', embedded labels);
```

Загрузка файла dif со встроенными заголовками:

```
LOAD * from file2.dif (ansi, dif, embedded labels);
```

Загрузка трех полей из файла с фиксированными записями без заголовков:

```
LOAD @1:2 as ID, @3:25 as Name, @57:80 as City from data4.fix (ansi, fix, no labels, header is
0, record is 80);
```

Загрузка файла QVX, указывающего абсолютный путь:

```
LOAD * from C:\qdssamples\xyz.qvx (qvx);
```

Загрузка веб-файлов
Загрузка при помощи URL-адреса по умолчанию, указанного в подключении к данным веб-файла:

```
LOAD * from [lib://MyWebFile];
```

Загрузка при помощи определенного URL-адреса с переопределением URL-адреса, указанного в подключении к данным веб-файла:

```
LOAD * from [lib://MyWebFile] (URL is 'http://localhost:8000/foo.bar');
```

Загрузка при помощи определенного URL-адреса, указанного в значении переменной с расширением со знаком доллара:

```
SET dynamicURL = 'http://localhost/foo.bar';
LOAD * from [lib://MyWebFile] (URL is '$(dynamicURL)');
```

Выбор определенных полей, переименование и вычисление полей
Загрузка только трех указанных полей из файла с разделителями:

```
LOAD FirstName, LastName, Number from data1.csv;
```

Переименование первого поля в A, а второго в B при загрузке файла без меток:

```
LOAD @1 as A, @2 as B from data3.txt (ansi, txt, delimiter is '\t', no labels);
```

Загрузка Name путем объединения FirstName, символа пробела и LastName:

```
LOAD FirstName&' '&LastName as Name from data1.csv;
```

Загрузка Quantity, Price и Value (продукт Quantity и Price):

```
LOAD Quantity, Price, Quantity*Price as Value from data1.csv;
```

Выбор определенных записей
Загрузка только уникальных записей, дубликаты будут удалены:

```
LOAD distinct FirstName, LastName, Number from data1.csv;
```

Загрузка только записей, где поле Litres имеет значение больше нуля:

```
LOAD * from Consumption.csv where Litres>0;
```

Загрузка данных не из файла и автоматически генерируемых данных
Загрузка таблицы с встроенными данными, двух полей с именами CatID и Category:

```
LOAD * Inline
[CatID, Category
0,Regular
1,Occasional
2,Permanent];
```

Загрузка таблицы с встроенными данными, трех полей с именами UserID, Password и Access:

```
LOAD * Inline [UserID, Password, Access
A, ABC456, User
B, VIP789, Admin];
```

Загрузка таблицы с 10 000 строк. Поле A будет содержать количество прочитанных записей (1,2,3,4,5...), а поле B будет содержать произвольное число в диапазоне от 0 до 1:

```
LOAD RecNo( ) as A, rand( ) as B autogenerate(10000);
```

Скобки после элемента autogenerate допускаются, но необязательны.

Загрузка данных из ранее загруженной таблицы
Сначала мы загружаем табличный файл с разделителями и присваиваем ему имя tab1:

```
tab1:
SELECT A,B,C,D from 'lib://DataFiles/data1.csv';
```

Загрузка полей из уже загруженной таблицы tab1 в таблицу tab2:

```
tab2:
LOAD A,B,month(C),A*B+D as E resident tab1;
```

Загрузка полей из уже загруженной таблицы tab1, но только записей, где A больше B:

```
tab3:
LOAD A,A+B+C resident tab1 where A>B;
```

Загрузка полей из уже загруженной таблицы tab1, сортированных по значению A:

```
LOAD A,B*C as E resident tab1 order by A;
```

Загрузка полей из уже загруженной таблицы tab1, сортированных по первому полю, а затем по второму полю:

```
LOAD A,B*C as E resident tab1 order by 1,2;
```

Загрузка полей из уже загруженной таблицы tab1, сортированных по значению C в порядке убывания, затем по значению B в порядке возрастания, а затем по первому полю в порядке убывания:

```
LOAD A,B*C as E resident tab1 order by C desc, B asc, 1 desc;
```

Загрузка данных из ранее загруженных полей
Загрузка поля Types из ранее загруженной таблицы Characters в качестве A:

```
LOAD A from_field (Characters, Types);
```

Загрузка данных из следующей таблицы (предварительная загрузка)
Загрузка полей A, B, а также вычисляемых полей X и Y из таблицы Table1, которая загружается в следующем операторе **SELECT**:

```
LOAD A, B, if(C>0,'positive','negative') as X, weekday(D) as Y;
SELECT A,B,C,D from Table1;
```

Группировка данных
Загрузка полей, группированных (агрегированных) по значению ArtNo:

```
LOAD ArtNo, round(Sum(TransAmount),0.05) as ArtNoTotal from table.csv group by ArtNo;
```

Загрузка полей, группированных (агрегированных) по значениям Week и ArtNo:

```
LOAD Week, ArtNo, round(Avg(TransAmount),0.05) as WeekArtNoAverages from table.csv group by
Week, ArtNo;
```

Последовательное чтение одной записи
В этом примере имеется входной файл Grades.csv, содержащий оценки для каждого студента, собранные в одном поле:

```
Student,Grades
Mike,5234
```

```
John,3345
Pete,1234
Paul,3352
```

Оценки по 5-балльной шкале выставлены по предметам: Math, English, Science и History. Оценки можно выделить в отдельные значения путем многократного считывания каждой записи с помощью предложения **while**, использующего функцию **IterNo( )** в качестве счетчика. При каждом считывании оценка извлекается функцией **Mid** и сохраняется в значении Grade, а предмет выбирается с помощью функции **pick** и сохраняется в значении Subject. Конечное предложение **while** содержит проверку на считывание всех оценок (четыре на студента в данном случае), что означает необходимость считывания записи о следующем студенте.

```
MyTab:
LOAD Student,
mid(Grades,IterNo( ),1) as Grade,
pick(IterNo( ), 'Math', 'English', 'Science', 'History') as Subject from Grades.csv
while IsNum(mid(Grades,IterNo(),1));
```

Результатом будет таблица, содержащая следующие данные:

Student	Subject	Grade
John	English	3
John	History	5
John	Math	3
John	Science	4
Mike	English	2
Mike	History	4
Mike	Math	5
Mike	Science	3
Paul	English	3
Paul	History	2
Paul	Math	3
Paul	Science	5
Pete	English	2
Pete	History	4
Pete	Math	1
Pete	Science	3

Загрузка из аналитических подключений

Используются следующие данные образца.

```
Values:
Load
 Rand() as A,
 Rand() as B,
 Rand() as C
AutoGenerate(50);
```

Загрузка данных при помощи функции

В этих примерах считается, что подключаемый модуль аналитического подключения с именем *P* содержит пользовательскую функцию *Calculate(Parameter1, Parameter2)*. Функция возвращает таблицу *Results*, содержащую поля *Field1* и *Field2*.

```
Load * Extension P.Calculate( Values{A, C} );
```
Загрузка всех полей, возвращенных при отправке полей A и C в функцию.

```
Load Field1 Extension P.Calculate( Values{A, C} );
```
Загрузка только поля Field1 при отправке полей A и C в функцию.

```
Load * Extension P.Calculate( Values );
```
Загрузка всех полей, возвращенных при отправке полей A и B в функцию. Так как поля не указаны, используются поля A и B, находящиеся на первом месте в таблице.

```
Load * Extension P.Calculate( Values {C, C});
```
Загрузка всех полей, возвращенных при отправке поля C в оба параметра функции.

```
Load * Extension P.Calculate( Values {String(A), Mixed(B)});
```
Загрузка всех полей, возвращенных при отправке в функцию поля A, принудительно обозначенного как строковое, и поля B, принудительно обозначенного как числовое.

Загрузка данных при помощи оценки скрипта

```
Load A as A_echo, B as B_echo Extension R.ScriptEval( 'q;', Values{A, B} );
```
Загрузка таблицы, возвращенной скриптом q при отправке значений A и B.

```
Load * Extension R.ScriptEval( '$(My_R_Script)', Values{A, B} );
```
Загрузка таблицы, возвращенной скриптом, сохраненным в переменной My_R_Script при отправке значений A и B.

```
Load * Extension R.ScriptEval( '$(My_R_Script)', Values{B as D, *} );
```
Загрузка таблицы, возвращенной скриптом, сохраненным в переменной My_R_Script при отправке значений B, переименованных в D, A и C. Использование * приводит к отправке оставшихся полей без ссылки.

При вводе расширения файла подключений DataFiles учитывается регистр. Например: .qvd.

Элементы спецификации формата

Каждый элемент спецификации формата задает определенное свойство табличного файла:

fspec-item ::= [**ansi | oem | mac | UTF-8 | Unicode | txt | fix | dif | biff | ooxml | html | xml | kml | qvd | qvx | delimiter is** `char` | **no eof | embedded labels | explicit labels | no labels | table is [**`tablename`**]** | **header is** `n` | **header is** `line` | **header is** `n` **lines** | **comment is** `string` | **record is** `n` | **record is** `line` | **record is** `n` **lines** | **no quotes |msq** | **URL is** `string` | **userAgent is** `string`]

Набор символов

Набор символов – это спецификатор файла для оператора **LOAD**, который определяет набор символов, используемый в файле.

Спецификаторы **ansi**, **oem** и **mac** использовались в программе QlikView и все еще работают. Но они не будут генерироваться при создании оператора **LOAD** с помощью программы Qlik Sense.

Синтаксис:

```
utf8 | unicode | ansi | oem | mac | codepage is
```

Аргументы:

Аргументы

Аргумент	Описание
utf8	Набор символов UTF-8
unicode	Набор символов Unicode
ansi	Windows, кодовая страница 1252
oem	DOS, OS/2, AS400 и другие
mac	Кодовая страница 10000
codepage is	Со спецификатором **codepage** можно использовать любую кодовую страницу Windows как *N*.

Ограничения:

Преобразование из набора символов **oem** не реализовано для MacOS. Если не выбран ни один набор, используется кодовая страница 1252 для Windows.

Пример:

```
LOAD * from a.txt (utf8, txt, delimiter is ',' , embedded labels)
LOAD * from a.txt (unicode, txt, delimiter is ',' , embedded labels)
LOAD * from a.txt (codepage is 10000, txt, delimiter is ',' , no labels)
```

См. также:

Load (page 103)

Формат таблицы

Формат таблицы – это спецификатор файла для оператора **LOAD**, который определяет тип файла. Если ничего не было указано, то используется формат *.txt*.

Типы формата таблицы

Type	Описание
txt	В текстовом файле с разделителями столбцы в таблице разделены символом разделителя.
fix	В файле с записями фиксированной длины каждое поле ограничено точным числом символов. Обычно многие файлы с фиксированной длиной содержат записи, разделенные символом перевода строки. Но существует много других вариантов, как указать размер записи в байтах или охватить более одной линии с помощью **Record is**. *Если данные содержат многобайтовые символы, разрывы полей могут сместиться, поскольку этот формат основан на фиксированной длине в байтах.*
dif	В файле *.dif* (Data Interchange Format – формат обмена данными) для определения таблицы используется особый формат.
biff	Программа Qlik Sense может также интерпретировать данные в стандартных файлах Excel средствами формата *biff* (Binary Interchange File Format).
ooxml	Для файлов Excel 2007 и более поздних версий используется формат ooxml *.xslx*.
html	Если таблица является частью html-страницы или файла, используйте формат html.
xml	xml (расширяемый язык разметки) – это обычный язык разметки, используемый для представления структур данных в текстовом формате.
qvd	Формат *qvd* представляет собой собственный формат файлов QVD, экспортируемых из приложения Qlik Sense.
qvx	Формат *qvx* представляет собой формат файла или потока для высокоэффективной передачи в программу Qlik Sense.

Delimiter is

Для табличных файлов с разделителями можно указать произвольный разделитель с помощью описателя **delimiter is**. Этот описатель применяется только к файлам с разделителем формата .txt.

Синтаксис:

```
delimiter is char
```

Аргументы:

Аргументы

Аргумент	Описание
char	Указывает один символ из 127 ASCII символов.

Могут использоваться следующие значения:

Дополнительные значения

Значение	Описание
'\t'	представляет знак табуляции и указывается с кавычками или без них.
'\\'	представляет обратную косую черту (\ ).
'spaces'	представляет все комбинации одного или нескольких пробелов. Непечатные символы ASCII с кодом менее 32, за исключением CR и LF, будут интерпретироваться как пробелы.

Если ничего не указано, используется **delimiter is ','**.

Пример:

```
LOAD * from a.txt (utf8, txt, delimiter is ',' , embedded labels);
```

См. также:

Load (page 103)

No eof

Спецификатор **no eof** используется для игнорирования символа конца файла при загрузке файлов с разделителями в формате **.txt**.

Синтаксис:

```
no eof
```

Если используется спецификатор **no eof**, символы с кодовой точкой 26, которая в противном случае обозначает конец файла, игнорируются и могут быть частью значения поля.

Этот спецификатор применяется только к текстовым файлам с разделителями.

Пример:

```
LOAD * from a.txt (txt, utf8, embedded labels, delimiter is ' ', no eof);
```

См. также:

- *Load (page 103)*

Labels

Labels – это спецификатор файла для оператора **LOAD**, который определяет нахождение имен полей в файле.

Синтаксис:

```
embedded labels|explicit labels|no labels
```

Имена полей могут находиться в разных местах файла. Если первая запись содержит имена полей, следует использовать **embedded labels**. Если имена полей не найдены, следует использовать **no labels**. В файлах *dif* иногда используются отдельные разделы заголовка с явными именами полей. В таких случаях следует использовать **explicit labels**. Если не выбран ни один параметр, для файлов *dif* также используются **embedded labels**.

Example 1:

```
LOAD * from a.txt (unicode, txt, delimiter is ',' , embedded labels
```

Example 2:

```
LOAD * from a.txt (codePage is 1252, txt, delimiter is ',' , no labels)
```

См. также:

- *Load (page 103)*

Header is

Задает размер заголовка в табличных файлах. Произвольная длина заголовка задается с помощью описателя **header is**. Заголовок представляет собой текстовый раздел, не используемый программой Qlik Sense.

Синтаксис:

```
header is n
header is line
header is n lines
```

Длина заголовка может быть задана в байтах (**header is n**) или строках (**header is line** или **header is n lines**). Значение **n** должно быть положительным целым числом, представляющим длину заголовка. Если ничего не указано, используется **header is 0**. Спецификатор **header is** применяется только к табличным файлам.

Пример:

Вот пример таблицы источника данных, содержащей строку с текстом заголовка, которая не должна интерпретироваться как данные программы Qlik Sense.

```
*Header line
Col1,Col2
a,B
c,D
```

С помощью спецификатора **header is 1 lines** первая линия не будет загружена как данные. В примере благодаря спецификатору **embedded labels** программа Qlik Sense интерпретирует первую неисключенную линию как содержащую метки поля.

```
LOAD Col1, Col2
FROM 'lib://files/header.txt'
(txt, embedded labels, delimiter is ',', msq, header is 1 lines);
```

В результате образуется таблица с двумя полями, Col1 и Col2.

См. также:

Load (page 103)

Record is

При использовании файлов с фиксированной длиной записи укажите длину записи с помощью описателя **record is**.

Синтаксис:

```
Record is n
Record is line
Record is n lines
```

Аргументы:

Аргументы

Аргумент	Описание
n	Указывает длину записи в байтах.
line	Указывает длину записи в качестве одной строки.
n lines	Указывает длину записи в строках, где n – это положительное целое число, представляющее длину записи.

Ограничения:

Спецификатор **record is** применяется только к файлам **fix**.

См. также:

Load (page 103)

Quotes

Элемент **Quotes** представляет собой файловый спецификатор для оператора **LOAD**, который определяет, могут ли использоваться кавычки, а также последовательность кавычек и разделителей. Только текстовые файлы.

Синтаксис:

```
no quotes
msq
```

Если спецификатор опущен, можно использовать стандартные кавычки " " или ' ', но только в том случае, если они являются первым и последним непустым символом в значении поля.

Аргументы:

Аргументы

Аргумент	Описание
no quotes	Используется, если кавычки не должны приниматься в текстовом файле.
msq	Используется для задания современного стиля кавычек, которые позволяют вводить в поля многострочное содержимое. Поля, содержащие символы конца строки, необходимо заключать в двойные кавычки. Существует одно ограничение для параметра msq: если в качестве первого или последнего символа в содержимом строки указан один символ двойных кавычек («/»), то он будет интерпретирован как начало многострочного содержимого, что в свою очередь может привести к непредсказуемым результатам при загрузке набора данных. В этом случае следует использовать стандартные кавычки для пропуска спецификатора.

XML

Этот спецификатор скрипта используется при загрузке файлов xml. Допустимые параметры для спецификатора **XML** перечислены в синтаксических правилах.

Невозможно загрузить файлы DTD в Qlik Sense.

Синтаксис:

```
xmlsimple
```

См. также:

Load (page 103)

KML

Спецификатор использует этот скрипт при загрузке файлов KML для использования в визуализации карты.

Синтаксис:

```
kml
```

Файл KML может представлять данные области (например, страны и регионы), представленные геометрическими объектами, данные линии (например, маршруты или дороги) либо данные точек (например, города и места), представленные точками в форме [шир., долг.].

URL is

Данный спецификатор скрипта служит для указания URL-адреса или подключения к данным веб-файла в ходе загрузки веб-файла.

Синтаксис:

```
URL is string
```

Аргументы:

Аргументы

Аргумент	Описание
string	Указывает URL-адрес файла для загрузки. Этот аргумент переопределяет URL-адрес, указанный в использующемся подключении к данным веб-файла.

Ограничения:

Спецификатор **URL is** применяется только к веб-файлам. Следует использовать существующее подключение к данным веб-файла.

См. также:

Load (page 103)

userAgent is

Данный спецификатор скрипта служит для указания агента пользователя браузера в ходе загрузки веб-файла.

Синтаксис:

```
userAgent is string
```

Аргументы:

Аргументы

Аргумент	Описание
string	Указывает строку агента пользователя браузера. Агент пользователя браузера "Mozilla/5.0" по умолчанию будет переопределен.

Ограничения:

Спецификатор **userAgent is** применяется только к веб-файлам.

См. также:

Load (page 103)

Let

Оператор **let** создан как дополнение к оператору **set**, используемому для определения переменных скрипта. Оператор **let**, в отличие от оператора **set**, вычисляет выражение, расположенное справа от знака «=» во время выполнения скрипта до присваивания его переменной.

Синтаксис:

```
Let variablename=expression
```

Примеры и результаты:

Пример	Результат

`Set x=3+4;`
 	
`Let y=3+4;`
 	
`z=$(y)+1;`
	`$(x)` будет вычислено как '3+4'
	
	`$(y)` будет вычислено как ' 7'
	
	`$(z)` будет вычислено как ' 8'
	
	Обратите внимание на различие между операторами **Set** и **Let**. Оператор **Set** назначает строку '3+4' переменной, а оператор **Let** вычисляет строку и назначает переменной результат (7).

Let T=now( );	`$(T)` получит значение текущего времени.

Loosen Table

Одну или несколько внутренних таблиц данных в программе Qlik Sense можно явно объявить слабосвязанными в ходе выполнения скрипта с помощью оператора **Loosen Table**. При преобразовании таблицы в слабосвязанную все связи между значениями полей в таблице удаляются. Похожего эффекта можно добиться, загрузив каждое поле слабосвязанной таблицы в

качестве независимой несвязанной таблицы. Слабосвязанная таблица может применяться в ходе проверки для временной изоляции различных частей структуры данных. Слабосвязанная таблица обозначена в обозревателе таблиц пунктирной линией. Использование одного или нескольких операторов **Loosen Table** в скрипте приведет к тому, что программа Qlik Sense будет игнорировать параметры таблиц, считая их ставшими слабосвязанными до выполнения скрипта.

Синтаксис:

```
Loosen Tabletablename [ , tablename2 ...]
Loosen Tablestablename [ , tablename2 ...]
```

Может использоваться следующий синтаксис: **Loosen Table** или **Loosen Tables**.

Если приложение Qlik Sense обнаруживает в структуре данных циклическую ссылку, которая не может быть разорвана таблицами, объявленными как слабосвязанные, в интерактивном или явном режиме в скрипте, то одна или несколько дополнительных таблиц будут считаться слабосвязанными до тех пор, пока не исчезнет такая циклическая связь. Если это произошло, в диалоговом окне ***Предупреждение о цикле*** *появится предупреждение.*

Пример:

```
Tab1:
SELECT * from Trans;
Loosen Table Tab1;
```

Map

Оператор **map ... using** используется для сопоставления определенных значений полей или выражений со значениями в определенной таблице сопоставления. Таблицу сопоставления можно создать с помощью оператора **Mapping**.

Синтаксис:

```
Map fieldlist Using  mapname
```

Автоматическое сопоставление выполняется для полей, загруженных после выполнения оператора **Map … Using** вплоть до конца выполнения скрипта или появления оператора **Unmap**.

Сопоставление в цепочке событий, заканчивающейся сохранением поля во внутренней таблице Qlik Sense, выполняется в последнюю очередь. Таким образом, сопоставление выполняется не при каждом появлении имени поля в выражении, а тогда, когда значение сохранено во внутренней таблице под определенным именем поля. Если необходимо выполнить сопоставление на уровне выражения, используйте функцию **Applymap()**.

Аргументы:

Аргументы

Аргумент	Описание
fieldlist	Разделенный запятыми список полей, которые следует сопоставить, начиная с этой точки выполнения скрипта. Символ * в качестве списка полей обозначает все поля. В именах полей разрешается использовать знаки подстановки * и ?. При использовании знаков подстановки, возможно, понадобится заключать имена полей в кавычки.
mapname	Имя таблицы сопоставления, считанной ранее в операторе **mapping load** или **mapping select**.

Примеры и результаты:

Пример	Результат
Map Country Using Cmap;	Позволяет выполнять сопоставление поля Country с помощью карты Cmap.
Map A, B, C Using X;	Позволяет выполнять сопоставление полей A, B и C с помощью карты X.
Map * Using GenMap;	Позволяет сопоставлять все поля с помощью элемента GenMap.

NullAsNull

Оператор **NullAsNull** отключает преобразование значений NULL в строчные значения, ранее заданные с помощью оператора **NullAsValue**.

Синтаксис:

```
NullAsNull *fieldlist
```

Оператор **NullAsValue** работает как переключатель и может быть включен/выключен несколько раз в рамках скрипта с помощью оператора **NullAsValue** или **NullAsNull**.

Аргументы:

Аргументы

Аргумент	Описание
*fieldlist	Список полей, разделенных запятыми, для которых следует включить **NullAsNull**. Символ * в качестве списка полей обозначает все поля. В именах полей разрешается использовать знаки подстановки * и ?. При использовании знаков подстановки, возможно, понадобится заключать имена полей в кавычки.

Пример:

```
NullAsNull A,B;
LOAD A,B from x.csv;
```

NullAsValue

Оператор **NullAsValue** указывает, для каких из полей обнаруженные значения NULL должны быть преобразованы в значения.

Синтаксис:

```
NullAsValue *fieldlist
```

По умолчанию программа Qlik Sense рассматривает значения NULL как отсутствующие или неопределенные сущности. Тем не менее, в некоторых контекстах баз данных значения NULL считаются особыми значениями, а не просто отсутствующими значениями. Связь значений NULL с другими значениями NULL, которая обычно запрещена, можно создать с помощью оператора **NullAsValue**.

Оператор **NullAsValue** работает как переключатель и выполняется для последующих операторов загрузки. Его можно снова выключить с помощью оператора **NullAsNull**.

Аргументы:

Аргументы

Аргумент	Описание
*fieldlist	Список полей, разделенных запятыми, для которых следует включить **NullAsValue**. Символ * в качестве списка полей обозначает все поля. В именах полей разрешается использовать знаки подстановки * и ?. При использовании знаков подстановки, возможно, понадобится заключать имена полей в кавычки.

Пример:

```
NullAsValue A,B;
Set NullValue = 'NULL';
LOAD A,B from x.csv;
```

Qualify

Оператор **Qualify** используется для включения квалификации имен полей, т. е. имена полей получат имя таблицы в качестве префикса.

Синтаксис:

```
Qualify *fieldlist
```

Автоматическое объединение полей с одинаковыми именами в разных таблицах можно отключить с помощью оператора **qualify**, который уточняет имя поля с помощью имени таблицы. В случае уточнения имена полей будут изменены после их нахождения в таблице. Новое имя будет иметь вид *tablename.fieldname*. *Tablename* соответствует метке текущей таблицы или, при отсутствии метки, имени после слова **from** в операторах **LOAD** и **SELECT**.

Уточнение будет выполнено для всех полей, загруженных после оператора **qualify**.

Когда запускается скрипт, функция уточнения всегда отключена по умолчанию. Уточнение имени поля можно включить в любое время с помощью оператора **qualify**. Уточнение можно выключить в любое время с помощью оператора **Unqualify**.

*Оператор **qualify** запрещается использовать в контексте частичной перезагрузки.*

Аргументы:

Аргументы

Аргумент	Описание
*fieldlist	Список полей, разделенных запятыми, для которых следует включить уточнение. Символ * в качестве списка полей обозначает все поля. В именах полей разрешается использовать знаки подстановки * и ?. При использовании знаков подстановки, возможно, понадобится заключать имена полей в кавычки.

Example 1:

```
Qualify B;
LOAD A,B from x.csv;
LOAD A,B from y.csv;
```

Две таблицы **x.csv** и **y.csv** связываются только через **A**. В результате будет три поля: A, x.B, y.B.

Example 2:

При работе с неизвестной базой данных сначала полезно убедиться в том, что связаны только одно или несколько полей, как показано в данном примере:

```
qualify *;
unqualify TransID;
SQL SELECT * from tab1;
SQL SELECT * from tab2;
SQL SELECT * from tab3;
```

Для связей между таблицами *tab1*, *tab2* и *tab3* будет использоваться только **TransID**.

Rem

Оператор **rem** служит для вставки замечаний или комментариев в скрипт или для временного отключения операторов скрипта без их удаления.

Синтаксис:

```
Rem string
```

Весь текст между элементом **rem** и следующей точкой с запятой **;** считается комментарием.

В скрипт можно добавить комментарии двумя другими способами:

1. Можно создать комментарий в любом месте в скрипте, за исключением текста между двумя кавычками, для чего необходимо заключить необходимый фрагмент в символы **/*** и ***/**.
2. При вводе **//** в скрипте весь последующий текст справа в той же строке становится комментарием. (Обратите внимание на исключение //:, которое обычно является частью интернет-адреса).

Аргументы:

Аргументы

Аргумент	Описание
string	Произвольный текст.

Пример:

```
Rem ** This is a comment **;
/* This is also a comment */
// This is a comment as well
```

Rename

Ключевое слово скрипта **Rename** можно использовать для переименовывания уже загруженных таблиц или полей.

Rename field

Эта функция скрипта переименовывает одно или несколько существующих полей в программе Qlik Sense после их загрузки.

Не рекомендуется использовать одинаковые имена для переменной и поля или функции в Qlik Sense.

Может использоваться следующий синтаксис: **rename field** или **rename fields**.

Синтаксис:

```
Rename Field (using mapname | oldname to newname{ , oldname to newname })
Rename Fields (using mapname | oldname to newname{ , oldname to newname })
```

Аргументы:

Аргумент	Описание
mapname	Имя ранее загруженной таблицы сопоставления, в которой содержится одна или несколько пар старых и новых имен полей.
oldname	Старое имя поля.
newname	Новое имя поля.

Ограничения:

Два поля не могут получить одинаковые имена при переименовании.

Example 1:

```
Rename Field XAZ0007 to Sales;
```

Example 2:

```
FieldMap:
Mapping SQL SELECT oldnames, newnames from datadictionary;
Rename Fields using FieldMap;
```

Rename table

Эта функция скрипта переименовывает одну или несколько существующих внутренних таблиц в программе Qlik Sense после их загрузки.

Может использоваться следующий синтаксис: **rename table** или **rename tables**.

Синтаксис:

```
Rename Table (using mapname | oldname to newname{ , oldname to newname })
Rename Tables (using mapname | oldname to newname{ , oldname to newname })
```

Аргументы:

Аргументы

Аргумент	Описание
mapname	Имя ранее загруженной таблицы сопоставления, в которой содержится одна или несколько пар старых и новых имен таблиц.
oldname	Старое имя таблицы.
newname	Новое имя таблицы.

Ограничения:

Две таблицы с разными именами не могут получить одинаковые имена при переименовании. Скрипт отобразит сообщение об ошибке при попытке переименовать таблицу именем существующей таблицы.

Example 1:

```
Tab1:
SELECT * from Trans;
Rename Table Tab1 to Xyz;
```

Example 2:

```
TabMap:
Mapping LOAD oldnames, newnames from tabnames.csv;
Rename Tables using TabMap;
```

Search

Оператор **Search** используется для включения или исключения полей из интеллектуального поиска.

Синтаксис:

```
Search Include *fieldlist
Search Exclude *fieldlist
```

Можно использовать несколько операторов Search, чтобы обновить выборку полей, которые необходимо включить. Операторы оцениваются сверху вниз.

Аргументы:

Аргументы

Аргумент	Описание
*fieldlist	Список полей, разделенных запятыми, которые необходимо включить или исключить из поиска в интеллектуальном поиске. Символ * в качестве списка полей обозначает все поля. В именах полей разрешается использовать знаки подстановки * и ?. При использовании знаков подстановки, возможно, понадобится заключать имена полей в кавычки.

Пример:

Примеры поиска

Оператор	Описание
`Search Include *;`	Включить все поля в поиске в интеллектуальном поиске.
`Search Exclude [*ID];`	Исключить все поля, заканчивающиеся элементом ID в поиске в интеллектуальном поиске.
`Search Exclude '*ID';`	Исключить все поля, заканчивающиеся элементом ID в поиске в интеллектуальном поиске.
`Search Include ProductID;`	Включить поле ProductID в поиске в интеллектуальном поиске.

Комбинированным результатом этих трех операторов в такой последовательности будет исключение из поиска в интеллектуальном поиске всех полей, оканчивающихся элементом ID за исключением ProductID.

Section

Оператор **section** позволяет определить, следует ли рассматривать последующие операторы **LOAD** и **SELECT** в качестве данных или определения прав доступа.

Синтаксис:

```
Section (access | application)
```

Если ничего не указано, используется **section application**. Определение **section** действительно до тех пор, пока не будет создан новый оператор **section**.

Пример:

```
Section access;
Section application;
```

Select

Выбор полей из источника данных ODBC или поставщика OLE DB осуществляется с помощью стандартных операторов SQL **SELECT**. Однако то, принимаются операторы **SELECT** или нет, зависит в основном от используемого драйвера ODBC или поставщика OLE DB. Для использования оператора **SELECT** требуется открытое подключение к источнику данных.

Синтаксис:

```
Select [all | distinct | distinctrow | top n [percent] ] fieldlist

From tablelist

[where criterion ]
```

```
[group by fieldlist [having criterion ] ]

[order by fieldlist [asc | desc] ]

[ (Inner | Left | Right | Full) join tablename on fieldref = fieldref ]
```

Более того, несколько операторов **SELECT** иногда могут объединяться в один посредством использования оператора **union**:

```
selectstatement Union selectstatement
```

Оператор **SELECT** интерпретируется драйвером ODBC или поставщиком OLE DB, поэтому могут возникать отклонения от общего синтаксиса SQL в зависимости от возможностей драйверов ODBC или поставщика OLE DB, например:

- **as** иногда недопустим, то есть *aliasname* должен сразу следовать за *fieldname*.
- **as** иногда является обязательным при использовании *aliasname*.
- **distinct**, **as**,**where**, **group by**, **order by** или **union** иногда не поддерживаются.
- Драйвер ODBC иногда допускает не все различные кавычки, перечисленные выше.

*Это не полное описание оператора SQL **SELECT**! Например операторы **SELECT** могут быть вложенными, несколько объединений могут создаваться в одном операторе **SELECT**, число функций, допустимых в выражении, иногда может быть довольно большим, и т. д.*

Аргументы:

Аргументы

Аргумент	Описание
distinct	**distinct** – это логическое условие, используемое в случае, если копии комбинаций значений в выбранных полях должны быть загружены только один раз.
distinctrow	**distinctrow** – это логическое условие, используемое в случае, если копии записей в таблице источника должны быть загружены только один раз.

Аргумент	Описание
fieldlist	**fieldlist ::= (*| field) {, field }** Список полей, которые необходимо выбрать. Символ «*» в качестве списка полей обозначает все поля таблицы. **fieldlist ::= field {, field }** Список одного или нескольких полей, разделенных запятыми. **field ::= (fieldref| expression) [as aliasname]** Выражение может, к примеру, быть числовой или строковой функцией, основанной на одном или нескольких других полях. Некоторые из обычно принимаемых операторов и функций: +, -, *, /, & (объединение строк), sum (fieldname), count(fieldname), avg(fieldname)(average), month(fieldname), и т. д. Дополнительную информацию см. в документации к драйверу ODBC. **fieldref ::= [tablename.] fieldname** **tablename** и **fieldname** являются текстовыми строками, идентичными тому, что они подразумевают. Они должны быть заключены в прямые двойные кавычки, если они содержат, например, пробелы. Предложение **as** используется для назначения полю нового имени.
from	**tablelist ::= table {, table }** Список таблиц, из которых выбираются поля. **table ::= tablename [[as] aliasname]** Элемент **tablename** может быть в кавычках, а может и не быть.
where	**where** – предложение, которое используется для указания того, нужно ли включить запись в выборку или нет. **criterion** является логическим выражением, которое иногда может быть очень сложным. Некоторые из принимаемых операторов: числовые операторы и функции, **=**, **<>** или **#(** не равно), **>**, **>=**, **<**, **<=**, **and**, **or**, **not**, **exists**, **some**, **all**, **in**, а также новые операторы **SELECT**. Дополнительную информацию можно получить в документации драйвера ODBC или поставщика OLE DB.
group by	**group by** – выражение, используемое для агрегирования (группировки) нескольких записей в одну. Внутри одной группы для определенного поля все записи должны иметь одинаковое значение или поле может использоваться только изнутри выражения, например, в виде суммы или среднего значения. Выражение, основанное на одном или нескольких полях, определяется в выражении символа поля.
having	**having** – это предложение, используемое для классификации групп подобно тому, как предложение **where** используется для классификации записей.
order by	**order by** – предложение, используемое для указания порядка сортировки результирующей таблицы оператора **SELECT**.

Аргумент	Описание
join	**join** – это классификатор, который указывает, необходимо ли объединить несколько таблиц в одну. Имена полей и имена таблиц должны заключаться в кавычки, если в них содержатся пробелы или буквы из национальных наборов символов. Когда программа Qlik Sense автоматически создаст скрипт, драйвер ODBC или поставщик OLE DB, указанный в определении источника данных в операторе **Connect**, определит используемые кавычки.

Example 1:

```
SELECT * FROM `Categories`;
```

Example 2:

```
SELECT `Category ID`, `Category Name` FROM `Categories`;
```

Example 3:

```
SELECT `Order ID`, `Product ID`,

`Unit Price` * Quantity * (1-Discount) as NetSales

FROM `Order Details`;
```

Example 4:

```
SELECT `Order Details`.`Order ID`,

Sum(`Order Details`.`Unit Price` * `Order Details`.Quantity) as `Result`

FROM `Order Details`, Orders

where Orders.`Order ID` = `Order Details`.`Order ID`

group by `Order Details`.`Order ID`;
```

Set

Оператор **set** используется для определения переменных скрипта. Эти переменные можно использовать для подстановки строк, путей, драйверов и т. д.

Синтаксис:

```
Set variablename=string
```

Example 1:

```
Set FileToUse=Data1.csv;
```

Example 2:

```
Set Constant="My string";
```

Example 3:

```
Set BudgetYear=2012;
```

Sleep

Оператор **sleep** приостанавливает выполнение скрипта на указанное время.

Синтаксис:

```
Sleep n
```

Аргументы:

Аргумент	Описание
n	Задается в миллисекундах, где *n* – положительное целое число, не превышающее 3600000 (то есть 1 час). В качестве значения может выступать выражение.

Example 1:

```
Sleep 10000;
```

Example 2:

```
Sleep t*1000;
```

SQL

Оператор **SQL** позволяет отправлять произвольную команду SQL посредством подключения ODBC или OLE DB.

Синтаксис:

```
SQL sql_command
```

При отправке операторов SQL, которые обновляют базу данных, будет возвращаться ошибка, если программа Qlik Sense открыла подключение ODBC в режиме «только чтение».

Синтаксис:

```
SQL SELECT * from tab1;
```

допускается и будет предпочтительным синтаксисом для **SELECT** с целью обеспечения согласованности. Тем не менее префикс SQL для операторов **SELECT** будет необязательным.

Аргументы:

Аргумент	Описание
sql_command	Допустимая команда SQL.

Example 1:

```
SQL leave;
```

Example 2:

```
SQL Execute <storedProc>;
```

SQLColumns

Оператор **sqlcolumns** возвращает набор полей с описанием столбцов источника данных ODBC или OLE DB, с которыми выполнена операция **connect**.

Синтаксис:

```
SQLcolumns
```

Эти поля можно объединить с полями, созданными командами **sqltables** и **sqltypes**, что позволит получить представление об определенной базе данных. Ниже перечислены 12 стандартных полей:

TABLE_QUALIFIER

TABLE_OWNER

TABLE_NAME

COLUMN_NAME

DATA_TYPE

TYPE_NAME

PRECISION

LENGTH

SCALE

RADIX

NULLABLE

REMARKS

Подробное описание этих полей см. в справочном руководстве по ODBC.

Пример:

```
Connect to 'MS Access 7.0 Database; DBQ=C:\Course3\DataSrc\QWT.mbd';
SQLcolumns;
```

Некоторые драйверы ODBC могут не поддерживать эту команду. Некоторые драйверы ODBC могут создавать дополнительные поля.

SQLTables

Оператор **sqltables** возвращает набор полей с описанием таблиц источника данных ODBC или OLE DB, с которыми выполнена операция **connect**.

Синтаксис:

```
SQLTables
```

Эти поля можно объединить с полями, созданными командами **sqlcolumns** и **sqltypes**, что позволит получить представление об определенной базе данных. Ниже перечислены пять стандартных полей:

TABLE_QUALIFIER

TABLE_OWNER

TABLE_NAME

TABLE_TYPE

REMARKS

Подробное описание этих полей см. в справочном руководстве по ODBC.

Пример:

```
Connect to 'MS Access 7.0 Database; DBQ=C:\Course3\DataSrc\QWT.mbd';
SQLTables;
```

Некоторые драйверы ODBC могут не поддерживать эту команду. Некоторые драйверы ODBC могут создавать дополнительные поля.

SQLTypes

Оператор **sqltypes** возвращает набор полей с описанием типов источника данных ODBC или OLE DB, с которыми выполнена операция **connect**.

Синтаксис:

```
SQLTypes
```

Эти поля можно объединить с полями, созданными командами **sqlcolumns** и **sqltables**, что позволит получить представление об определенной базе данных. Ниже перечислены 15 стандартных полей:

TYPE_NAME

DATA_TYPE

PRECISION

LITERAL_PREFIX

LITERAL_SUFFIX

CREATE_PARAMS

NULLABLE

CASE_SENSITIVE

SEARCHABLE

UNSIGNED_ATTRIBUTE

MONEY

AUTO_INCREMENT

LOCAL_TYPE_NAME

MINIMUM_SCALE

MAXIMUM_SCALE

Подробное описание этих полей см. в справочном руководстве по ODBC.

Пример:

```
Connect to 'MS Access 7.0 Database; DBQ=C:\Course3\DataSrc\QWT.mbd';
SQLTypes;
```

Некоторые драйверы ODBC могут не поддерживать эту команду. Некоторые драйверы ODBC могут создавать дополнительные поля.

Star

Строку, которая представляет набор всех значений поля в базе данных, можно определить с помощью оператора **star**. Она влияет на последующие операторы **LOAD** и **SELECT**.

Синтаксис:

```
Star is[ string ]
```

Аргументы:

Аргументы

Аргумент	Описание
string	Произвольный текст. Обратите внимание, что при наличии в строке пробелов она должна быть заключена в кавычки. Если значение не указано, то по умолчанию используется **star is;**; то есть символ звездочки отсутствует, если он не будет указан явным образом. Это действительно до тех пор, пока не будет создан новый оператор **star**.

Оператор **Star is** не рекомендуется использовать в той части скрипта, где содержатся данные (в **Section Application**), если используется доступ к секции. При этом символ звезды для защищенных полей в части **Section Access** скрипта полностью поддерживается. В этом случае использовать явный оператор **Star is** не требуется, поскольку он всегда является неявным при доступе к секции.

Ограничения

- Использовать символ звезды с ключевыми полями (т. е. полями, которые связывают таблицы) нельзя.
- Символ звезды нельзя использовать ни с какими полями, которые затрагивает оператор **Unqualify**, поскольку это может отрицательно влиять на поля, связывающие таблицы.
- Символ звезды нельзя использовать с таблицами, которые не являются логическими, например таблицами загрузки информации и таблицами загрузки сопоставлений.
- Когда символ звезды используется в каком-либо сокращающем поле (поле, которое связывается с данными) при доступе к секции, этот символ представляет значения, указанные в данном поле раздела доступа к секции. Он не представляет другие значения, которые могут существовать в данных, но не указаны при доступе к секции.
- Символ звезды нельзя использовать с полями, которые затрагивает любая форма сокращения количества данных вне области **Доступ к секции**.

Пример

Следующий пример представляет собой извлечение из скрипта загрузки данных, описывающее доступ к секции.

```
Star is *;



Section Access;

LOAD * INLINE [

ACCESS, USERID, OMIT

ADMIN, ADMIN,

USER, USER1, SALES
```

```
USER, USER2, WAREHOUSE

USER, USER3, EMPLOYEES

USER, USER4, SALES

USER, USER4, WAREHOUSE

USER, USER5, *

];



Section Application;

LOAD * INLINE [

SALES, WAREHOUSE, EMPLOYEES, ORDERS

1, 2, 3, 4

];
```

Применяются следующие условия:

- *Star* соответствует символу *.
- Пользователь *ADMIN* видит все поля. Ничего не опускается.
- Пользователь *USER1* не видит поле *SALES*.
- Пользователь *USER2* не видит поле *WAREHOUSE* .
- Пользователь *USER3* не видит поле *EMPLOYEES*.
- Пользователь *USER4* дважды добавлен в программу; для двух полей *SALES* и *WAREHOUSE* для данного пользователя должно быть применено поле OMIT.
- К *USER5* добавлена звездочка (*), то есть все поля, перечисленные в OMIT, недоступны: пользователь *USER5* не видит поля *SALES*, *WAREHOUSE* и *EMPLOYEES* , но видит поле *ORDERS*.

Store

Оператор **Store** создает файл QVD, CSV или text.

Синтаксис:

```
Store [ fieldlist from] table into filename [ format-spec ];
```

Оператор создаст файл QVD, CSV или TXT с заданным именем.

Оператор может экспортировать поля только из одной таблицы данных. Если требуется экспортировать поля из нескольких таблиц, необходимо заранее сформировать явное объединение join в скрипте для создания таблицы данных, которую следует экспортировать.

Текстовые значения экспортируются в файл CSV в формате UTF-8. Можно указать разделитель. См. **LOAD**. Оператор **store** для файла CSV не поддерживает экспорт BIFF.

Аргументы:

Сохранить аргументы команды

Аргумент	Описание
*fieldlist::= (* | field) { , field })*	Список полей, которые необходимо выбрать. Символ «*» в качестве списка полей обозначает все поля. *field::= fieldname* [**as** *aliasname]* *fieldname* – это текст, идентичный имени поля в элементе *table*. (Обратите внимание, что для указания имени поля необходимо заключить его в прямые двойные кавычки или квадратные скобки, если имя содержит пробелы или другие нестандартные символы.) *aliasname* – альтернативное имя поля, которое предназначено для использования в результирующем файле QVD или CSV.
table	Метка скрипта, представляющая уже загруженную таблицу, которую планируется использовать в качестве источника данных.

Аргумент	Описание
filename	Имя целевого файла, включая действительный путь к существующему подключению к данным папки. **Пример:** *'lib://Table Files/target.qvd'* В прежней версии режима написания скриптов следующие форматы пути тоже поддерживаются: • абсолютный **Пример:** *c:\data\sales.qvd* • относительно рабочего каталога приложения Qlik Sense. **Пример:** *data\sales.qvd* Если путь отсутствует, программа Qlik Sense сохраняет файл в каталоге, указанном оператором **Directory**. Если оператора **Directory** нет, программа Qlik Sense сохраняет файл в рабочем каталоге *C:\Users\{user}\Documents\Qlik\Sense\Apps*.
format-spec ::=((**txt** | **qvd**))	С целью указания формата используется текст **txt** для обозначения текстовых файлов или текст **qvd** – для файлов qvd. Если формат не указан, то используется **qvd**.

Примеры:

```
Store mytable into xyz.qvd (qvd);

Store * from mytable into 'lib://FolderConnection/myfile.qvd';

Store Name, RegNo from mytable into xyz.qvd;

Store Name as a, RegNo as b from mytable into 'lib://FolderConnection/myfile.qvd';

Store mytable into myfile.txt (txt);

Store * from mytable into 'lib://FolderConnection/myfile.qvd';
```

При вводе расширения файла подключений DataFiles учитывается регистр. Например: .qvd.

Table/Tables

Ключевые слова скрипта **Table** и **Tables** используются в операторах **Drop**, **Comment** и **Rename**, также как спецификатор формата в операторах **Load**.

Tag

Этот оператор скрипта позволяет присваивать теги одному или нескольким полям или таблицам. Если делается попытка присвоить тег полю или таблице, отсутствующим в приложении, то эта операция будет проигнорирована. Если обнаружены конфликты между именами полей или тегов, то используется последнее значение.

Синтаксис:

```
Tag [field|fields] fieldlist with tagname
```

```
Tag [field|fields] fieldlist using mapname
```

```
Tag table tablelist with tagname
```

Аргументы

Аргумент	Описание
fieldlist	Одно или несколько полей, которые должны быть помечены тегами, в списке разделенных запятой значений.
mapname	Имя таблицы сопоставления, считанной ранее в операторе **mapping Load** или **mapping Select**.
tablelist	Список разделенных запятой таблиц, которые должны быть помечены тегами.
tagname	Имя тега, применяемого к полю.

Example 1:

```
tagmap:
mapping LOAD * inline [
a,b
Alpha,MyTag
Num,MyTag
];
tag fields using tagmap;
```

Example 2:

```
tag field Alpha with 'MyTag2';
```

Trace

Оператор **trace** записывает строку в окно **Ход выполнения скрипта** и в файл журнала скрипта, если тот используется. Он очень полезен для отладки. Расширение $, добавляемое к переменным, вычисляемым до оператора **trace**, позволяет настроить сообщение.

Синтаксис:

```
Trace string
```

Example 1:

Следующий оператор можно использовать сразу после оператора Load, который загружает таблицу Main.

```
Trace Main table loaded;
```

Он будет показывать текст 'Main table loaded' (Таблица Main загружена) в диалоговом окне выполнения скрипта и в файле журнала.

Example 2:

Следующие операторы можно использовать сразу после оператора Load, который загружает таблицу Main.

```
Let MyMessage = NoOfRows('Main') & ' rows in Main table';

Trace $(MyMessage);
```

Они будут отображать текст с указанием количества строк в диалоговом окне выполнения скрипта и в файле журнала, например '265,391 rows in Main table' (265 391 строка в таблице Main).

Unmap

Оператор **Unmap** деактивирует значение поля mapping, заданное предыдущим оператором **Map ... Using** для последующих загружаемых полей.

Синтаксис:

```
Unmap *fieldlist
```

Аргументы:

Аргументы

Аргумент	Описание
*fieldlist	разделенный запятыми список полей, которые не нужно больше сопоставлять, начиная с этой точки выполнения скрипта. Символ * в качестве списка полей обозначает все поля. В именах полей разрешается использовать знаки подстановки * и ?. При использовании знаков подстановки, возможно, понадобится заключать имена полей в кавычки.

Примеры и результаты:

Пример	Результат
Unmap Country;	Отключает сопоставление поля Country.
Unmap A, B, C;	Отключает сопоставление полей A, B и C.
Unmap * ;	Отключает сопоставление всех полей.

Unqualify

Оператор **Unqualify** используется для снятия уточнения имен полей, которое ранее было включено оператором **Qualify**.

Синтаксис:

```
Unqualify *fieldlist
```

Аргументы:

Аргументы

Аргумент	Описание
*fieldlist	Список полей, разделенных запятыми, для которых следует включить уточнение. Символ * в качестве списка полей обозначает все поля. В именах полей разрешается использовать знаки подстановки * и ?. При использовании знаков подстановки, возможно, понадобится заключать имена полей в кавычки. Дополнительные сведения см. в документации по оператору **Qualify**.

Example 1:

При работе с неизвестной базой данных сначала полезно убедиться в том, что связаны только одно или несколько полей, как показано в данном примере:

```
qualify *;
unqualify TransID;
SQL SELECT * from tab1;
SQL SELECT * from tab2;
SQL SELECT * from tab3;
```

Сначала квалификация включена для всех полей.
Затем квалификация выключена для **TransID**.
Для связей между таблицами *tab1*, *tab2* и *tab3* будет использоваться только **TransID**. Все остальные будут квалифицированы с именем таблицы.

Untag

Этот оператор скрипта позволяет удалять теги из полей или таблиц. Если делается попытка удалить тег из поля или таблицы, отсутствующим в приложении, то эта операция будет проигнорирована.

Синтаксис:

```
Untag [field|fields] fieldlist with tagname

Untag [field|fields] fieldlist using mapname

Untag table tablelist with tagname
```

Аргументы:

Аргументы

Аргумент	Описание
fieldlist	Одно или несколько полей, теги которых должны быть удалены, в списке разделенных запятой значений.
mapname	Имя таблицы сопоставления, загруженной ранее в оператор сопоставления **LOAD** или **SELECT**.
tablelist	Список разделенных запятой таблиц, теги которых должны быть сняты.
tagname	Имя тега, который следует снять с поля.

Example 1:

```
tagmap:
mapping LOAD * inline [
a,b
Alpha,MyTag
Num,MyTag
];
Untag fields using tagmap;
```

Example 2:

```
Untag field Alpha with MyTag2;
```

2.6 Рабочий каталог

Если в операторе скрипта есть ссылка на файл, а путь не указан, программа Qlik Sense выполняет поиск файла в следующем порядке.

1. Каталог, указанный оператором **Directory** (поддерживается только в прежней версии режима написания скриптов).
2. Если оператора **Directory** нет, программа Qlik Sense выполняет поиск в рабочем каталоге.

Рабочий каталог Qlik Sense Desktop

В Qlik Sense Desktop рабочим каталогом является *C:\Users\{user}\Documents\Qlik\Sense\Apps*.

Рабочий каталог Qlik Sense

При установке на сервере Qlik Sense рабочий каталог указывается в программе Qlik Sense Repository Service, по умолчанию это *C:\ProgramData\Qlik\Sense\Apps*. Для получения более подробной информации см. Qlik Management Console.

2 Работа с переменными в редакторе загрузки данных

Переменная в Qlik Sense является контейнером, содержащим статическое значение или вычисление, например числовое или буквенно-числовое значение. При использовании этой переменной в приложении любое изменение, выполненное в переменной, применяется везде, где эта переменная используется. Переменные можно определить в окне обзора переменных или в скрипте с помощью редактора загрузки данных. Для установки значения переменной можно использовать операторы **Let** или **Set** в скрипте загрузки данных.

При редактировании листа можно также работать с переменными Qlik Sense с помощью окна обзора переменных.

2.7 Обзор

Если первый символ в значении переменной – это знак равенства «=», то программа Qlik Sense рассчитывает значение по формуле (выражение Qlik Sense) и выводит или возвращает результат, а не визуальное написание формулы.

При использовании вместо переменной подставляется ее значение. Переменные можно использовать в скрипте для расширения со знаком доллара и в различных операторах управления. Это очень удобно, если одна и та же строка повторяется в скрипте множество раз, например путь.

В начале выполнения скрипта программа Qlik Sense устанавливает некоторые особые системные переменные независимо от их предыдущих значений.

2.8 Определение переменной

Переменные дают возможность хранить статические значения или результат вычислений. При определении переменной используйте следующий синтаксис:

```
set variablename = string
```

или

```
let variable = expression
```

Оператор **Set** используется для присвоения строки. Он присваивает переменной текст справа от знака равенства. Оператор **Let** оценивает выражение справа от знака равенства во время выполнения скрипта и присваивает результат выражения переменной.

В переменных учитывается регистр.

Не рекомендуется использовать одинаковые имена для переменной и поля или функции в Qlik Sense.

Примеры:

`set x = 3 + 4; //` переменная получит в качестве значения строку '3 + 4'.

`let x = 3 + 4; //` возвращает 7 в качестве значения.

`set x = Today(); //` возвращает 'Today()' в качестве значения.

`let x = Today(); //` возвращает в качестве значения сегодняшнюю дату, например ‘9/27/2021’.

2.9 Удаление переменной

Если удалить переменную из скрипта и перезагрузить данные, переменная будет существовать в приложении. Чтобы полностью удалить переменную из приложения, необходимо также удалить ее из диалогового окна переменных.

2.10 Загрузка значения переменной в качестве значения поля

Для загрузки значения переменной в качестве значения поля в оператор **LOAD** и получения расширения со знаком доллара в виде текста, а не числового значения или выражения, необходимо заключить развернутую переменную в одинарные кавычки.

Пример:

В этом примере выполняется загрузка системной переменной, содержащей список ошибок скрипта в таблице. Обратите внимание, что расширение ScriptErrorCount в предложении **If** не требует кавычек, в то время, как расширение ScriptErrorList необходимо заключить в кавычки.

```
IF $(ScriptErrorCount) >= 1 THEN

   LOAD '$(ScriptErrorList)' AS Error AutoGenerate 1; END IF
```

2.11 Вычисление переменной

Существует несколько способов использования переменных с вычисляемыми значениями в программе Qlik Sense. Результат зависит от того, как это будет определено и названо в выражении.

В этом примере загружаются некоторые встроенные данные:

```
LOAD * INLINE [
    Dim, Sales
    A, 150
    A, 200
```

```
    B, 240
    B, 230
    C, 410
    C, 330
];
```

Давайте определим две переменные.

```
Let vSales  = 'Sum(Sales)' ;
Let vSales2  = '=Sum(Sales)' ;
```

Во второй переменной мы добавляем знак равенства перед выражением. В результате переменная будет вычислена до того, как она будет расширена, а выражение оценено.

При использовании неизмененной переменной vSales, например, в мере, результатом будет строка Sum(Sales), то есть вычисления не будут выполнены.

В случае добавления расширения со знаком доллара и вызова элемента $(vSales) в выражении переменная будет расширена, а сумма Sales отобразится.

Наконец, если будет вызван элемент $(vSales2), вычисление переменной будет выполнено до ее расширения. Это означает, что отображаемый результат – это итоговая сумма элементов Sales. Разницу использования элементов =$(vSales) и =$(vSales2) в качестве выражений мер можно увидеть в этой диаграмме с отображением результатов:

Результаты

Dim	$(vSales)	$(vSales2)
A	350	1560
B	470	1560
C	740	1560

Как можно увидеть, элемент $(vSales) показывает частичную сумму для значения измерения, а элемент $(vSales2) показывает итоговую сумму.

Доступны следующие переменные скрипта:

- *Ошибка переменных (page 173)*
- *Переменные интерпретации числа (page 158)*
- *Системные переменные (page 149)*
- *Переменные обработки значений (page 156)*

2.12 Системные переменные

Системные переменные, некоторые из которых определяются системой, обеспечивают информацию о системе и приложении Qlik Sense.

Обзор системных переменных

Некоторые функции подробно описаны после обзора. Для этих функций можно щелкнуть имя функции в синтаксисе, чтобы получить немедленный доступ к подробной информации об этой конкретной функции.

Floppy

Возвращает буквенное обозначение первого найденного дисковода гибких дисков, обычно *a:*. Эта переменная определяется системой.

```
Floppy
```

Эта переменная не поддерживается в стандартном режиме.

CD

Возвращает буквенное обозначение первого найденного дисковода CD-ROM. Если дисковод CD-ROM не найден, возвращается *c:*. Эта переменная определяется системой.

```
CD
```

Эта переменная не поддерживается в стандартном режиме.

Include

Переменная **Include/Must_Include** указывает файл, содержащий текст, который необходимо включить в скрипт и который рассматривается в качестве кода скрипта. Она не используется для добавления данных. Можно сохранять часть кода скрипта в отдельный текстовый файл и использовать его в разных приложениях. Эта переменная определяется пользователем.

```
$(Include=filename)
$(Must_Include=filename)
```

HidePrefix

Все имена полей, начинающиеся этой строкой текста, будут скрыты так же, как и системные поля. Эта переменная определяется пользователем.

```
HidePrefix
```

HideSuffix

Все имена полей, которые заканчиваются этой строкой текста, будут скрыты так же, как и системные поля. Эта переменная определяется пользователем.

```
HideSuffix
```

QvPath

Возвращает строку обзора в выполняемый модуль Qlik Sense. Эта переменная определяется системой.

```
QvPath
```

Эта переменная не поддерживается в стандартном режиме.

QvRoot

Возвращает корневой каталог выполняемого модуля Qlik Sense. Эта переменная определяется системой.

```
QvRoot
```

Эта переменная не поддерживается в стандартном режиме.

QvWorkPath

Возвращает строку обзора в текущее приложение Qlik Sense. Эта переменная определяется системой.

```
QvWorkPath
```

Эта переменная не поддерживается в стандартном режиме.

QvWorkRoot

Возвращает корневой каталог текущего приложения Qlik Sense. Эта переменная определяется системой.

```
QvWorkRoot
```

Эта переменная не поддерживается в стандартном режиме.

StripComments

Если для этой переменной установлено значение 0, исключение комментариев /*..*/ и // в скрипте будет блокироваться. Если эта переменная не определена, всегда выполняется исключение комментариев.

```
StripComments
```

Verbatim

Обычно предшествующие и завершающие символы пробела (ASCII 32) автоматически исключаются из всех значений поля до их загрузки в базу данных Qlik Sense. Установка для этой переменной значения 1 приостанавливает исключение символов пробела. Символ табуляции (ASCII 9) и твердый пробел (ANSI 160) никогда не исключаются.

```
Verbatim
```

OpenUrlTimeout

Эта переменная определяет время ожидания в секундах, которое программа Qlik Sense использует при получении данных из источников URL (например, HTML -страниц). При отсутствии данной переменной время ожидания составляет 20 минут.

```
OpenUrlTimeout
```

WinPath

Возвращает строку обзора в Windows. Эта переменная определяется системой.

```
WinPath
```

Эта переменная не поддерживается в стандартном режиме.

WinRoot

Возвращает корневой каталог Windows. Эта переменная определяется системой.

```
WinRoot
```

Эта переменная не поддерживается в стандартном режиме.

CollationLocale

Указывает, какую локаль использовать для порядка сортировки и сопоставления поиска. Значением является имя культуры локали, например, «en-US».Эта переменная определяется системой.

```
CollationLocale
```

CreateSearchIndexOnReload

Данная переменная определяет, будут ли создаваться файлы поискового индекса в ходе повторной загрузки данных.

```
CreateSearchIndexOnReload
```

CreateSearchIndexOnReload

Данная переменная определяет, будут ли создаваться файлы поискового индекса в ходе повторной загрузки данных.

Синтаксис:

```
CreateSearchIndexOnReload
```

Можно настроить создание файлов индекса поиска в ходе повторной загрузки данных или после ввода пользователем первого поискового запроса. Преимущество создания файлов индекса поиска в ходе повторной загрузки данных заключается в устранении периода ожидания, с которым сталкивается первый пользователь, выполняющий поиск. Необходимо учесть то, что создание индекса поиска приводит к увеличению продолжительности повторной загрузки данных.

В случае пропуска данной переменной файлы индекса поиска в ходе повторной загрузки данных создаваться не будут.

Вне зависимости от значения данной переменной для приложений сеанса файлы индекса поиска в ходе повторной загрузки данных создаваться не будут.

Example 1: Создавать файлы индекса поиска в ходе повторной загрузки данных

```
set CreateSearchIndexOnReload=1;
```

Example 2: Создавать поля индекса поиска после первого поискового запроса

```
set CreateSearchIndexOnReload=0;
```

HidePrefix

Все имена полей, начинающиеся этой строкой текста, будут скрыты так же, как и системные поля. Эта переменная определяется пользователем.

Синтаксис:

```
HidePrefix
```

Пример:

```
set HidePrefix='_' ;
```

При использовании этого оператора имена полей, начинающиеся с нижнего подчеркивания, не отображаются в списках имен полей, если скрыты системные поля.

HideSuffix

Все имена полей, которые заканчиваются этой строкой текста, будут скрыты так же, как и системные поля. Эта переменная определяется пользователем.

Синтаксис:

```
HideSuffix
```

Пример:

```
set HideSuffix='%';
```

При использовании этого оператора имена полей, заканчивающиеся знаком %, не отображаются в списках имен полей, если скрыты системные поля.

Include

Переменная **Include/Must_Include** указывает файл, содержащий текст, который необходимо включить в скрипт и который рассматривается в качестве кода скрипта. Она не используется для добавления данных. Можно сохранять часть кода скрипта в отдельный текстовый файл и использовать его в разных приложениях. Эта переменная определяется пользователем.

Эта переменная поддерживает только подключения к данным в папке в стандартном режиме.

Синтаксис:

```
$(Include=filename)
```

```
$(Must_Include=filename)
```

Существует две версии переменной.

- Переменная **Include** не создает ошибку, если не удалось найти файл, и сообщение об ошибке не отображается.
- Переменная **Must_Include** создает ошибку, если не удалось найти файл.

Если не указать путь, имя файла будет отнесено к рабочему каталогу приложения Qlik Sense. Можно также указать абсолютный путь файла или путь к подключению к папке lib://. Перед знаком равенства или после него не следует ставить пробел.

*Конструкция **set Include =**filename не применяется.*

Примеры:

```
$(Include=abc.txt);
```

```
$(Must_Include=lib://DataFiles/abc.txt);
```

Ограничения

Ограниченная перекрестная совместимость между файлами в кодировке UTF-8 под Windows и Linux.

Необязательно использовать UTF-8 с BOM (Byte Order Mark, метка порядка байтов). BOM может мешать использованию UTF-8 в программах, которые не ожидают не-ASCII байтов в начале файла, но которые в противном случае могли бы обрабатывать текстовый поток.

- Системы Windows используют BOM в кодировке UTF-8, чтобы определить, что файл в кодировке UTF-8, хотя в байтовом хранилище нет неоднозначности.

- Unix/Linux используют UTF-8 для Unicode, но не используют BOM, так как это мешает синтаксису командных файлов.

Это имеет некоторые последствия для Qlik Sense.

- В Windows любой файл, начинающийся с BOM UTF-8, считается файлом скрипта UTF-8. В противном случае предполагается кодировка ANSI.
- В Linux 8-битная системная кодовая страница по умолчанию – UTF-8. Поэтому UTF-8 работает, хотя и не содержит BOM.

В результате переносимость не может быть гарантирована. Не всегда возможно создать файл под Windows, который может быть интерпретирован под Linux, и наоборот. Между двумя системами нет перекрестной совместимости по отношению к файлам в кодировке UTF-8 из-за различной работы с BOM.

OpenUrlTimeout

Эта переменная определяет время ожидания в секундах, которое программа Qlik Sense использует при получении данных из источников URL (например, HTML - страниц). При отсутствии данной переменной время ожидания составляет 20 минут.

Синтаксис:

```
OpenUrlTimeout
```

Пример:

```
set OpenUrlTimeout=10;
```

StripComments

Если для этой переменной установлено значение 0, исключение комментариев /*..*/ и // в скрипте будет блокироваться. Если эта переменная не определена, всегда выполняется исключение комментариев.

Синтаксис:

```
StripComments
```

В определенных драйверах базы данных используются /*..*/ в качестве подсказок по оптимизации в операторах **SELECT**. В таком случае комментарии не должны исключаться перед отправкой оператора **SELECT** в драйвер базы данных.

Рекомендуется сбросить эту переменную на значение 1 сразу после оператора(-ов) там, где это необходимо.

Пример:

```
set StripComments=0;
SQL SELECT * /* <optimization directive> */ FROM Table ;
```

```
set StripComments=1;
```

Verbatim

Обычно предшествующие и завершающие символы пробела (ASCII 32) автоматически исключаются из всех значений поля до их загрузки в базу данных Qlik Sense. Установка для этой переменной значения 1 приостанавливает исключение символов пробела. Символ табуляции (ASCII 9) и твердый пробел (ANSI 160) никогда не исключаются.

Синтаксис:

```
Verbatim
```

Пример:

```
set Verbatim = 1;
```

2.13 Переменные обработки значений

В этом разделе описаны переменные, которые используются для обработки значений NULL и других значений.

Обзор значений, обрабатывающих переменные

Каждая функция подробно описана после обзора. Также можно щелкнуть имя функции в синтаксисе, чтобы получить немедленный доступ к подробной информации об этой конкретной функции.

NullDisplay

Указанным символом заменяются все значения NULL из ODBC и коннекторов на самом нижнем уровне данных. Эта переменная определяется пользователем.

```
NullDisplay
```

NullInterpret

При нахождении указанного символа в текстовом файле, файле Excel или во встроенном операторе он интерпретируется как значение NULL. Эта переменная определяется пользователем.

```
NullInterpret
```

NullValue

Если используется оператор **NullAsValue**, определенный символ будет заменять все значения NULL в указанных полях **NullAsValue** указанной строкой.

```
NullValue
```

OtherSymbol
Определяет символ, который будет обрабатываться как все другие значения перед оператором **LOAD/SELECT**. Эта переменная определяется пользователем.

```
OtherSymbol
```

NullDisplay

Указанным символом заменяются все значения NULL из ODBC и коннекторов на самом нижнем уровне данных. Эта переменная определяется пользователем.

Синтаксис:

```
NullDisplay
```

Пример:

```
set NullDisplay='<NULL>';
```

NullInterpret

При нахождении указанного символа в текстовом файле, файле Excel или во встроенном операторе он интерпретируется как значение NULL. Эта переменная определяется пользователем.

Синтаксис:

```
NullInterpret
```

Примеры:

```
set NullInterpret=' ';
set NullInterpret =;
```

не возвращает значения NULL для пустых значений в Excel, (в текстовом файле CSV возвращает)

```
set NullInterpret ='';
```

возвращает значения NULL для пустых значений в Excel.

NullValue

Если используется оператор **NullAsValue**, определенный символ будет заменять все значения NULL в указанных полях **NullAsValue** указанной строкой.

Синтаксис:

```
NullValue
```

Пример:

```
NullAsValue Field1, Field2;
set NullValue='<NULL>';
```

OtherSymbol

Определяет символ, который будет обрабатываться как все другие значения перед оператором **LOAD/SELECT**. Эта переменная определяется пользователем.

Синтаксис:

```
OtherSymbol
```

Пример:

```
set OtherSymbol='+';
LOAD * inline
[X, Y
a, a
b, b];
LOAD * inline
[X, Z
a, a
+, c];
```

Значение поля Y='b' теперь будет связано с Z='c' через другой символ.

2.14 Переменные интерпретации числа

Переменные интерпретации числа определяет система, то есть они создаются автоматически при создании нового приложения в соответствии с текущими региональными настройками операционной системы. В Qlik Sense Desktop это выполняется согласно настройкам операционной системы компьютера, а в Qlik Sense это выполняется согласно операционной системе сервера, на котором установлена программа Qlik Sense.

Переменные указываются в верхней части скрипта нового приложения Qlik Sense и заменяют стандартные настройки операционной системы для определенных параметров формата чисел во время выполнения скрипта. Эти переменные можно свободно удалять, редактировать или копировать.

Если необходимо создать приложение для определенной локали, самый простой способ, возможно, это использование Qlik Sense Desktop на компьютере с требуемыми параметрами локали в операционной системе для создания приложения. В таком случае приложение будет содержать соответствующие региональные параметры этой локали, и вы можете переместить ее на сервер Qlik Sense для дальнейших разработок.

Обзор переменных интерпретации числа

Каждая функция подробно описана после обзора. Также можно щелкнуть имя функции в синтаксисе, чтобы получить немедленный доступ к подробной информации об этой конкретной функции.

Форматирование валюты

MoneyDecimalSep

Указанный десятичный разделитель заменяет символ десятичного знака для валюты, используемый в операционной системе (региональные настройки).

```
MoneyDecimalSep
```

MoneyFormat

Указанный символ заменяет символ валюты, используемый в операционной системе (региональные настройки).

```
MoneyFormat
```

MoneyThousandSep

Указанный разделитель тысяч заменяет группирующий символ знаков для валюты, используемый в операционной системе (региональные настройки).

```
MoneyThousandSep
```

Форматирование чисел

DecimalSep

Заданный десятичный разделитель заменяет символ десятичного знака, используемый в операционной системе (региональные настройки).

```
DecimalSep
```

ThousandSep

Указанный разделитель тысяч заменяет группирующий символ знаков, используемый в операционной системе (региональные настройки).

```
ThousandSep
```

NumericalAbbreviation

Числовая аббревиатура определяет, какие аббревиатуры использовать для чисел и префиксов величины, к примеру M для значений «мега» или «миллион» (10^6) и μ для значения «микро» (10^{-6}).

```
NumericalAbbreviation
```

Форматирование времени

DateFormat

Эта переменная среды определяет формат даты, используемый в приложении по умолчанию. Этот формат используется как для интерпретации, так и для форматирования дат. Если переменная не определена, при выполнении скрипта будет извлекаться формат даты из региональных настроек операционной системы.

```
DateFormat
```

TimeFormat

Указанный формат заменяет формат времени, используемый в операционной системе (региональные настройки).

```
TimeFormat
```

TimestampFormat

Указанный формат заменяет форматы даты и времени, используемые в операционной системе (региональные настройки).

```
TimestampFormat
```

MonthNames

Указанный формат заменяет обозначение имен месяцев, используемое в операционной системе (региональные настройки).

```
MonthNames
```

LongMonthNames

Указанный формат заменяет обозначение полных имен месяцев, используемое в операционной системе (региональные настройки).

```
LongMonthNames
```

DayNames

Указанный формат заменяет имена дней недели, используемые в операционной системе (региональные настройки).

```
DayNames
```

LongDayNames

Указанный формат заменяет обозначение полных имен дней недели, используемое в операционной системе (региональные настройки).

```
LongDayNames
```

FirstWeekDay

Целое число, которое определяет, какой день использовать в качестве первого дня недели.

```
FirstWeekDay
```

BrokenWeeks

Этот параметр определяет, какими должны быть недели: целыми или разбитыми.

```
BrokenWeeks
```

ReferenceDay

Этот параметр определяет, какой день в январе должен быть задан в качестве дня ссылки, чтобы определить неделю 1.

```
ReferenceDay
```

FirstMonthOfYear
С помощью этой настройки определяется месяц, который будет использован в качестве первого месяца года. Его можно использовать для определения финансовых годов, в которых используется смещение по месяцам, например, начало будет 1 апреля.

Данная настройка в настоящее время не используется, но зарезервирована для будущего использования.

Допустимые настройки: от 1 (январь) до 12 (декабрь). Параметр по умолчанию – 1.

Синтаксис:

```
FirstMonthOfYear
```

Пример:

```
Set FirstMonthOfYear=4; //Sets the year to start in April
```

BrokenWeeks

Этот параметр определяет, какими должны быть недели: целыми или разбитыми.

Синтаксис:

```
BrokenWeeks
```

По умолчанию в функциях Qlik Sense используются целые недели. Это означает следующее:

- В одних годах 1-я неделя начинается в декабре, а в других годах 52-я или 53-я неделя заканчивается в январе.
- В 1-ой неделе всегда не менее четырех дней в январе.

В качестве альтернативы можно использовать разбиение недель:

- 52-я или 53-я неделя не будет продолжена в январе следующего года.
- 1-я неделя будет начинаться 1 января и в большинстве случаев она будет неполной.

Могут использоваться следующие значения:

- 0 (= использовать целые недели)
- 1 (= использовать разбитые недели)

Примеры:

```
Set BrokenWeeks=0; //(use unbroken weeks)

Set BrokenWeeks=1; //(use broken weeks)
```

DateFormat

Эта переменная среды определяет формат даты, используемый в приложении по умолчанию. Этот формат используется как для интерпретации, так и для форматирования дат. Если переменная не определена, при выполнении скрипта будет извлекаться формат даты из региональных настроек

операционной системы.

Синтаксис:

```
DateFormat
```

Примеры:

```
Set DateFormat='M/D/YY'; //(US format)

Set DateFormat='DD/MM/YY'; //(UK date format)

Set DateFormat='YYYY-MM-DD'; //(ISO date format)
```

DayNames

Указанный формат заменяет имена дней недели, используемые в операционной системе (региональные настройки).

Синтаксис:

```
DayNames
```

Пример:

```
Set DayNames='Mon;Tue;Wed;Thu;Fri;Sat;Sun';
```

DecimalSep

Заданный десятичный разделитель заменяет символ десятичного знака, используемый в операционной системе (региональные настройки).

Синтаксис:

```
DecimalSep
```

Примеры:

```
Set DecimalSep='.';

Set DecimalSep=',';
```

FirstWeekDay

Целое число, которое определяет, какой день использовать в качестве первого дня недели.

Синтаксис:

```
FirstWeekDay
```

По умолчанию системные переменные Qlik Sense определяют `FirstWeekDay=6`. Это означает, что воскресенье является первым днем недели.

Значения, которые могут быть заданы в качестве FirstWeekDay

Значение	День
0	Понедельник
1	Вторник
2	Среда
3	Четверг
4	Пятница
5	Суббота
6	Воскресенье

LongDayNames

Указанный формат заменяет обозначение полных имен дней недели, используемое в операционной системе (региональные настройки).

Синтаксис:

```
LongDayNames
```

Пример:

```
Set LongDayNames='Monday;Tuesday;Wednesday;Thursday;Friday;Saturday;Sunday';
```

LongMonthNames

Указанный формат заменяет обозначение полных имен месяцев, используемое в операционной системе (региональные настройки).

Синтаксис:

```
LongMonthNames
```

Пример:

```
Set
LongMonthNames='January;February;March;April;May;June;July;August;September;October;November;D
ecember';
```

MoneyDecimalSep

Указанный десятичный разделитель заменяет символ десятичного знака для валюты, используемый в операционной системе (региональные настройки).

Синтаксис:

```
MoneyDecimalSep
```

Пример:

```
Set MoneyDecimalSep='.';
```

MoneyFormat

Указанный символ заменяет символ валюты, используемый в операционной системе (региональные настройки).

Синтаксис:

```
MoneyFormat
```

Пример:

```
Set MoneyFormat='$ #,##0.00; ($ #,##0.00)';
```

MoneyThousandSep

Указанный разделитель тысяч заменяет группирующий символ знаков для валюты, используемый в операционной системе (региональные настройки).

Синтаксис:

```
MoneyThousandSep
```

Пример:

```
Set MoneyThousandSep=',';
```

MonthNames

Указанный формат заменяет обозначение имен месяцев, используемое в операционной системе (региональные настройки).

Синтаксис:

```
MonthNames
```

Пример:

```
Set MonthNames='Jan;Feb;Mar;Apr;May;Jun;Jul;Aug;Sep;Oct;Nov;Dec';
```

NumericalAbbreviation

Числовая аббревиатура определяет, какие аббревиатуры использовать для чисел и префиксов величины, к примеру M для значений «мега» или «миллион» (10^6) и μ для значения «микро» (10^{-6}).

Синтаксис:

```
NumericalAbbreviation
```

Для переменной `NumericalAbbreviation` устанавливается строка, содержащая список пар определений аббревиатуры, разделенных точкой с запятой. Каждая пара определений аббревиатуры должна содержать параметр охвата (показатель степени в десятичной системе) и аббревиатуру, разделенные двоеточием, к примеру `6:M` для миллиона.

Параметр по умолчанию — `'3:k;6:M;9:G;12:T;15:P;18:E;21:Z;24:Y;-3:m;-6:µ;-9:n;-12:p;-15:f;-18:a;-21:z;-24:y'`.

Примеры:

Данный параметр заменяет префикс тысячи на t и префикс миллиарда на B. Это полезно для финансовых приложений, где встречаются аббревиатуры вида t$, M$ и B$.

```
Set NumericalAbbreviation='3:t;6:M;9:B;12:T;15:P;18:E;21:Z;24:Y;-3:m;-6:µ;-9:n;-12:p;-15:f;-18:a;-21:z;-24:y';
```

ReferenceDay

Этот параметр определяет, какой день в январе должен быть задан в качестве дня ссылки, чтобы определить неделю 1.

Синтаксис:

```
ReferenceDay
```

По умолчанию в функциях Qlik Sense используется 4 как день ссылки. Это значит, что неделя 1 должна содержать значение «январь 4» , или, другими словами, в неделе 1 всегда должно быть не меньше 4 дней в январе.

Используйте следующие значения, чтобы задать день ссылки:

- 1 (= январь 1)
- 2 (= январь 2)
- 3 (= январь 3)
- 4 (= январь 4)
- 5 (= январь 5)
- 6 (= январь 6)
- 7 (= январь 7)

Примеры:

```
Set ReferenceDay=3; //(set January 3 as the reference day)
```

ThousandSep

Указанный разделитель тысяч заменяет группирующий символ знаков, используемый в операционной системе (региональные настройки).

Синтаксис:

```
ThousandSep
```

Примеры:

```
Set ThousandSep=','; //(for example, seven billion must be specified as: 7,000,000,000)

Set ThousandSep=' ';
```

TimeFormat

Указанный формат заменяет формат времени, используемый в операционной системе (региональные настройки).

Синтаксис:

```
TimeFormat
```

Пример:

```
Set TimeFormat='hh:mm:ss';
```

TimestampFormat

Указанный формат заменяет форматы даты и времени, используемые в операционной системе (региональные настройки).

Синтаксис:

```
TimestampFormat
```

Пример:

В следующих примерах используются данные метки времени *1983-12-14T13:15:30Z* в целях демонстрации результатов применения разных операторов **SET TimestampFormat**. Используется формат даты **YYYYMMDD** и формат времени **h:mm:ss TT**. Формат даты указан в операторе **SET DateFormat**, формат времени – в операторе **SET TimeFormat** в верхней части скрипта загрузки данных.

Результаты

Пример	Результат
`SET TimestampFormat='YYYYMMDD';`	19831214
`SET TimestampFormat='M/D/YY hh:mm:ss[.fff]';`	12/14/83 13:15:30
`SET TimestampFormat='DD/MM/YYYY hh:mm:ss[.fff]';`	14/12/1983 13:15:30
`SET TimestampFormat='DD/MM/YYYY hh:mm:ss[.fff] TT';`	14/12/1983 1:15:30 PM
`SET TimestampFormat='YYYY-MM-DD hh:mm:ss[.fff] TT';`	1983-12-14 01:15:30

Примеры: Скрипт загрузки

Пример: Скрипт загрузки

В первом скрипте загрузки используется *SET TimestampFormat='DD/MM/YYYY h:mm:ss[.fff] TT'*. Во втором скрипте загрузки формат метки времени изменен на *SET TimestampFormat='MM/DD/YYYY hh:mm:ss[.fff]'*. Оператор **SET TimeFormat** дает разные результаты при использовании с различными форматами данных времени.

В таблице ниже представлен набор данных, используемый в следующих скриптах загрузки. Во втором столбце таблицы представлен формат каждой метки времени в наборе данных. Первые пять меток времени, в отличие от шестой метки, соответствуют стандарту ISO 8601.

Набор данных

Таблица, в которой представлены используемые данные о времени и формат каждой метки времени в наборе данных.

transaction_timestamp	time data format
2018-08-30	YYYY-MM-DD
20180830T193614.857	YYYYMMDDhhmmss.sss
20180830T193614.857+0200	YYYYMMDDhhmmss.sss±hhmm
2018-09-16T12:30-02:00	YYYY-MM-DDhh:mm±hh:mm
2018-09-16T13:15:30Z	YYYY-MM-DDhh:mmZ
9/30/18 19:36:14	M/D/YY hh:mm:ss

В **Редакторе загрузки данных** создайте новый раздел, добавьте образец скрипта и запустите его. Затем добавьте на лист приложения как минимум поля, указанные в столбце с результатами, чтобы увидеть результаты.

Скрипт загрузки

```
SET FirstWeekDay=0; SET BrokenWeeks=1; SET ReferenceDay=0; SET
DayNames='Mon;Tue;Wed;Thu;Fri;Sat;Sun'; SET
LongDayNames='Monday;Tuesday;Wednesday;Thursday;Friday;Saturday;Sunday'; SET
DateFormat='YYYYMMDD'; SET TimestampFormat='DD/MM/YYYY h:mm:ss[.fff] TT';   Transactions: Load
*, Timestamp(transaction_timestamp, 'YYYY-MM-DD hh:mm:ss[.fff]') as LogTimestamp ;   Load *
Inline [ transaction_id, transaction_timestamp, transaction_amount, transaction_quantity,
discount, customer_id, size, color_code 3750, 2018-08-30, 12423.56, 23, 0,2038593, L, Red
3751, 20180830T193614.857, 5356.31, 6, 0.1, 203521, m, orange 3752, 20180830T193614.857+0200,
15.75, 1, 0.22, 5646471, S, blue 3753, 2018-09-16T12:30-02:00, 1251, 7, 0, 3036491, l, Black
3754, 2018-09-16T13:15:30Z, 21484.21, 1356, 75, 049681, xs, Red 3755, 9/30/18 19:36:14, -
59.18, 2, 0.333333333333333, 2038593, M, Blue ];
```

Результаты

Таблица Qlik Sense, в которой представлены результаты использования переменной интерпретации TimestampFormat в скрипте загрузки. Последняя метка времени в наборе данных не возвращает верную дату.

transaction_id	transaction_timestamp	LogTimeStamp
3750	2018-08-30	2018-08-30 00:00:00
3751	20180830T193614.857	2018-08-30 19:36:14
3752	20180830T193614.857+0200	2018-08-30 17:36:14
3753	2018-09-16T12:30-02:00	2018-09-16 14:30:00
3754	2018-09-16T13:15:30Z	2018-09-16 13:15:30
3755	9/30/18 19:36:14	-

Следующий скрипт загрузки использует тот же набор данных. Однако он использует *SET TimestampFormat='MM/DD/YYYY hh:mm:ss[.fff]'* для шестой метки времени, формат которой отличен от ISO 8601.

В **Редакторе загрузки данных** замените предыдущий образец скрипта другим скриптом, приведенным ниже, и запустите его. Затем добавьте на лист приложения как минимум поля, указанные в столбце с результатами, чтобы увидеть результаты.

Скрипт загрузки

```
SET FirstWeekDay=0; SET BrokenWeeks=1; SET ReferenceDay=0; SET
DayNames='Mon;Tue;Wed;Thu;Fri;Sat;Sun'; SET
LongDayNames='Monday;Tuesday;Wednesday;Thursday;Friday;Saturday;Sunday'; SET
DateFormat='YYYYMMDD'; SET TimestampFormat='MM/DD/YYYY hh:mm:ss[.fff]';   Transactions: Load
*, Timestamp(transaction_timestamp, 'YYYY-MM-DD hh:mm:ss[.fff]') as LogTimestamp ;   Load *
Inline [ transaction_id, transaction_timestamp, transaction_amount, transaction_quantity,
discount, customer_id, size, color_code 3750, 2018-08-30, 12423.56, 23, 0,2038593, L, Red
3751, 20180830T193614.857, 5356.31, 6, 0.1, 203521, m, orange 3752, 20180830T193614.857+0200,
15.75, 1, 0.22, 5646471, S, blue 3753, 2018-09-16T12:30-02:00, 1251, 7, 0, 3036491, l, Black
3754, 2018-09-16T13:15:30Z, 21484.21, 1356, 75, 049681, xs, Red 3755, 9/30/18 19:36:14, -
59.18, 2, 0.333333333333333, 2038593, M, Blue ];
```

Результаты

Таблица Qlik Sense, в которой представлены результаты использования переменной интерпретации TimestampFormat в скрипте загрузки.

transaction_id	transaction_timestamp	LogTimeStamp
3750	2018-08-30	2018-08-30 00:00:00
3751	20180830T193614.857	2018-08-30 19:36:14
3752	20180830T193614.857+0200	2018-08-30 17:36:14

transaction_id	transaction_timestamp	LogTimeStamp
3753	2018-09-16T12:30-02:00	2018-09-16 14:30:00
3754	2018-09-16T13:15:30Z	2018-09-16 13:15:30
3755	9/30/18 19:36:14	2018-09-16 19:36:14

2.15 Переменные Direct Discovery

Системные переменные Direct Discovery

DirectCacheSeconds

Можно установить предел кэширования для результатов выполнения запросов Direct Discovery для визуализации. При достижении этого предела времени Qlik Sense очищает кэш, когда создаются новые запросы Direct Discovery. Qlik Sense запрашивает исходные данные для выборок и создает кэш снова для указанного временного предела. Кэширование результатов для каждой комбинации выборок выполняется независимо друг от друга. Иначе говоря, кэш обновляется для каждой выборки отдельно таким образом, что одна выборка обновляет кэш только для выбранных полей, а вторая выборка обновляет кэш для соответствующих полей. Если вторая выборка включает в себя поля, обновленные в первой выборке, они не обновляются в кэше повторно, если не достигнут предел кэширования.

Кэш Direct Discovery не применяется к визуализациям **Таблица**. Выборки таблицы каждый раз запрашивают источник данных.

Предельное значение должно быть указано в секундах. Предел кэша по умолчанию составляет 1800 секунд (30 минут).

Значение, используемое для **DirectCacheSeconds**, представляет собой значение, которое устанавливается во время выполнения оператора **DIRECT QUERY**. Это значение невозможно изменить во время выполнения.

Пример:

```
SET DirectCacheSeconds=1800;
```

DirectConnectionMax

Можно выполнять асинхронные параллельные вызовы базы данных с помощью функции объединения подключений. Синтаксис загрузки скрипта для настройки функции объединения:

```
SET DirectConnectionMax=10;
```

Числовой параметр указывает максимальное количество подключений к базе данных, которые можно использовать коду Direct Discovery при обновлении листа. По умолчанию параметр имеет значение 1.

Эту переменную следует использовать с осторожностью. Значение параметра больше 1 может привести к возникновению проблем при подключении к Microsoft SQL Server.

DirectUnicodeStrings
Direct Discovery поддерживает выбор расширенных данных Юникода путем использования стандартного формата SQL для строковых литералов расширенных символов (N'<расширенная строка>'), как это требуют некоторые базы данных (в частности SQL Server). Этот синтаксис можно включить для Direct Discovery с помощью переменной скрипта **DirectUnicodeStrings**.

Если установить для этой переменной значение «true», то перед строковыми символами будет использоваться "N" – строковый маркер ANSI стандартной ширины. Не все базы данных поддерживают этот стандарт. По умолчанию параметр имеет значение «false».

DirectDistinctSupport
Когда значение поля **DIMENSION** выбрано в объекте Qlik Sense, для исходной базы данных создается запрос. Если для запроса требуется группировка, Direct Discovery использует ключевое слово **DISTINCT** для выбора только уникальных значений. Однако для некоторых баз данных необходимо ключевое слово **GROUP BY**. Установите для параметра **DirectDistinctSupport** значение 'false', чтобы создавать запросы **GROUP BY** вместо **DISTINCT** для получения уникальных значений.

```
SET DirectDistinctSupport='false';
```
Если для параметра DirectDistinctSupport установлено значение «true», будет использоваться **DISTINCT**. Если значение не установлено, по умолчанию используется **DISTINCT**.

DirectEnableSubquery
В многотабличных сценариях с большим количеством элементов можно создавать запросы, вложенные в запрос SQL, вместо создания длинного предложения IN. Для этого активируйте параметр **DirectEnableSubquery**, выбрав значение 'true'. По умолчанию установлено значение 'false'.

*Если параметр **DirectEnableSubquery** включен, невозможно загружать таблицы, которые не в режиме Direct Discovery.*

```
SET DirectEnableSubquery='true';
```

Переменные чередования запросов Teradata

Чередование запросов Teradata – это функция, которая позволяет корпоративным приложениям работать совместно с основной базой данных Teradata для повышения эффективности учета, определения приоритетов и управления рабочей нагрузкой. Чередование запросов позволяет переносить метаданные, такие как учетные данные пользователя, в запросе.

Доступны две переменные, которые являются строками. Они оцениваются и отправляются в базу данных.

SQLSessionPrefix
Эта строка отправляется, когда создается подключение к базе данных.

```
SET SQLSessionPrefix = 'SET QUERY_BAND = ' & Chr(39) & 'Who=' & OSuser() & ';' & Chr(39) & '
FOR SESSION;';
```

Например, если **OSuser()** возвращает *WA\sbt*, это будет оценено как `SET QUERY_BAND = 'Who=WA\sbt;' FOR SESSION;`, что и будет отправлено в базу данных при создании подключения.

SQLQueryPrefix
Эта строка отправляется для каждого одиночного запроса.

```
SET SQLSessionPrefix = 'SET QUERY_BAND = ' & Chr(39) & 'Who=' & OSuser() & ';' & Chr(39) & '
FOR TRANSACTION;';
```

Direct DiscoveryСимвольные переменные

DirectFieldColumnDelimiter

Можно задать символ, используемый в качестве разделителя полей в операторах **Direct Query** для баз данных, где в качестве разделителя требуется символ, отличный от запятой. В операторе **SET**указанный символ должен быть заключен в одинарные кавычки.

```
SET DirectFieldColumnDelimiter= '|'
```

DirectStringQuoteChar

Можно задать символ, чтобы заключать строки в кавычки в созданном запросе. По умолчанию это одинарные кавычки. В операторе **SET**указанный символ должен быть заключен в одинарные кавычки.

```
SET DirectStringQuoteChar= '"';
```

DirectIdentifierQuoteStyle

Можно указать, что в созданных запросах можно использовать кавычки не в кодировке ANSI для идентификаторов. В настоящее время доступные кавычки не в кодировке ANSI – только GoogleBQ. По умолчанию используется элемент ANSI. Можно использовать символы в верхнем регистре, в нижнем регистре и в разных регистрах (ANSI, ansi, Ansi).

```
SET DirectIdentifierQuoteStyle="GoogleBQ";
```

Например, кавычки ANSI используются в следующем операторе **SELECT**:

```
SELECT [Quarter] FROM [qvTest].[sales] GROUP BY [Quarter]
```

Если для параметра **DirectIdentifierQuoteStyle** установлено значение "GoogleBQ", оператор **SELECT** будет использовать следующие кавычки:

```
SELECT [Quarter] FROM [qvTest.sales] GROUP BY [Quarter]
```

DirectIdentifierQuoteChar

Можно задать символ, чтобы управлять заключением идентификаторов в кавычки в созданном запросе. Можно задать как один символ (двойные кавычки), так и два символа (квадратные скобки). По умолчанию это двойные кавычки.

```
SET DirectIdentifierQuoteChar='[]';
SET DirectIdentifierQuoteChar='``';
SET DirectIdentifierQuoteChar=' ';
SET DirectIdentifierQuoteChar='""';
```

DirectTableBoxListThreshold
Когда поля Direct Discovery используются в визуализации **Таблица**, устанавливается пороговое значение, чтобы ограничить количество отображаемых строк. Пороговое значение по умолчанию составляет 1000 записей. Пороговое значение по умолчанию можно изменить, настроив переменную **DirectTableBoxListThreshold** в скрипте загрузки. Пример.

```
SET DirectTableBoxListThreshold=5000;
```

Параметр порогового значения применим только к визуализациям **Таблица**, содержащим поля Direct Discovery. Визуализации **Таблица**, содержащие только поля в памяти, не ограничены параметром **DirectTableBoxListThreshold**.

В визуализации **Таблица** не будут отображаться поля, пока количество записей в выборке не будет меньше порогового значения.

Переменные интерпретации числа Direct Discovery

DirectMoneyDecimalSep
Заданный разделитель десятичной части заменяет символ десятичного разделителя для валюты в операторе SQL, созданном для загрузки данных с помощью Direct Discovery. Этот символ должен совпадать с символом, используемом в **DirectMoneyFormat**.

По умолчанию используется значение `'.'`

Пример:

```
Set DirectMoneyDecimalSep='.';
```

DirectMoneyFormat
Заданный разделитель заменяет формат валюты в операторе SQL, созданном для загрузки данных с помощью Direct Discovery. Символ валюты для разделителя тысяч не включается.

По умолчанию используется значение `'#.0000'`

Пример:

```
Set DirectMoneyFormat='#.0000';
```

DirectTimeFormat
Заданный формат времени заменяет формат времени в операторе SQL, созданном для загрузки данных с помощью Direct Discovery.

Пример:

```
Set DirectTimeFormat='hh:mm:ss';
```

DirectDateFormat
Заданный формат даты заменяет формат даты в операторе SQL, созданном для загрузки данных с помощью Direct Discovery.

Пример:

```
Set DirectDateFormat='MM/DD/YYYY';
```

DirectTimeStampFormat
Заданный формат заменяет формат даты и времени в операторе SQL, созданном в операторе SQL для загрузки данных с помощью Direct Discovery.

Пример:

```
Set DirectTimestampFormat='M/D/YY hh:mm:ss[.fff]';
```

2.16 Ошибка переменных

Значения всех ошибок переменных остаются после выполнения скрипта. Первая переменная, ErrorMode, – это входные данные от пользователя, а последние три – выходные данные от программы Qlik Sense с информацией об ошибках в скрипте.

Обзор ошибок переменных

Каждая переменная подробно описывается после обзора. Также можно щелкнуть имя переменной в синтаксисе, чтобы сразу получить подробные сведения об этой конкретной переменной.

Для получения дополнительных сведений о переменных см. интерактивную справку Qlik Sense.

ErrorMode
Эта переменная ошибки определяет действие, которое должно быть предпринято в программе Qlik Sense при обнаружении ошибки в ходе выполнения скрипта.

```
ErrorMode
```

ScriptError
Эта переменная ошибки возвращает код ошибки для последнего выполненного оператора скрипта.

```
ScriptError
```

ScriptErrorCount
Эта переменная ошибки возвращает общее число операторов, которые привели к возникновению ошибки в ходе выполнения текущего скрипта. В начале выполнения скрипта для этой переменной всегда восстанавливается значение 0.

```
ScriptErrorCount
```

ScriptErrorList
Эта переменная ошибки будет содержать объединенный список всех ошибок в скрипте, возникших в ходе выполнения последнего скрипта. Каждая ошибка отделяется символом перевода строки.

```
ScriptErrorList
```

ErrorMode

Эта переменная ошибки определяет действие, которое должно быть предпринято в программе Qlik Sense при обнаружении ошибки в ходе выполнения скрипта.

Синтаксис:

```
ErrorMode
```

Аргументы:

Аргументы

Аргумент	Описание
ErrorMode=1	Настройка по умолчанию. Выполнение скрипта останавливается, и пользователь получает запрос на выполнение действия (в непакетном режиме).
ErrorMode =0	Программа Qlik Sense просто проигнорирует ошибку и продолжит выполнение скрипта со следующего оператора скрипта.
ErrorMode =2	Программа Qlik Sense отобразит сообщение об ошибке «Сбой выполнения скрипта...» при возникновении ошибки без предварительного запроса о действии пользователя.

Пример:

```
set ErrorMode=0;
```

ScriptError

Эта переменная ошибки возвращает код ошибки для последнего выполненного оператора скрипта.

Синтаксис:

```
ScriptError
```

Эта переменная сбрасывается на 0 после каждого успешно выполненного оператора скрипта. При возникновении ошибки переменной присваивается внутренний код ошибки Qlik Sense. Коды ошибок являются двойными значениями, включающими текстовый и числовой компонент. Существуют следующие коды ошибок:

Коды ошибок скрипта

Код ошибки	Описание
0	Нет ошибки. Пустой текст двойного значения.

Код ошибки	Описание
1	Общая ошибка.
2	Ошибка синтаксиса.
3	Общая ошибка ODBC.
4	Общая ошибка OLE DB.
5	Общая ошибка настраиваемой базы данных.
6	Общая ошибка XML.
7	Общая ошибка HTML.
8	Файл не найден.
9	База данных не найдена.
10	Таблица не найдена.
11	Поле не найдено.
12	Неверный формат файла.
16	Семантическая ошибка.

Пример:

```
set ErrorMode=0;

LOAD * from abc.qvf;

if ScriptError=8 then

exit script;

//no file;

end if
```

ScriptErrorCount

Эта переменная ошибки возвращает общее число операторов, которые привели к возникновению ошибки в ходе выполнения текущего скрипта. В начале выполнения скрипта для этой переменной всегда восстанавливается значение 0.

Синтаксис:

```
ScriptErrorCount
```

ScriptErrorList

Эта переменная ошибки будет содержать объединенный список всех ошибок в скрипте, возникших в ходе выполнения последнего скрипта. Каждая ошибка отделяется символом перевода строки.

Синтаксис:

```
ScriptErrorList
```

2 Выражения скрипта

Выражения можно использовать как в операторе **LOAD**, так и **SELECT**. Описываемые в данном разделе синтаксис и функции применяются к оператору **LOAD**, а не к оператору **SELECT**, поскольку последний интерпретируется драйвером ODBC, а не программой Qlik Sense. Тем не менее большинство драйверов ODBC зачастую могут интерпретировать ряд описанных ниже функций.

Выражения состоят из функций, полей и операторов, соединенных по синтаксическим правилам.

Все выражения в скрипте Qlik Sense возвращают число и/или строку, в зависимости от ситуации. Логические функции и операторы возвращают значение 0 для элемента False и -1 для элемента True. Преобразования числа в строку и наоборот являются неявными. Логические операторы и функции интерпретируют значение 0 как False, а все остальные как True.

Ниже представлен общий синтаксис выражения:

Общий синтаксис

Выражение	Поля	Operator
expression ::= (constant	constant	|
expression ::= (constant	fieldref	|
expression ::= (constant	operator1 expression	|
expression ::= (constant	expression operator2 expression	|
expression ::= (constant	function	|
expression ::= (constant	(expression)	)

где:

- элемент **constant** – строка (текст, дата или время), заключенная в одиночные прямые кавычки, или число. Константы записываются без разделителя тысяч, а в качестве десятичного разделителя используется десятичная точка.
- **fieldref** – имя поля загруженной таблицы.
- элемент **operator1** – унарный оператор (работающий над одним выражением, справа).
- элемент **operator2** – бинарный оператор (работающий над двумя выражениями, по одному с каждой стороны).
- **function ::= functionname(parameters)**
- **parameters ::= expression { , expression }**

Число и типы параметров не являются произвольными. Они зависят от используемой функции.

Следовательно, выражения и функции можно свободно вкладывать, и, пока выражение возвращает интерпретируемое значение, программа Qlik Sense не будет выдавать никаких сообщений об ошибках.

3 Выражения диаграммы

Диаграмма (визуализация) – это комбинация функций, полей и математических операторов (+ * / =) и других мер. Выражения используются для обработки данных в приложении, чтобы выдать результат, который можно увидеть в визуализации. Их можно использовать не только с мерами. Можно построить более динамичные и интересные визуализации с выражениями для заголовков, подзаголовков, сносок и даже измерений.

Это значит, например, что вместо заголовка визуализации, который является статичным текстом, можно использовать выражение, результат которого изменяется в зависимости от выборки.

Более подробную информацию о функциях скрипта и диаграммах см. в Синтаксис скрипта и функции диаграммы.

3.1 Определение объема агрегирования

Обычно два фактора в совокупности определяют записи, которые используются для определения значения агрегирования в выражении. При работе в визуализациях эти факторы следующие:

- Значение измерения (в случае агрегирования в выражении диаграммы)
- Выборки

Вместе эти факторы определяют объем агрегирования. Возможны ситуации, когда необходимо проигнорировать в вычислениях выборку и/или измерение. В функциях диаграммы этого можно достичь с помощью классификатора TOTAL, анализа множеств или их комбинации.

Агрегирование: Метод и описание

Способ	Описание
Классификатор TOTAL	Использование классификатора total в функции агрегирования игнорирует значение измерения. Агрегирование будет выполнено в отношении всех возможных значений поля. После классификатора **TOTAL** может быть указан список, включающий одно или несколько имен полей в угловых скобках. Эти имена полей должны быть поднабором переменных измерений диаграммы. В этом случае при вычислении будут проигнорированы все переменные измерений диаграммы, кроме перечисленных, то есть одно значение возвращается для каждого сочетания значений полей в перечисленных полях измерений. Поля, которые в текущий момент не являются измерением в диаграмме, могут также включаться в список. Это может быть полезно для измерений группы, в которых поля измерений не фиксированы. Перечисление всех переменных в группе вызывает выполнение функции при изменении уровня детализации.
Анализ множеств	Использование анализа множеств в агрегировании переопределяет выборку. Агрегирование будет выполнено в отношении всех значений по всем измерениям.
Классификатор TOTAL и анализ множеств	Использование классификатора **TOTAL** и анализа множеств в агрегировании переопределяет выборку и игнорирует измерения.

Классификатор ALL	Использование классификатора **ALL** в агрегировании игнорирует выборку и измерения. Того же можно добиться при помощи оператора анализа множеств {1} и классификатора **TOTAL** :
	
	`=sum(All Sales)`
	
	`=sum({1} Total Sales)`

Пример: Классификатор TOTAL

В следующем примере показано, как классификатор TOTAL можно применить для вычисления доли совместного использования. При условии, что выбран элемент Q2, при использовании классификатора TOTAL рассчитывается сумма всех значений без учета измерений.

Пример: Классификатор Total

Year	Quarter	Sum(Amount)	Sum(TOTAL Amount)	Sum(Amount)/Sum(TOTAL Amount)
		3000	3000	100%
2012	Q2	1700	3000	56,7%
2013	Q2	1300	3000	43,3%

Чтобы показать числа в процентном выражении, выберите на панели свойств для меры, которую необходимо отобразить в процентном выражении, в разделе ***Формат чисел*** *параметр* ***Число****, а в разделе* ***Форматирование*** *выберите параметр* ***Простой*** *и один из форматов %.*

Пример: Анализ множеств

В следующем примере показано, как анализ множеств может быть использован для сравнения наборов данных перед выполнением выборок. При условии, что выбран элемент Q2, при использовании анализа множеств с установленным описанием {1} рассчитывается сумма всех значений без учета выборок, которые не разделены измерениями.

Пример: Анализ множеств

Year	Quarter	Sum(Amount)	Sum({1} Amount)	Sum(Amount)/Sum({1} Amount)
		3000	10800	27,8%
2012	Q1	0	1100	0%
2012	Q3	0	1400	0%
2012	Q4	0	1800	0%
2012	Q2	1700	1700	100%
2013	Q1	0	1000	0%
2013	Q3	0	1100	0%
2013	Q4	0	1400	0%
2013	Q2	1300	1300	100%

Пример: Классификатор TOTAL и анализ множеств

В следующем примере показано, как анализ множеств и классификатор TOTAL можно совместить для сравнения наборов данных перед выполнением выборок и по всем измерениям. При условии, что выбран элемент Q2, при использовании анализа множеств с установленным описанием {1} и префикса TOTAL рассчитывается сумма всех значений без учета выборок и измерений.

Пример: Классификатор TOTAL и анализ множеств

Year	Quarter	Sum (Amount)	Sum({1} TOTAL Amount)	Sum(Amount)/Sum({1} TOTAL Amount)
		3000	10800	27,8%
2012	Q2	1700	10800	15,7%
2013	Q2	1300	10800	12%

Данные, используемые в примерах:

```
AggregationScope: LOAD * inline [ Year Quarter Amount 2012 Q1 1100 2012 Q2 1700 2012 Q3 1400
2012 Q4 1800 2013 Q1 1000 2013 Q2 1300 2013 Q3 1100 2013 Q4 1400] (delimiter is ' ');
```

3.2 Анализ множеств

При создании выборки в приложении необходимо определить подмножество записей в данных. Функции агрегирования, такие как `Sum()`, `Max()`, `Min()`, `Avg()` и `Count()`, вычисляются на основе этого подмножества.

Другими словами, выборка определяет область агрегирования. Она определяет множество записей, на основе которых выполняются вычисления.

Анализ множеств предлагает способ определения области, отличной от множества записей, определяемого текущей выборкой. Новую область также можно рассматривать как альтернативную выборку.

Это может быть полезным, когда требуется сравнить текущую выборку с определенным значением, например с прошлогодним значением или долей глобального рынка.

Выражения множества

Выражения множества используются в функциях агрегирования и заключаются в фигурные скобки. Пример.

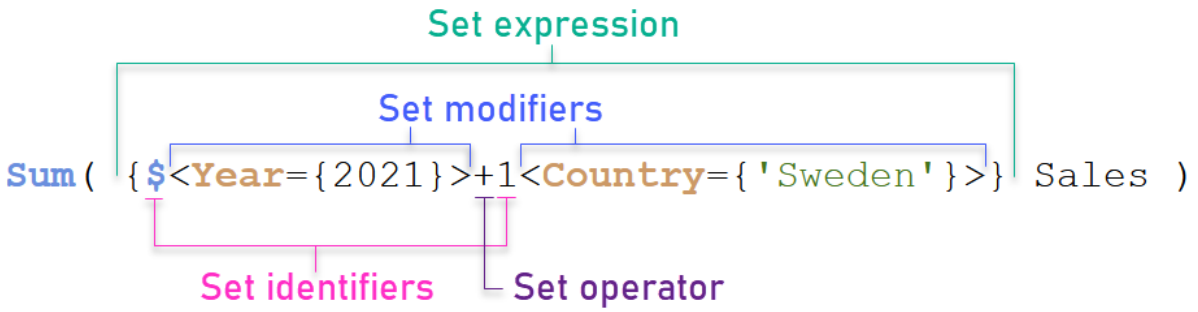
```
Sum( {$<Year={2021}>} Sales )
```

Выражение множества состоит из комбинации следующих элементов.

- **Идентификаторы** Идентификатор множества представляет выборку, определенную в другом месте. Он также представляет конкретное множество записей в данных. Это может быть текущая выборка, выборка из закладки или выборка из альтернативного состояния. Простое выражение множества состоит из одиночного идентификатора, например знака доллара США {$}, что означает все записи в текущей выборке.
  Примеры: `$`, `1`, `BookMark1`, `State2`
- **Операторы** Оператор множества можно использовать для создания объединений, разностей или пересечений между разными идентификаторами множеств. Таким образом, можно создать подмножество или супермножество выборок, определенных идентификаторами множеств.
  Примеры: `+`, `-`, `*`, `/`
- **Модификаторы** Можно добавить модификатор множества в идентификатор множества, чтобы изменить его выборку. Модификатор также можно использовать самостоятельно, в таком случае он будет применяться к идентификатору по умолчанию. Модификатор необходимо заключать в угловые скобки `<…>`.
  Примеры: `<Year={2020}>`, `<Supplier={ACME}>`

Элементы объединяются для формирования выражений множества.

Элементы в выражении множества



К примеру, приведенное выше выражение множества строится на основе агрегирования `Sum (Sales)`.

Первый операнд возвращает продажи за `2021` год для текущей выборки, на которую указывает идентификатор множества `$` и модификатор, содержащий выборку `2021` года. Второй операнд возвращает значение `Sales` для `Sweden` и игнорирует текущую выборку, на которую указывает идентификатор множества `1`.

Наконец, выражение возвращает множество, состоящее из записей, принадлежащих любому из двух операндов множества, на что указывает оператор множества `+`.

Примеры

Примеры, в которых объединены описанные выше элементы выражения множества, приводятся в следующих разделах:

Натуральные множества

Обычно выражение множества представляет множество записей в модели данных и выборку, определяющую этот набор данных. В таком случае множество называется натуральным.

Идентификаторы множеств, с модификаторами множеств или без, всегда представляют натуральные множества.

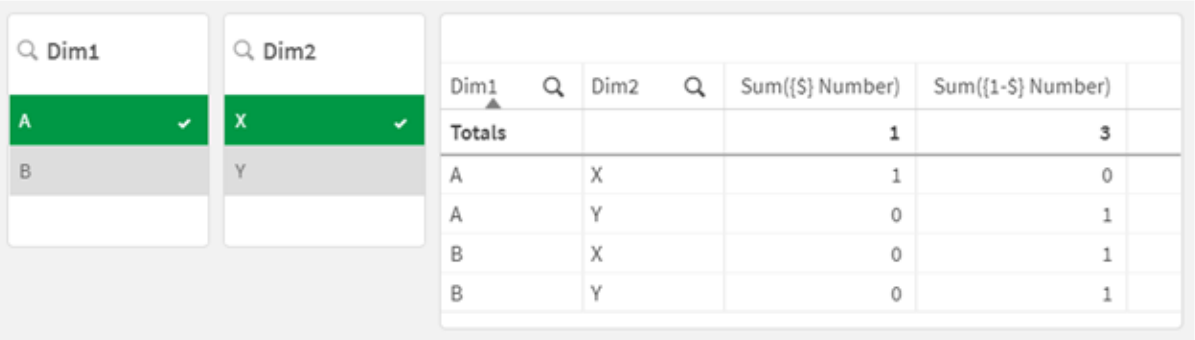
Однако выражение множества, использующее операторы множества, также представляет подмножество записей, но, как правило, уже не может быть описано как использующее выборку значений поля. Такое выражение является ненатуральным множеством.

Например, множество, заданное в `{1-$}`, не может всегда определяться как выборка. Поэтому оно не является натуральным множеством. Чтобы продемонстрировать это, можно загрузить следующие данные, добавить их в таблицу, а затем создать выборки с помощью фильтров.

```
Load * Inline [Dim1, Dim2, Number A, X, 1 A, Y, 1 B, X, 1 B, Y, 1];
```

Создавая выборки для `Dim1` и `Dim2`, получается вид, показанный в следующей таблице.

Таблица с натуральными и ненатуральными множествами



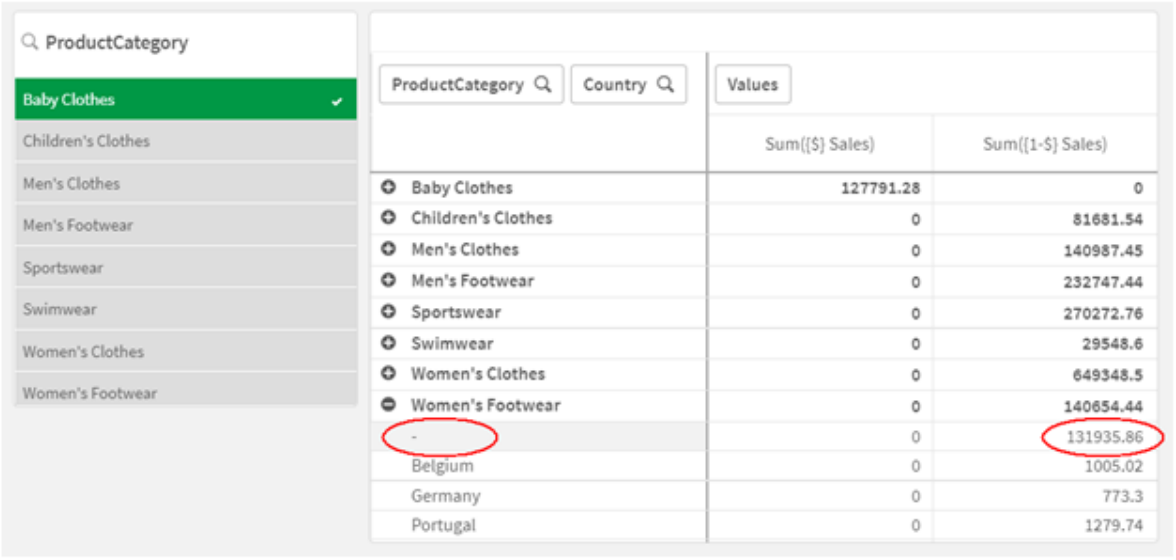
Dim1	Dim2	Sum({$} Number)	Sum({1-$} Number)
Totals		1	3
A	X	1	0
A	Y	0	1
B	X	0	1
B	Y	0	1

Выражение множества в первой мере использует натуральное множество: оно соответствует выборке, созданной в `{$}`.

Вторая мера отличается. В ней используется `{1-$}`. Невозможно создать выборку, соответствующую этому множеству, так как оно является ненатуральным.

Такое различие имеет ряд последствий:

- Модификаторы множеств можно применять только к идентификаторам множеств. Их нельзя применить к произвольному выражению множества. Например, невозможно использовать выражение множества, так как:
  `{ (BM01 * BM02) <Field={x,y}> }`
  В данном случае обычные (круглые) скобки обозначают, что пересечение между `BM01` и `BM02` должно вычисляться до применения модификатора множества. Причина заключается в том, что отсутствует множество элементов, доступное для модификации.
- Нельзя использовать ненатуральные множества в функциях элементов `P()` и `E()`. Эти функции возвращают множество элементов, но при этом невозможно вывести множество элементов из ненатурального множества.
- Мера, использующая ненатуральное множество, не всегда приписывается правильному значению измерения, если модель данных включает много таблиц. Например, в следующей диаграмме несколько исключенных значений продаж приписаны правильным Country, а для других NULL указано как Country.
  Диаграмма с ненатуральным множеством



Правильность назначения зависит от модели данных. В данном случае номер невозможно назначить, если он относится к стране, исключенной из выборки.

Идентификаторы множества

Идентификатор представляет множество записей в данных: все данные или подмножество данных. Это множество записей, определенное путем выборки. Это может быть текущая выборка, все данные (без выборки), выборка из закладки или выборка из альтернативного состояния.

В примере `Sum( {$<Year = {2009}>} Sales )` идентификатором является знак доллара: `$`. Это представляет текущую выборку. Он также представляет все возможные записи. Этот набор в дальнейшем может изменяться частью модификатора выражения множества: добавляется выборка `2009` в `Year`.

В более сложном выражении множества можно использовать два идентификатора вместе с оператором, образующих объединение, разность или пересечение двух множеств записей.

В следующей таблице показано несколько обычных идентификаторов.

Примеры часто используемых идентификаторов

Идентификатор	Описание
`1`	Представляет полное множество всех записей в приложении, независимо от выборок.
`$`	Представляет записи текущей выборки в состоянии по умолчанию. Выражение множества `{$}`, таким образом, эквивалентно неутверждению выражения множества.

Идентификатор	Описание
$1	Представляет предыдущую выборку в состоянии по умолчанию. $2 представляет предпоследнюю выборку и т. д.
$_1	Представляет следующую выборку (+1). $_2 представляет выборку, через одну (+2) и т. д.
BM01	Можно использовать любой ID закладки или имя закладки.
AltState	Можно ссылаться на альтернативное состояние, указав имя этого состояния.
AltState::BM01	Закладка содержит выборки всех состояний, можно ссылаться на конкретную закладку, указав ее имя.

В следующей таблице показано несколько примеров различных идентификаторов.

Примеры различных идентификаторов

Пример	Результат
Sum ({1} Sales)	Возвращает общий объем продаж в приложении, игнорируя выборки, но не измерение.
Sum ({$} Sales)	Возвращает продажи для текущей выборки, т. е. это то же самое, что элемент Sum(Sales).
Sum ({$1} Sales)	Возвращает продажи для предыдущей выборки.
Sum ({BM01} Sales)	Возвращает продажи для закладки с именем BM01.

Операторы множеств

Операторы множества используются для включения, исключения или пересечения множеств данных. Все операторы используют множества в качестве операндов и в результате возвращают множество.

Можно использовать операторы множеств в двух различных ситуациях:

- Выполнение операции множества с идентификаторами множества, представляет множества записей в данных.
- Выполнение операции множества с множествами элементов, значениями полей или внутри модификатора множества.

В следующей таблице показаны операторы, которые можно использовать в выражениях множества.

Операторы

Оператор	Описание
+	Объединение. Данная бинарная операция возвращает множество, состоящее из записей или элементов, принадлежащих любому из двух операндов множества.
-	Исключение. Данная бинарная операция возвращает множество записей или элементов, принадлежащих первому из двух операндов множества. Также, при использовании в качестве унарного оператора, она возвращает дополнительное множество.
*	Пересечение. Данная бинарная операция возвращает множество, состоящее из записей или элементов, принадлежащих обоим операндам множества.
/	Симметрическая разность (XOR). Данная бинарная операция возвращает множество, состоящее из записей или элементов, принадлежащих одному из операндов множества, но не сразу обоим.

В следующей таблице показано несколько примеров с операторами.

Примеры с операторами

Пример	Результат
`Sum ({1-$} Sales)`	Возвращает продажи для всего, исключенного текущей выборкой.
`Sum ({$*BM01} Sales)`	Возвращает продажи для пересечения между выборкой и закладкой BM01.
`Sum ({-($+BM01)} Sales)`	Возвращает продажи, исключенные выборкой и закладкой BM01
`Sum ({$<Year={2009}>+1<Country={'Sweden'}>} Sales)`	Возвращает продажи за 2009 год, связанные с текущими выборками, и добавляет полное множество данных, связанных со страной Sweden за все годы.
`Sum ({$<Country={"S*"}+{"*land"}>} Sales)`	Возвращает объемы продаж для стран, которые начинаются с «S» или заканчиваются на «land».

Модификаторы множества

Выражения множества используются для определения области вычисления. Центральной частью выражения множества является модификатор множества, который указывает выборку. Он используется для модификации пользовательской выборки или выборки в идентификаторе множества, а результат определяет новую область вычислений.

Модификатор множества состоит из одного или нескольких имен полей, для каждого имени перечислены значения, применяемые к полю. Модификатор заключен в угловые скобки: < >

Пример.

- `Sum ( {$<Year = {2015}>} Sales )`
- `Count ( {1<Country = {Germany}>} distinct OrderID )`
- `Sum ( {$<Year = {2015}, Country = {Germany}>} Sales )`

Множества элементов

Элемент можно определить используя следующие средства:

- Список значений
- Поиск
- Ссылка на другое поле
- Функция множества

Если определение множества элементов пропущено, модификатор множества сбросит любую выборку в этом поле. Пример.

```
Sum( {$<Year = >} Sales )
```

Примеры: Выражения диаграммы для модификаторов множества на основе множеств элементов

Примеры. Выражения диаграммы

Скрипт загрузки

Загрузите следующие данные через встроенную загрузку в редакторе загрузки данных, чтобы создать примеры с выражениями диаграммы, показанные ниже.

```
MyTable: Load * Inline [ Country, Year, Sales Argentina, 2014, 66295.03 Argentina, 2015, 140037.89 Austria, 2014, 54166.09 Austria, 2015, 182739.87 Belgium, 2014, 182766.87 Belgium, 2015, 178042.33 Brazil, 2014, 174492.67 Brazil, 2015, 2104.22 Canada, 2014, 101801.33 Canada, 2015, 40288.25 Denmark, 2014, 45273.25 Denmark, 2015, 106938.41 Finland, 2014, 107565.55 Finland, 2015, 30583.44 France, 2014, 115644.26 France, 2015, 30696.98 Germany, 2014, 8775.18 Germany, 2015, 77185.68 ];
```

Выражения диаграммы

Создайте таблицу на листе Qlik Sense со следующими выражениями диаграммы.

Таблица. Модификаторы множества на основе множеств элементов

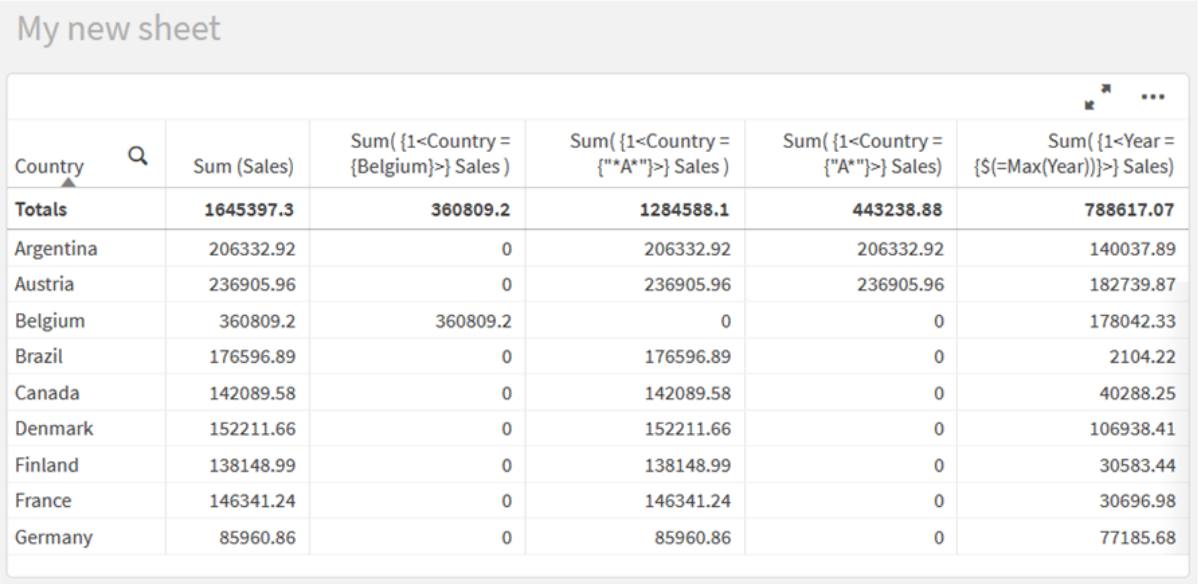
Country	Sum(Sales)	Sum ({1<Country= {Belgium}>} Sales)	Sum ({1<Country= {"*A*"}>} Sales)	Sum ({1<Country= {"A*"}>} Sales)	Sum ({1<Year= {$(=Max (Year))}>} Sales)
Итоги	1645397.3	360809.2	1284588.1	443238.88	788617.07
Argentina	206332.92	0	206332.92	206332.92	140037.89

Country	Sum(Sales)	Sum ({1<Country={Belgium}>} Sales)	Sum ({1<Country={"*A*"}>} Sales)	Sum ({1<Country={"A*"}>} Sales)	Sum ({1<Year={$(=Max(Year))}>} Sales)
Austria	236905.96	0	236905.96	236905.96	182739.87
Belgium	360809.2	360809.2	0	0	178042.33
Brazil	176596.89	0	176596.89	0	2104.22
Canada	142089.58	0	142089.58	0	40288.25
Denmark	152211.66	0	152211.66	0	106938.41
Finland	138148.99	0	138148.99	0	30583.44
France	146341.24	0	146341.24	0	30696.98
Germany	85960.86	0	85960.86	0	77185.68

Объяснение

- Измерения:
  - `Country`
- Меры:
  - `Sum(Sales)`
    Суммировать `Sales` без выражения множества.
  - `Sum({1<Country={Belgium}>}Sales)`
    Выбрать `Belgium`, а затем суммировать соответствующие `Sales`.
  - `Sum({1<Country={"*A*"}>}Sales)`
    Выбрать все страны, у которых есть `A`, затем суммировать соответствующие `Sales`.
  - `Sum({1<Country={"A*"}>}Sales)`
    Выбрать все страны, название которых начинается с `A`, а затем суммировать соответствующие `Sales`.
  - `Sum({1<Year={$(=Max(Year))}>}Sales)`
    Вычислить `Max(Year)`, который равен `2015`, а затем суммировать соответствующие `Sales`.

Модификаторы множества на основе множеств элементов


My new sheet

Country	Sum (Sales)	Sum({1<Country = {Belgium}>} Sales)	Sum({1<Country = {"*A*"}>} Sales)	Sum({1<Country = {"A*"}>} Sales)	Sum({1<Year = {$(=Max(Year))}>} Sales)
Totals	**1645397.3**	**360809.2**	**1284588.1**	**443238.88**	**788617.07**
Argentina	206332.92	0	206332.92	206332.92	140037.89
Austria	236905.96	0	236905.96	236905.96	182739.87
Belgium	360809.2	360809.2	0	0	178042.33
Brazil	176596.89	0	176596.89	0	2104.22
Canada	142089.58	0	142089.58	0	40288.25
Denmark	152211.66	0	152211.66	0	106938.41
Finland	138148.99	0	138148.99	0	30583.44
France	146341.24	0	146341.24	0	30696.98
Germany	85960.86	0	85960.86	0	77185.68

Перечисленные значения

Наиболее распространенный пример множества элементов – это выражение, основанное на списке значений поля, заключенном в фигурные скобки. Пример.

- `{$<Country = {Canada, Germany, Singapore}>}`
- `{$<Year = {2015, 2016}>}`

Фигурные скобки внутри определяют множество элементов. Отдельные значения разделяются запятыми.

Кавычки и чувствительность к регистру

Если значения содержат пробелы или специальные символы, они должны быть заключены в кавычки. При использовании одинарных кавычек возвращаются результаты с точным соответствием, с учетом регистра и одним значением поля. При использовании двойных кавычек возвращаются результаты без учета регистра с одним или несколькими значениями поля. Пример.

- `<Country = {'New Zealand'}>`
  Соответствует только `New Zealand`.
- `<Country = {"New Zealand"}>`
  Соответствует `New Zealand`, `NEW ZEALAND` и `new zealand`.

Даты должны быть заключены в кавычки, должен использоваться формат даты в рассматриваемом поле. Пример.

- `<ISO_Date = {'2021-12-31'}>`
- `<US_Date = {'12/31/2021'}>`
- `<UK_Date = {'31/12/2021'}>`

Двойные кавычки можно заменить квадратными скобками или апострофами.

Поиски

Множества элементов также можно создавать через поиски. Пример.

- `<Country = {"C*"}>`
- `<Ingredient = {"*garlic*"}>`
- `<Year = {">2015"}>`
- `<Date = {">12/31/2015"}>`

В текстовых поисках можно использовать подстановочные знаки: Звездочка (`*`) представляет любое количество символов, а знак вопроса (`?`) — один символ. Реляционные операторы могут использоваться для определения числовых поисков.

Для поисков следует всегда использовать двойные кавычки. При поиске регистр не учитывается.

Расширения со знаком доллара

Расширения со знаком доллара необходимы, если нужно использовать вычисления внутри набора элементов. Например, если нужно выполнить поиск только за последний возможный год, используйте:

```
<Year = {$(=Max(Year))}>
```

Выбранные значения в других полях

Модификаторы можно построить на основе выбранных значений другого поля. Пример.

```
<OrderDate = DeliveryDate>
```

Данный модификатор возьмет выбранные значения из элемента `DeliveryDate` и применит их в качестве выборки к элементу `OrderDate`. Если присутствует множество уникальных значений (больше пары сотен), то данная операция потребует большой загрузки ЦП, поэтому ее следует избегать.

Функции множества элементов

Множество элементов может быть основано на функциях множества `P()` (возможные значения) и `E()` (исключенные значения).

Например, если требуется выбрать страну, где продается продукт «`Cap`», можно использовать:

```
<Country = P({1<Product={Cap}>} Country)>
```

Аналогично, если требуется выбрать страны, где не продается продукт «`Cap`», можно использовать:

```
<Country = E({1<Product={Cap}>} Country)>
```

Модификаторы множества с поиском

Можно создавать множества элементов посредством поисковых запросов с модификаторами множества.

Пример.

- `<Country = {"C*"}>`
- `<Year = {">2015"}>`
- `<Ingredient = {"*garlic*"}>`

Поисковые запросы следует всегда заключать в двойные кавычки, квадратные скобки или апострофы. Можно использовать список, содержащий буквальные строки (в одинарных кавычках) и поиски (в двойных кавычках). Пример.

```
<Product = {'Nut', "*Bolt", Washer}>
```

Текстовые поиски

В текстовых поисках можно использовать подстановочные знаки и другие символы:

- Звездочка (*) будет представлять любое количество символов.
- Знак вопроса (?) будет представлять один символ.
- Циркумфлекс (^) будет обозначать начало слова.

Пример.

- `<Country = {"C*", "*land"}>`
  Найти все страны, название которых начинается с `C` или оканчивается на `land`.
- `<Country = {"*^z*"}>`
  Будут возвращены все страны, название которых начинается с `z`, например `New Zealand`.

Числовые поиски

Для числовых поисков можно использовать следующие реляционные операторы: `>`, `>=`, `<`, `<=`

Числовой поиск всегда начинается с одного из этих операторов. Пример.

- `<Year = {">2015"}>`
  Найти данные за 2016 и все последующие годы.
- `<Date = {">=1/1/2015<1/1/2016"}>`
  Найти данные для всех дат в 2015 году. Обратите внимание на синтаксис для обозначения диапазона времени между двумя датами. Формат даты должен соответствовать формату даты рассматриваемого поля.

Поиски выражений

Можно использовать поиски выражений для выполнения расширенного поиска. В таком случае агрегирование оценивается для каждого значения поля в поле поиска. Выбираются все значения, для которых выражение поиска возвращает значение true.

Поиск выражения всегда начинается со знака равенства. `=`

Пример.

```
<Customer = {"=Sum(Sales)>1000"}>
```

По этому запросу будут возвращены все клиенты со значением объема продаж больше 1000. `Sum (Sales)` рассчитывается на основании текущей выборки. Это означает, что если имеется выборка в другом поле, например в поле `Product`, будут возвращены клиенты, которые удовлетворяют условию объема продаж только для выбранных продуктов.

Если требуется использовать условие независимо от выборки, необходимо использовать анализ множеств внутри строки поиска. Пример.

```
<Customer = {"=Sum({1} Sales)>1000"}>
```

Выражения после знака равенства будут интерпретироваться как булево значение. Другими словами, если выражение дает другой результат, то любое число, отличное от нуля, интерпретируется как true, а 0 и строковые значения – как false.

Кавычки

Используйте кавычки, когда строки поиска содержат пробелы или специальные символы. При использовании одинарных кавычек возвращаются результаты с точным соответствием, с учетом регистра и одним значением поля. При использовании двойных кавычек поиск выполняется без учета регистра, а результаты могут соответствовать нескольким значениям поля.

Пример.

- `<Country = {'New Zealand'}>`
  Соответствует только `New Zealand`.
- `<Country = {"New Zealand"}>`
  Соответствует `New Zealand`, `NEW ZEALAND` и `new zealand`.

Двойные кавычки можно заменить квадратными скобками или апострофами.

> *В предыдущих версиях Qlik Sense не было различий в использовании одинарных и двойных кавычек. Все строки, заключенные в кавычки, обрабатывались как поиски. В целях обеспечения обратной совместимости приложения, созданные при помощи более ранних версий Qlik Sense, будут работать в том же порядке, что и в предыдущих версиях. В приложениях, созданных при помощи Qlik Sense November 2017 и более поздних версий, учитывается различие между двумя типами кавычек.*

Примеры: Выражения диаграммы для модификаторов множества с поисками

Примеры. Выражения диаграммы

Скрипт загрузки

Загрузите следующие данные через встроенную загрузку в редакторе загрузки данных, чтобы создать примеры с выражениями диаграммы, показанные ниже.

```
MyTable: Load  Year(Date) as Year, Date#(Date,'YYYY-MM-DD') as ISO_Date,  Date(Date#
(Date,'YYYY-MM-DD'),'M/D/YYYY') as US_Date,  Country, Product, Amount  Inline [Date, Country,
Product, Amount 2018-02-20, Canada, Washer, 6 2018-07-08, Germany, Anchor bolt, 10 2018-07-14,
```

```
Germany, Anchor bolt, 3 2018-08-31, France, Nut, 2 2018-09-02, Czech Republic, Bolt, 1 2019-
02-11, Czech Republic, Bolt, 3 2019-07-31, Czech Republic, Washer, 6 2020-03-13, France,
Anchor bolt, 1 2020-07-12, Canada, Anchor bolt, 8 2020-09-16, France, Washer, 1];
```

Пример 1. Выражения диаграммы с текстовыми поисками

Создайте таблицу на листе Qlik Sense со следующими выражениями диаграммы.

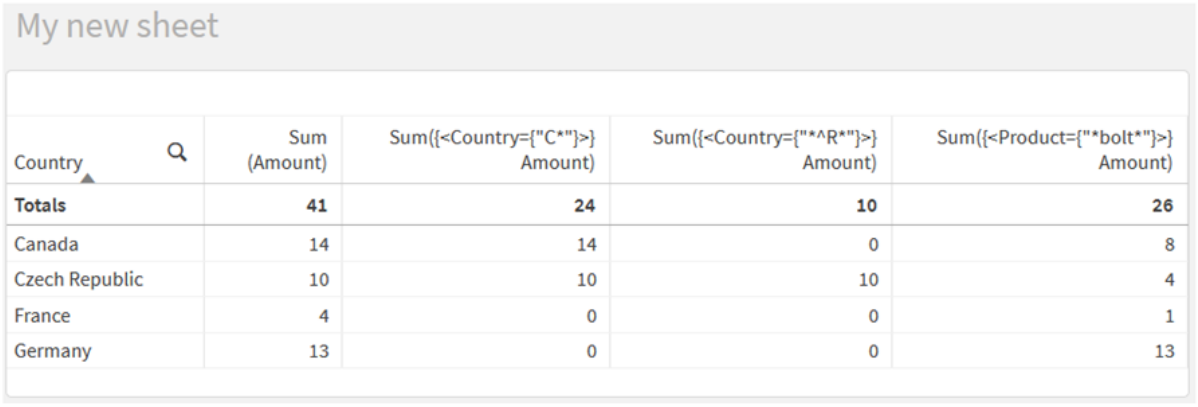
Таблица. Модификаторы множества с текстовыми поисками

Country	Sum (Amount)	Sum({<Country={"C*"}>} Amount)	Sum({<Country={"*^R*"}>} Amount)	Sum({<Product={"*bolt*"}>} Amount)
Итоги	**41**	**24**	**10**	**26**
Canada	14	14	0	8
Czech Republic	10	10	10	4
France	4	0	0	1
Germany	13	0	0	13

Объяснение

- Измерения:
  - `Country`
- Меры:
  - `Sum(Amount)`
    Суммировать `Amount` без выражения множества.
  - `Sum({<Country={"C*"}>}Amount)`
    Суммировать `Amount` для всех стран, название которых начинается с `C`, например `Canada` и `Czech Republic`.
  - `Sum({<Country={"*^R*"}>}Amount)`
    Суммировать `Amount` для всех стран, включающих слово, которое начинается с `R`, например `Czech Republic`.
  - `Sum({<Product={"*bolt*"}>}Amount)`
    Суммировать `Amount` для всех продуктов, содержащих строку `bolt`, например `Bolt` и `Anchor bolt`.

Модификаторы множества с текстовыми поисками



Country	Sum (Amount)	Sum({<Country={"C*"}>} Amount)	Sum({<Country={"*^R*"}>} Amount)	Sum({<Product={"*bolt*"}>} Amount)
Totals	**41**	**24**	**10**	**26**
Canada	14	14	0	8
Czech Republic	10	10	10	4
France	4	0	0	1
Germany	13	0	0	13

Пример 2. Выражения диаграммы с числовыми поисками

Создайте таблицу на листе Qlik Sense со следующими выражениями диаграммы.

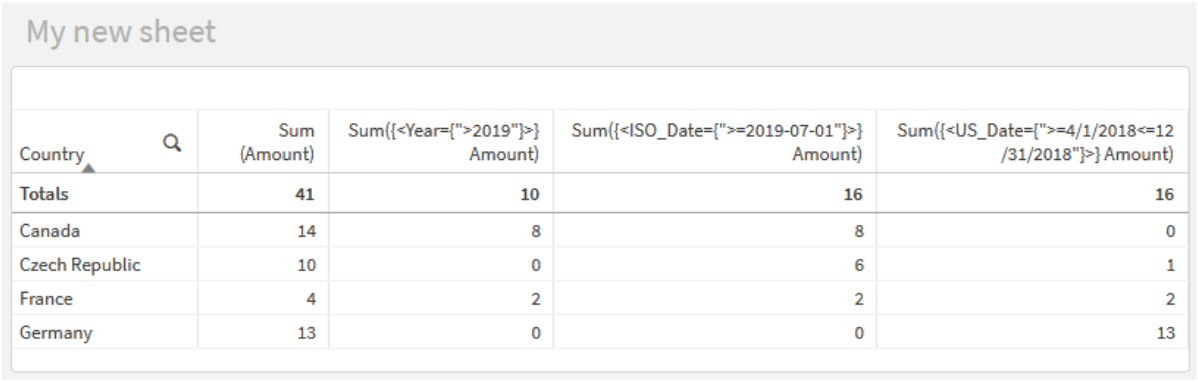
Таблица. Модификаторы множества с числовыми поисками

Country	Sum (Amount)	Sum({<Year={">2019"}>} Amount)	Sum({<ISO_Date={">=2019-07-01"}>} Amount)	Sum({<US_Date={">=4/1/2018<=12/31/2018"}>} Amount)
Итоги	**41**	**10**	**16**	**16**
Canada	14	8	8	0
Czech Republic	10	0	6	1
France	4	2	2	2
Germany	13	0	0	13

Объяснение

- Измерения:
  - `Country`
- Меры:
  - `Sum(Amount)`
    Суммировать `Amount` без выражения множества.
  - `Sum({<Year={">2019"}>}Amount)`
    Суммировать `Amount` за все годы после `2019`.
  - `Sum({<ISO_Date={">=2019-07-01"}>}Amount)`
    Суммировать `Amount` для `2019-07-01` и последующих дат. Формат даты в поиске должен соответствовать формату поля.
  - `Sum({<US_Date={">=4/1/2018<=12/31/2018"}>}Amount)`
    Суммировать `Amount` для всех дат с `4/1/2018` по `12/31/2018` включительно. Формат дат в поиске должен соответствовать формату поля.

Модификаторы множества с числовыми поисками



Country	Sum (Amount)	Sum({<Year={">2019"}>} Amount)	Sum({<ISO_Date={">=2019-07-01"}>} Amount)	Sum({<US_Date={">=4/1/2018<=12/31/2018"}>} Amount)
Totals	**41**	**10**	**16**	**16**
Canada	14	8	8	0
Czech Republic	10	0	6	1
France	4	2	2	2
Germany	13	0	0	13

Пример 3: Выражения диаграммы с поисками выражения

Создайте таблицу на листе Qlik Sense со следующими выражениями диаграммы.

Table - Set modifiers with expression searches

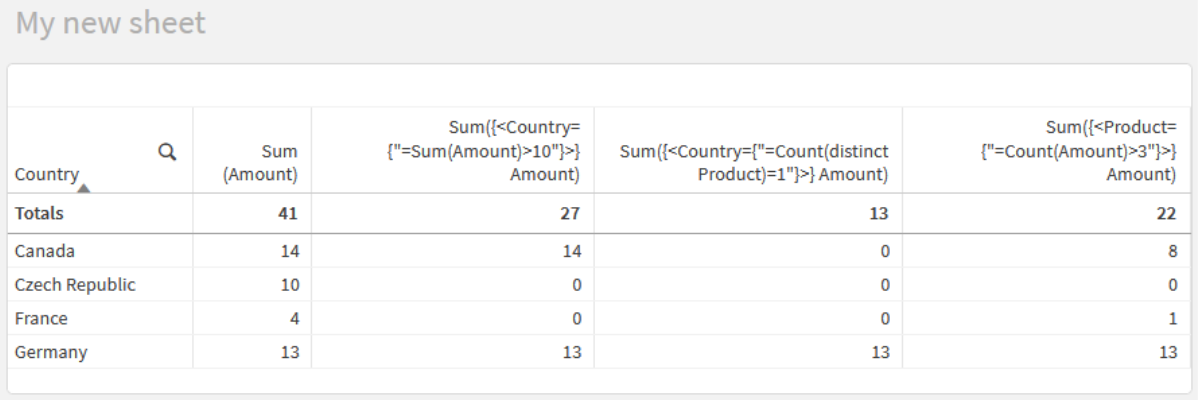
Country	Sum (Amount)	Sum({<Country={"=Sum (Amount)>10"}>} Amount)	Sum({<Country={"=Count(distinct Product)=1"}>} Amount)	Sum({<Product={"=Count (Amount)>3"}>} Amount)
Totals	**41**	**27**	**13**	**22**
Canada	14	14	0	8
Czech Republic	10	0	0	0
France	4	0	0	1
Germany	13	13	13	13

Объяснение

- Измерения:
  - `Country`
- Меры:
  - `Sum(Amount)`
    Суммировать `Amount` без выражения множества.
  - `Sum({<Country={"=Sum(Amount)>10"}>}Amount)`
    Суммировать `Amount` для всех стран, у которых агрегированная сумма `Amount` больше 10.
  - `Sum({<Country={"=Count(distinct Product)=1"}>}Amount)`
    Суммировать `Amount` для всех стран, связанных только с одним уникально идентифицируемым продуктом.
  - `Sum({<Product={"=Count(Amount)>3"}>}Amount)`

Суммировать `Amount` для всех стран, для которых имеется больше трех транзакций в базе данных.

Модификаторы множества с поисками выражения


My new sheet

Country	Sum (Amount)	Sum({<Country={"=Sum(Amount)>10"}>} Amount)	Sum({<Country={"=Count(distinct Product)=1"}>} Amount)	Sum({<Product={"=Count(Amount)>3"}>} Amount)
Totals	**41**	**27**	**13**	**22**
Canada	14	14	0	8
Czech Republic	10	0	0	0
France	4	0	0	1
Germany	13	13	13	13

Модификаторы множества с расширениями со знаком доллара

Расширения со знаком доллара – это конструкции, которые рассчитываются перед анализом и вычислением выражения. Затем результат подставляется в выражение вместо `$(…)`. Затем вычисляется выражение с использованием результата расширения со знаком доллара.

В редакторе выражения отображается предварительный просмотр расширения со знаком доллара, которое позволяет проверить, какой результат дает его вычисление.

Предварительный просмотр расширения со знаком доллара в редакторе выражения

Используйте расширения со знаком доллара, когда требуется использовать вычисления внутри множества элементов.

Например, если требуется найти только самый последний год, можно использовать следующую конструкцию:

```
<Year = {$(=Max(Year))}>
```

Сначала рассчитывается `Max(Year)`, а полученный результат подставляется в выражение вместо `$(…)`.

Результат после расширения со знаком доллара будет включать выражение, например:

```
<Year = {2021}>
```

Выражение внутри расширения со знаком доллара вычисляется на основе текущей выборки. Это означает, что если имеются выбранные элементы в другом поле, это повлияет на результат вычисления выражения.

Если требуется вычислить выражение независимо от выборки, необходимо использовать анализ множеств внутри расширения со знаком доллара. Пример.

```
<Year = {$(=Max({1} Year))}>
```

Строки

Когда расширение со знаком доллара должно возвращать строку, применяются обычные правила использования кавычек. Пример.

```
<Country = {'$(=FirstSortedValue(Country,Date)'}>
```

Результат после расширения со знаком доллара будет включать выражение, например:

```
<Country = {'New Zealand'}>
```

Если не используются кавычки, возвращается ошибка синтаксиса.

Числа

Когда расширение со знаком доллара должно возвращать число, оно должно иметь тот же формат, что и в поле. Это означает, что иногда необходимо заключить выражение в функцию форматирования.

Пример.

```
<Amount = {$(=Num(Max(Amount), '###0.00'))}>
```

Результат после расширения со знаком доллара будет включать выражение, например:

```
<Amount = {12362.00}>
```

Используйте хэш, чтобы в расширении всегда использовалась десятичная запятая и чтобы не использовался разделитель разряда тысяч. Пример.

```
<Amount = {$(#=Max(Amount))}>
```

Даты

Когда расширение со знаком доллара должно возвращать дату, убедитесь, что в нем используется правильное форматирование. Это означает, что иногда необходимо заключить выражение в функцию форматирования.

Пример.

```
<Date = {'$(=Date(Max(Date)))'}>
```

Результат после расширения со знаком доллара будет включать выражение, например:

```
<Date = {'12/31/2015'}>
```

Как и со строками, необходимо правильно использовать кавычки.

Чаще всего рекомендуется ограничивать расчеты последним месяцем (или годом). Затем можно использовать числовой поиск вместе с функцией `AddMonths()`.

Пример.

```
<Date = {">=$(=AddMonths(Today(),-1))"}>
```

Результат после расширения со знаком доллара будет включать выражение, например:

```
<Date = {">=9/31/2021"}>
```

При этом будут возвращены все события за последний месяц.

Пример: Выражения диаграммы для модификаторов множества с расширениями со знаком доллара

Пример: выражения диаграммы

Скрипт загрузки

Загрузите следующие данные через встроенную загрузку в редакторе загрузки данных, чтобы создать примеры с выражениями диаграммы, показанные ниже.

```
Let vToday = Today(); MyTable: Load  Year(Date) as Year, Date#(Date,'YYYY-MM-DD') as ISO_Date,
 Date(Date#(Date,'YYYY-MM-DD'),'M/D/YYYY') as US_Date,  Country, Product, Amount  Inline
[Date, Country, Product, Amount 2018-02-20, Canada, Washer, 6 2018-07-08, Germany, Anchor
bolt, 10 2018-07-14, Germany, Anchor bolt, 3 2018-08-31, France, Nut, 2 2018-09-02, Czech
Republic, Bolt, 1 2019-02-11, Czech Republic, Bolt, 3 2019-07-31, Czech Republic, Washer, 6
2020-03-13, France, Anchor bolt, 1 2020-07-12, Canada, Anchor bolt, 8 2021-10-15, France,
Washer, 1];
```

Выражения диаграммы с расширениями со знаком доллара

Создайте таблицу на листе Qlik Sense со следующими выражениями диаграммы.

Таблица. Модификаторы множества с расширениями со знаком доллара

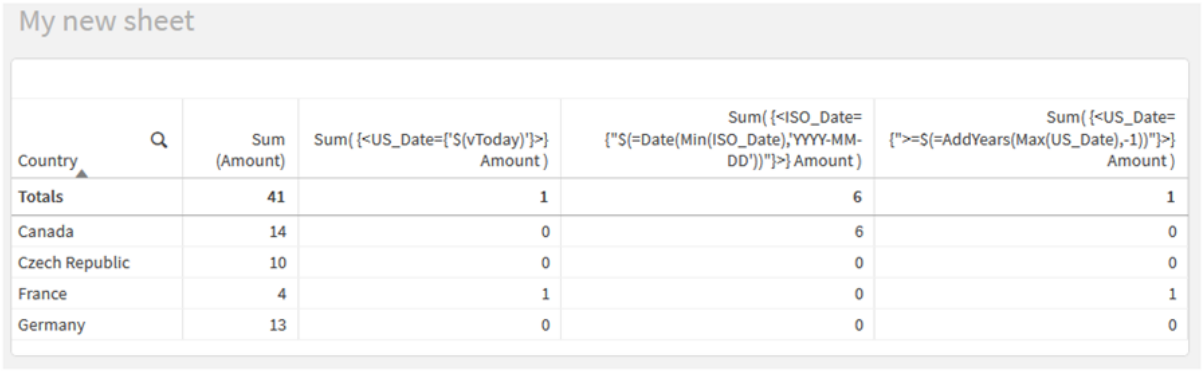
Country	Sum (Amount)	Sum({<US_ Date= {'$(vToday)'}>} Amount)	Sum({<ISO_Date= {"$(=Date(Min(ISO_ Date),'YYYY-MM-DD'))"}>} Amount)	Sum({<US_Date= {">=$(=AddYears(Max (US_Date),-1))"}>} Amount)
Итоги	**41**	**1**	**6**	**1**
Canada	14	0	6	0
Czech Republic	10	0	0	0

Country	Sum (Amount)	Sum({<US_ Date= {'$(vToday)'}>} Amount)	Sum({<ISO_Date= {"$(=Date(Min(ISO_ Date),'YYYY-MM-DD'))"}>} Amount)	Sum({<US_Date= {">=$(=AddYears(Max (US_Date),-1))"}>} Amount)
Итоги	**41**	**1**	**6**	**1**
France	4	1	0	1
Germany	13	0	0	0

Объяснение

- Измерения:
  - `Country`
- Меры:
  - `Sum(Amount)`
    Суммировать `Amount` без выражения множества.
  - `Sum({<US_Date={'$(vToday)'}>}Amount)`
    Суммировать `Amount` для всех записей, в которых `US_Date` совпадает с переменной `vToday`.
  - `Sum({<ISO_Date={"$(=Date(Min(ISO_Date),'YYYY-MM-DD'))"}>}Amount)`
    Суммировать `Amount` для всех записей, в которых `ISO_Date` совпадает с первым (наименьшим) возможным элементом `ISO_Date`. Функция `Date()` необходима для того, чтобы формат даты соответствовал полю.
  - `Sum({<US_Date={">=$(=AddYears(Max(US_Date),-1))"}>}Amount)`
    Суммировать `Amount` для всех записей, содержащих `US_Date` эту же дату или последующие даты за год до самого последнего (самого крупного) `US_Date`. Функция `AddYears()` будет возвращать дату в формате, заданным переменной`DateFormat`, и этот формат должен соответствовать содержимому поля `US_Date`.

Модификаторы множества с расширениями со знаком доллара



Модификаторы множества с операторами множества

Операторы множества используются для включения, исключения или пересечения различных множеств элементов. Они объединяют различные методы для определения множеств элементов.

Используются те же операторы, что для идентификаторов множеств.

Операторы

Оператор	Описание
+	Объединение. Данная бинарная операция возвращает множество, состоящее из записей или элементов, принадлежащих любому из двух операндов множества.
-	Исключение. Данная бинарная операция возвращает множество записей или элементов, принадлежащих первому из двух операндов множества. Также, при использовании в качестве унарного оператора, она возвращает дополнительное множество.
*	Пересечение. Данная бинарная операция возвращает множество, состоящее из записей или элементов, принадлежащих обоим операндам множества.
/	Симметрическая разность (XOR). Данная бинарная операция возвращает множество, состоящее из записей или элементов, принадлежащих одному из операндов множества, но не сразу обоим.

Например, следующие два модификатора определяют то же множество значений поля:

- `<Year = {1997, "20*"}>`
- `<Year = {1997} + {"20*"}>`

Оба выражения выбирают 1997 и годы, которые начинаются с 20. Другими словами, это объединение двух условий.

Операторы множества также позволяют использовать более сложные определения. Пример.

```
<Year = {1997, "20*"} - {2000}>
```

Это выражение выберет те же годы, что выше, но также исключит год 2000.

.

Примеры: Выражения диаграммы для модификаторов множества с операторами множества

Примеры. Выражения диаграммы

Скрипт загрузки

Загрузите следующие данные через встроенную загрузку в редакторе загрузки данных, чтобы создать примеры с выражениями диаграммы, показанные ниже.

```
MyTable: Load  Year(Date) as Year, Date#(Date,'YYYY-MM-DD') as ISO_Date,  Date(Date#
(Date,'YYYY-MM-DD'),'M/D/YYYY') as US_Date,  Country, Product, Amount  Inline [Date, Country,
Product, Amount 2018-02-20, Canada, Washer, 6 2018-07-08, Germany, Anchor bolt, 10 2018-07-14,
Germany, Anchor bolt, 3 2018-08-31, France, Nut, 2 2018-09-02, Czech Republic, Bolt, 1 2019-
02-11, Czech Republic, Bolt, 3 2019-07-31, Czech Republic, Washer, 6 2020-03-13, France,
Anchor bolt, 1 2020-07-12, Canada, Anchor bolt, 8 2020-09-16, France, Washer, 1];
```

Выражения диаграммы

Создайте таблицу на листе Qlik Sense со следующими выражениями диаграммы.

Таблица. Модификаторы множества с операторами множества

Country	Sum (Amount)	Sum({<Year={">2018"}-{2020}>} Amount)	Sum ({<Country=-{Germany}>} Amount)	Sum({<Country={Germany}+P({<Product={Nut}>}Country)>} Amount)
Итоги	**41**	**9**	**28**	**17**
Canada	14	0	14	0
Czech Republic	10	9	10	0
France	4	0	4	4
Germany	13	0	0	13

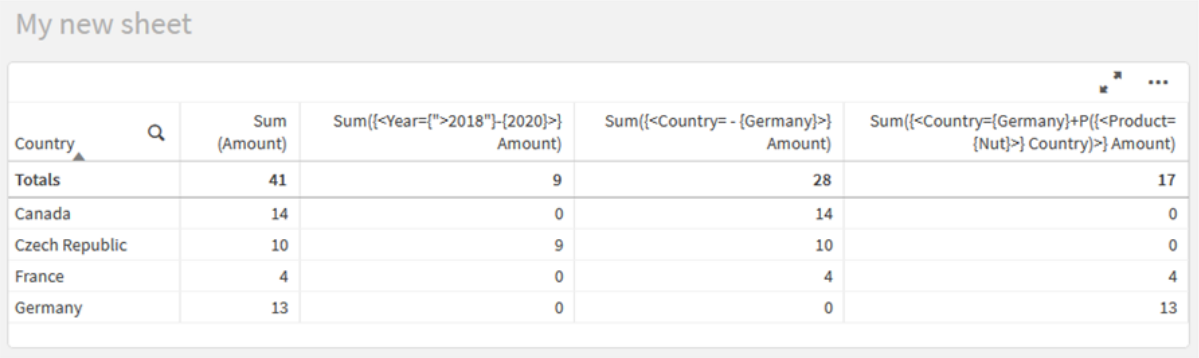
Объяснение

- Измерения:
  - `Country`
- Меры:
  - `Sum(Amount)`
    Суммировать `Amount` без выражения множества.
  - `Sum({<Year={">2018"}-{2020}>}Amount)`
    Суммировать `Amount` за все годы после `2018`, кроме `2020`.
  - `Sum({<Country=-{Germany}>}Amount)`

Суммировать `Amount` для всех стран, кроме `Germany`. Обратите внимание на унарный оператор исключения.

- `Sum({<Country={Germany}+P({<Product={Nut}>}Country)>}Amount)`
  Суммировать `Amount` для `Germany` и всех стран, связанных с продуктом `Nut`.

Модификаторы множества с операторами множества


My new sheet

Country	Sum (Amount)	Sum({<Year={">2018"}-{2020}>} Amount)	Sum({<Country= - {Germany}>} Amount)	Sum({<Country={Germany}+P({<Product={Nut}>} Country)>} Amount)
Totals	41	9	28	17
Canada	14	0	14	0
Czech Republic	10	9	10	0
France	4	0	4	4
Germany	13	0	0	13

Модификаторы множества с неявными операторами множества

Стандартный способ записи выборок в модификаторе множеств подразумевает использование знака равенства. Пример.

```
Year = {">2015"}
```

Выражение справа от знака равенства в модификаторе множества называется множеством элементов. Оно определяет множество отдельных значений полей, то есть выборку.

Эта нотация определяет новую выборку, игнорируя текущую выборку в поле. Таким образом, если идентификатор множества содержит выборку в этом поле, старая выборка будет заменена той, которая определена во множестве элементов.

Когда требуется создать выборку на основе текущей выборки в поле, необходимо использовать другое выражение.

Например, если требуется учитывать старую выборку и добавить требование для выбора лет после 2015 года, можно записать следующее выражение:

```
Year = Year * {">2015"}
```

Звездочка – это оператор множества, определяющий пересечение между текущей выборкой в `Year` и дополнительным требованием к выбору года после `2015`. Это можно записать по-другому:

```
Year *= {">2015"}
```

Другими словами, оператор присваивания (*=) неявно определяет пересечение.

Аналогично можно определить неявные объединения, исключения и симметричные разности, используя следующее: `+=`, `-=`, `/=`

Примеры: Выражения диаграммы для модификаторов множества с неявными операторами множества

Примеры. Выражения диаграммы

Скрипт загрузки

Загрузите следующие данные через встроенную загрузку в редакторе загрузки данных, чтобы создать примеры с выражениями диаграммы, показанные ниже.

```
 MyTable: Load  Year(Date) as Year, Date#(Date,'YYYY-MM-DD') as ISO_Date,  Date(Date#
(Date,'YYYY-MM-DD'),'M/D/YYYY') as US_Date,  Country, Product, Amount  Inline [Date, Country,
Product, Amount 2018-02-20, Canada, Washer, 6 2018-07-08, Germany, Anchor bolt, 10 2018-07-14,
Germany, Anchor bolt, 3 2018-08-31, France, Nut, 2 2018-09-02, Czech Republic, Bolt, 1 2019-
02-11, Czech Republic, Bolt, 3 2019-07-31, Czech Republic, Washer, 6 2020-03-13, France,
Anchor bolt, 1 2020-07-12, Canada, Anchor bolt, 8 2020-09-16, France, Washer, 1];
```

Выражения диаграммы с неявными операторами множества

Создайте таблицу на листе Qlik Sense со следующими выражениями диаграммы.

Выберите `Canada` и `Czech Republic` из списка стран.

Таблица. Выражения диаграммы с неявными операторами множества

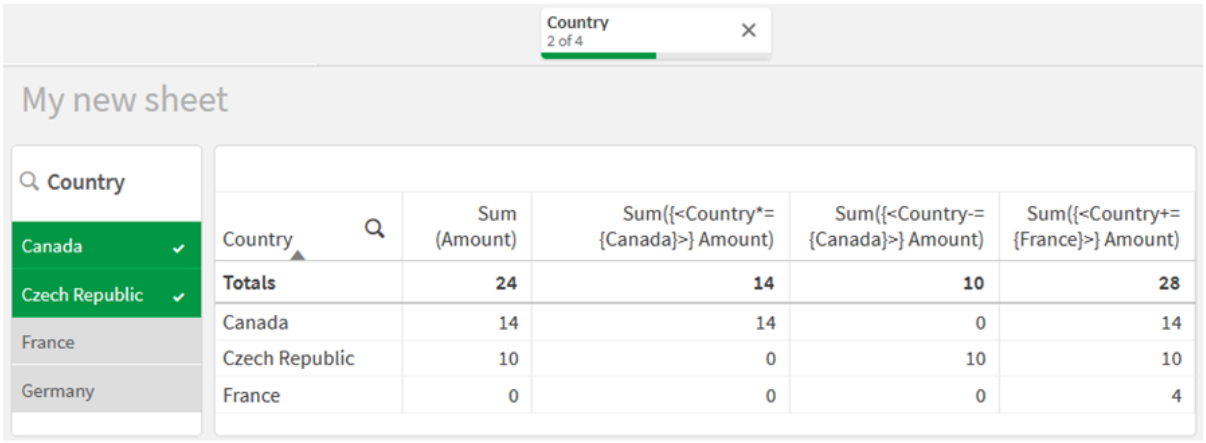
Country	Sum (Amount)	Sum({<Country*= {Canada}>} Amount)	Sum({<Country-= {Canada}>} Amount)	Sum({<Country+= {France}>} Amount)
Итоги	**24**	**14**	**10**	**28**
Canada	14	14	0	14
Czech Republic	10	0	10	10
France	0	0	0	4

Объяснение

- Измерения:
  - `Country`
- Меры:
  - `Sum(Amount)`
    Суммировать `Amount` для текущей выборки. Обратите внимание, что только `Canada` и `Czech Republic` имеют ненулевые значения.
  - `Sum({<Country*={Canada}>}Amount)`
    Суммировать `Amount` для текущей выборки, пересеченной требованием к `Country` — `Canada`. Если `Canada` не является частью пользовательской выборки, выражение множества возвращает пустое множество и столбец будет содержать `0` во всех строках.

- `Sum({<Country-={Canada}>}Amount)`
  Суммировать `Amount` для текущей выборки, предварительно исключив `Canada` из выборки `Country`. Если `Canada` не является частью пользовательской выборки, выражение множества не будет изменять числа.
- `Sum({<Country+={France}>}Amount)`
  Суммировать `Amount` для текущей выборки, предварительно добавив `France` в выборку `Country`. Если `France` уже является частью пользовательской выборки, выражение множества не будет изменять числа.

Модификаторы множества с неявными операторами множества



Country	Sum (Amount)	Sum({<Country*= {Canada}>} Amount)	Sum({<Country-= {Canada}>} Amount)	Sum({<Country+= {France}>} Amount)
Totals	**24**	**14**	**10**	**28**
Canada	14	14	0	14
Czech Republic	10	0	10	10
France	0	0	0	4

Модификаторы множества с использованием функций множества

Иногда может требоваться определить множество значений поля с помощью вложенного определения множества. Например, может потребоваться выбрать всех клиентов, которые приобрели определенный продукт, не выбирая этот продукт.

В таких случаях следует использовать функции множества элементов `P()` и `E()`. Они возвращают множества элементов с возможными и исключенными значениями поля соответственно. Внутри скобок можно указать рассматриваемое поле и выражение множества, которое определяет область действия. Пример.

```
P({1<Year = {2021}>} Customer)
```

Это выражение вернет множество клиентов, у которых были транзакции в 2021 году. Затем результат можно использовать в модификаторе множества. Пример.

```
Sum({<Customer = P({1<Year = {2021}>} Customer)>} Amount)
```

Это выражение множества выбирает указанных клиентов, но не ограничивает выборку 2021 годом.

Эти функции не могут использоваться в других выражениях.

Кроме того, внутри функций множества элементов можно использовать только натуральные множества. Это множество записей, которое можно определить путем простой выборки.

Например, множество, заданное значением `{1-$}`, не всегда можно определить путем выборки, следовательно, оно не является натуральным. При использовании этих функций с ненатуральными множествами, результаты могут быть неудовлетворительными.

Примеры: Выражения диаграммы для модификаторов множества с использованием функций множества

Примеры. Выражения диаграммы

Скрипт загрузки

Загрузите следующие данные через встроенную загрузку в редакторе загрузки данных, чтобы создать примеры с выражениями диаграммы, показанные ниже.

```
MyTable: Load  Year(Date) as Year, Date#(Date,'YYYY-MM-DD') as ISO_Date,  Date(Date#
(Date,'YYYY-MM-DD'),'M/D/YYYY') as US_Date,  Country, Product, Amount  Inline [Date, Country,
Product, Amount 2018-02-20, Canada, Washer, 6 2018-07-08, Germany, Anchor bolt, 10 2018-07-14,
Germany, Anchor bolt, 3 2018-08-31, France, Nut, 2 2018-09-02, Czech Republic, Bolt, 1 2019-
02-11, Czech Republic, Bolt, 3 2019-07-31, Czech Republic, Washer, 6 2020-03-13, France,
Anchor bolt, 1 2020-07-12, Canada, Anchor bolt, 8 2020-09-16, France, Washer, 1];
```

Выражения диаграммы

Создайте таблицу на листе Qlik Sense со следующими выражениями диаграммы.

Таблица. Модификаторы множества с использованием функций множества

Country	Sum (Amount)	Sum({<Country=P ({<Year= {2019}>}Country)>} Amount)	Sum({<Product=P ({<Year= {2019}>}Product)>} Amount)	Sum({<Country=E ({<Product= {Washer}>}Country)>} Amount)
Итоги	**41**	**10**	**17**	**13**
Canada	14	0	6	0
Czech Republic	10	10	10	0
France	4	0	1	0
Germany	13	0	0	13

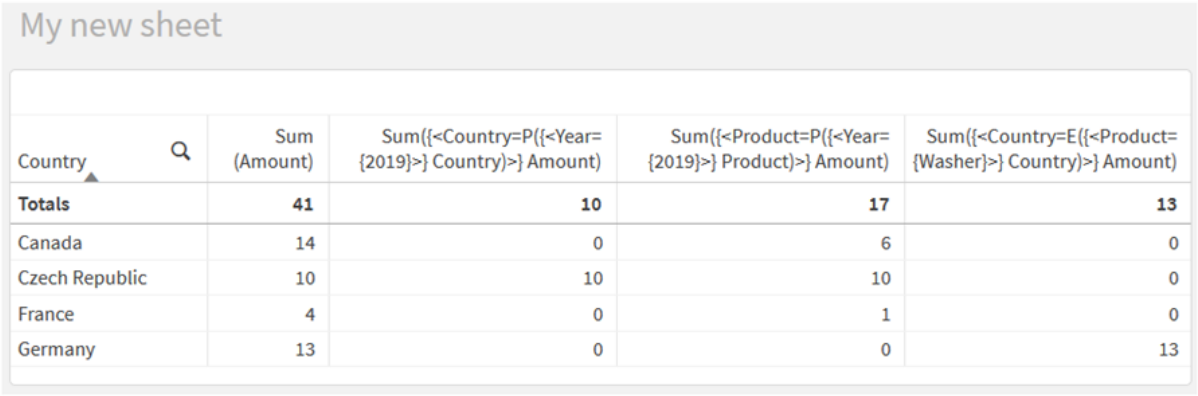
Объяснение

- Измерения:
  - `Country`
- Меры:
  - `Sum(Amount)`
    Суммировать `Amount` без выражения множества.
  - `Sum({<Country=P({<Year={2019}>} Country)>} Amount)`

Суммировать `Amount` для всех стран, связанных с годом `2019`. Однако при этом вычисление не ограничивается до `2019`.

- `Sum({<Product=P({<Year={2019}>} Product)>} Amount)`
  Суммировать `Amount` для всех продуктов, связанных с годом `2019`. Однако при этом вычисление не ограничивается до `2019`.
- `Sum({<Country=E({<Product={Washer}>} Country)>} Amount)`
  Суммировать `Amount` для всех стран, не связанных с годом `Washer`.

Модификаторы множества с использованием функций множества


My new sheet

Country	Sum (Amount)	Sum({<Country=P({<Year={2019}>} Country)>} Amount)	Sum({<Product=P({<Year={2019}>} Product)>} Amount)	Sum({<Country=E({<Product={Washer}>} Country)>} Amount)
Totals	**41**	**10**	**17**	**13**
Canada	14	0	6	0
Czech Republic	10	10	10	0
France	4	0	1	0
Germany	13	0	0	13

Учебное пособие – создание выражения множества

Для поддержки анализа данных можно создавать выражения множества. В этом контексте анализ часто называют анализом множеств. Анализ множеств предлагает способ определения области, отличной от множества записей, определяемого текущей выборкой в приложении.

Что вы узнаете

В этом учебном пособии представлены данные и выражения диаграммы для построения выражений множества с использованием идентификаторов, операторов и модификаторов множества.

Кому следует ознакомиться с этим учебным пособием

Это учебное пособие предназначено для разработчиков приложений, которые умеют работать с редактором скриптов и выражениями диаграмм.

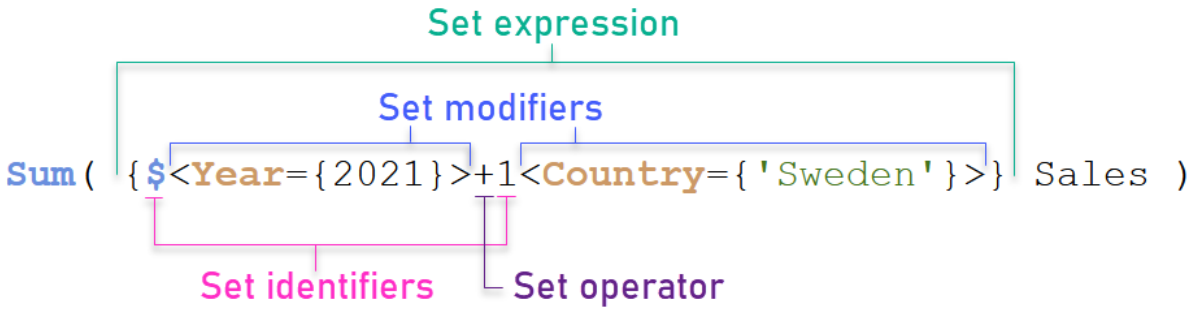
Что требуется сделать перед началом работы

Доступ к Qlik Sense Enterprise Professional, который позволяет загружать данные и создавать приложения.

Элементы в выражении множества

Выражения множества заключаются в функцию агрегирования, например `Sum()`, `Max()`, `Min()`, `Avg()` или `Count()`. Выражения множества создаются из строительных блоков, известных как элементы. Этими элементами являются модификаторы, идентификаторы и операторы множества.

Элементы в выражении множества



К примеру, приведенное выше выражение множества строится на основе агрегирования `Sum (Sales)`. Выражение множества заключено во внешние фигурные скобки: `{ }`

Первый операнд множества: `$<Year={2021}>`

Этот операнд возвращает объем продаж за `2021` год для текущей выборки. Модификатор, `<Year={2021}>`, содержит выбор года 2021. Идентификатор множества `$` указывает, что выражение множества основано на текущей выборке.

Второй операнд множества: `1<Country={'Sweden'}>`

Этот операнд возвращает Sales для Sweden. Модификатор, `<Country={'Sweden'}>`, содержит выбор страны Sweden. Идентификатор множества `1` указывает, что выборки в приложении будут игнорироваться.

Наконец, оператор множества `+` указывает на то, что выражение возвращает множество, состоящее из записей, принадлежащих любому из двух операндов множества.

Учебное пособие по созданию выражения множества

Выполните следующие процедуры для создания выражений множества, показанных в этом учебном пособии.

Создание нового приложения и загрузка данных

Выполните следующие действия.

1. Создайте новое приложение.
2. Щелкните **Редактор скриптов**. Или можно щелкнуть **Подготовить** > **Редактор загрузки данных** на панели навигации.
3. Создайте новый раздел в **редакторе загрузки данных**.
4. Скопируйте следующие данные и вставьте их в новый раздел: *Данные учебного пособия по выражениям множества (page 215)*
5. Щелкните команду **Загрузить данные**. Данные загружаются в качестве встроенной загрузки.

Создание выражений множества с использованием модификаторов

Модификатор множества состоит из одного или нескольких имен полей, для каждого имени перечислены значения, применяемые к полю. Модификатор заключен в угловые скобки. Например, в этом выражении множества:

```
Sum ( {<Year = {2015}>} Sales )
```

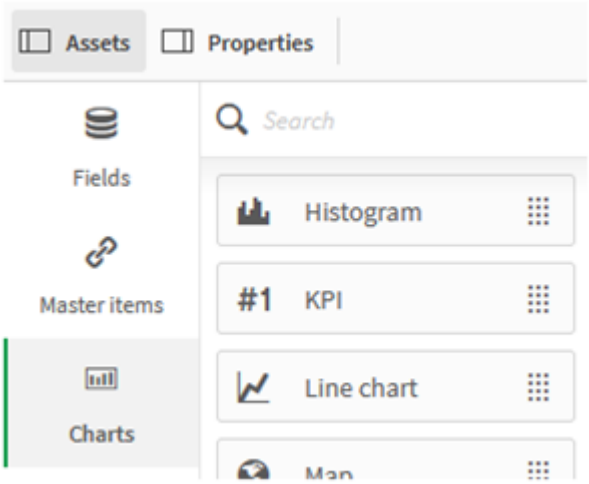
Модификатор:

```
<Year = {2015}>
```

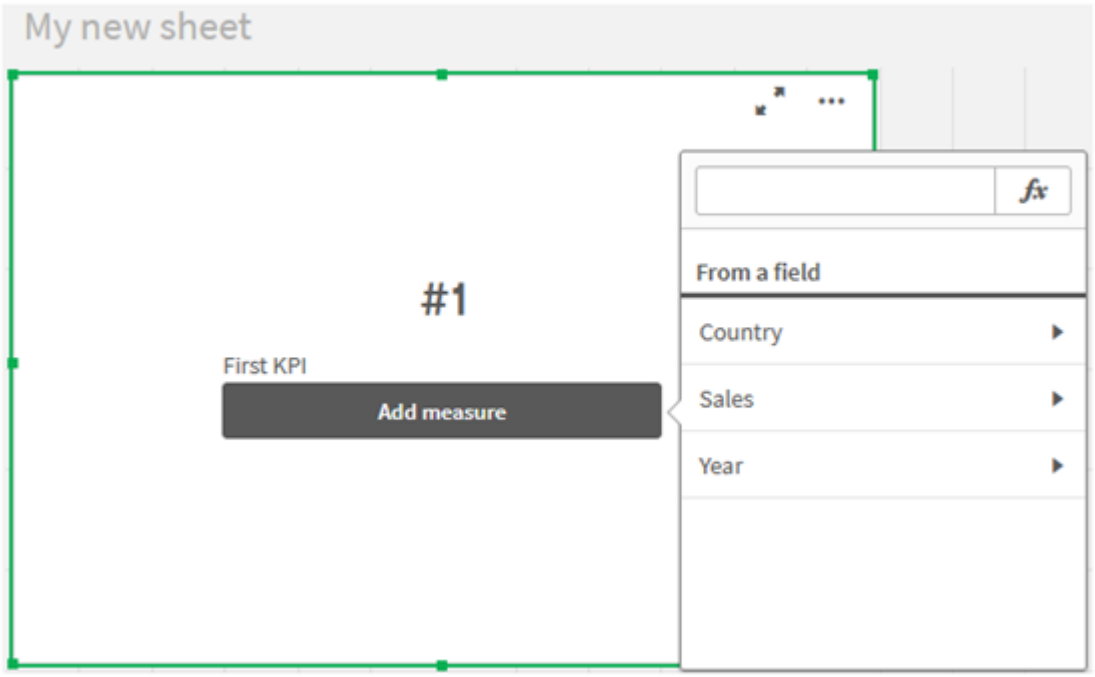
Этот модификатор задает выборку данных из 2015 года. Фигурные скобки, в которые заключен модификатор, обозначают выражение множества.

Выполните следующие действия.

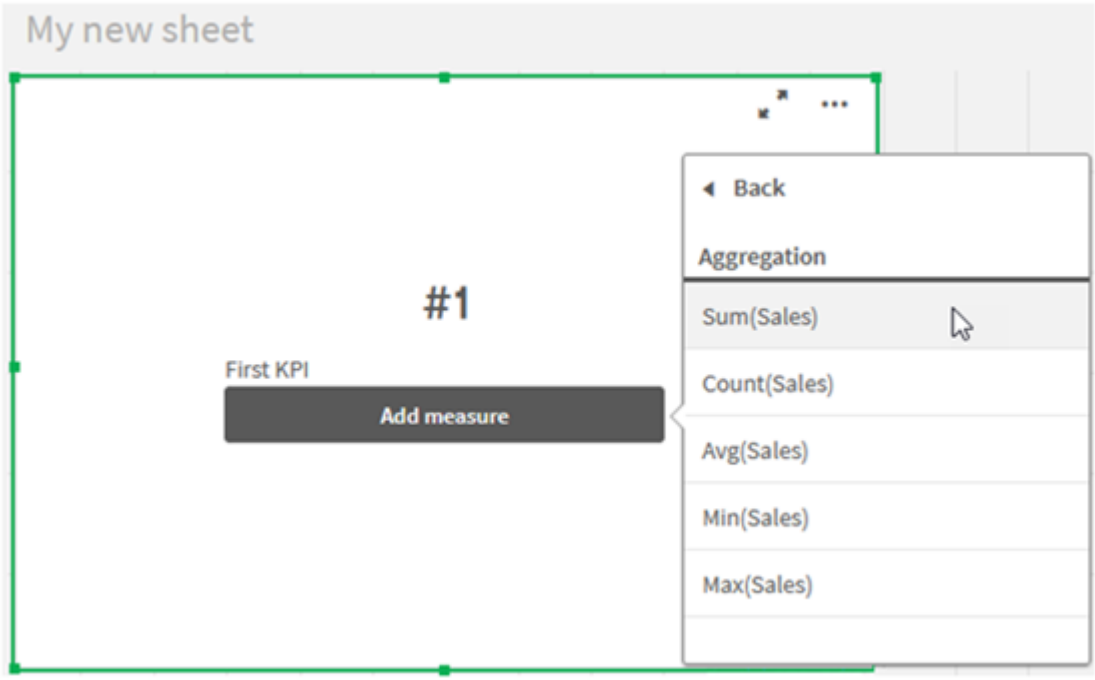
1. На листе откройте панель **Ресурсы** с панели навигации, затем выберите **Диаграммы**.



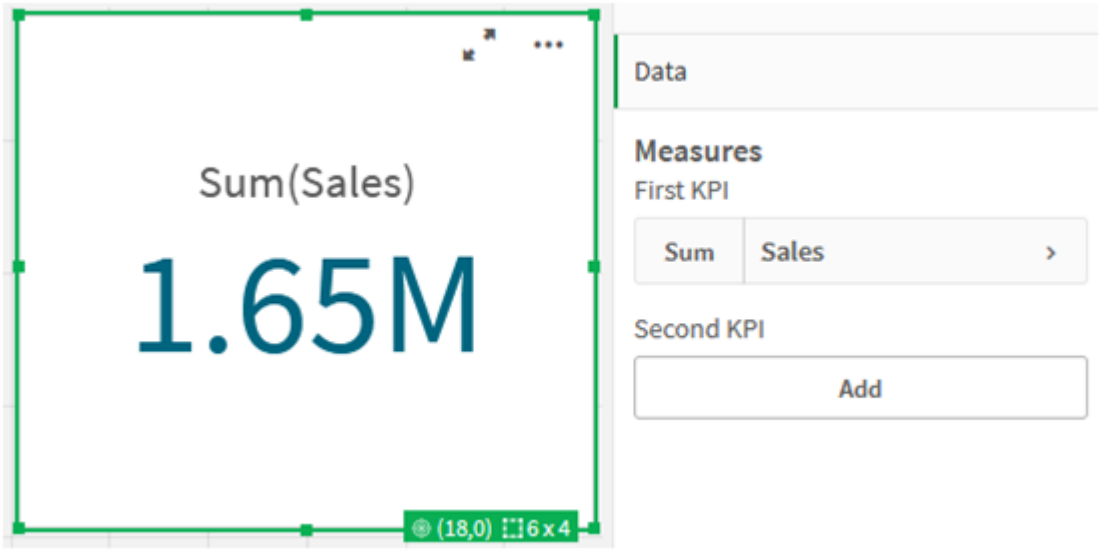
2. Перетащите **ключевой показатель эффективности** на лист, затем щелкните **Добавить меру**.



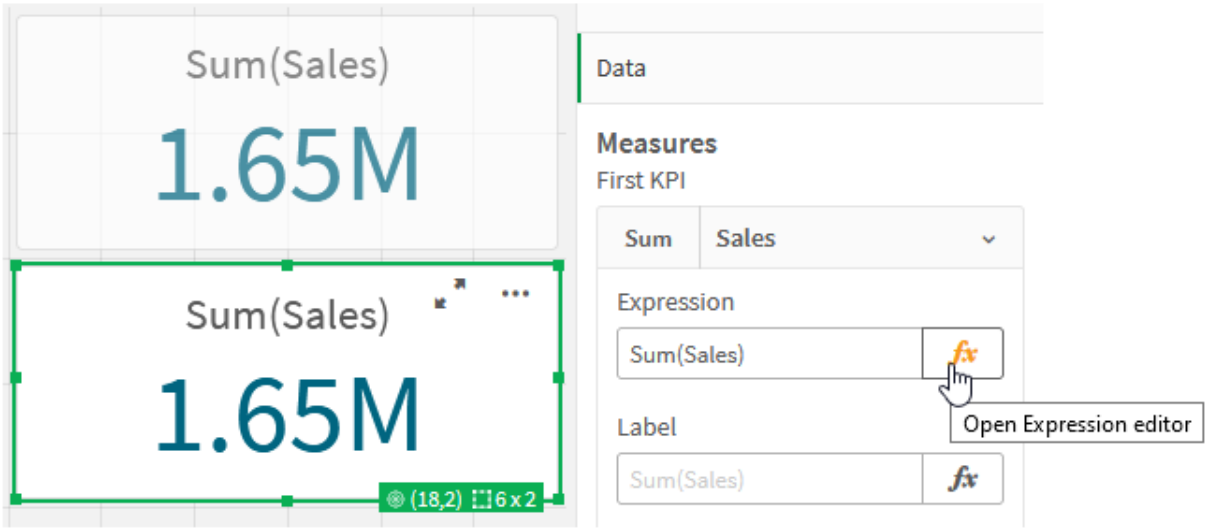
3. Щелкните `Sales`, затем выберите `Sum(Sales)` для агрегирования.



KPI показывает сумму объемов продаж за все годы.

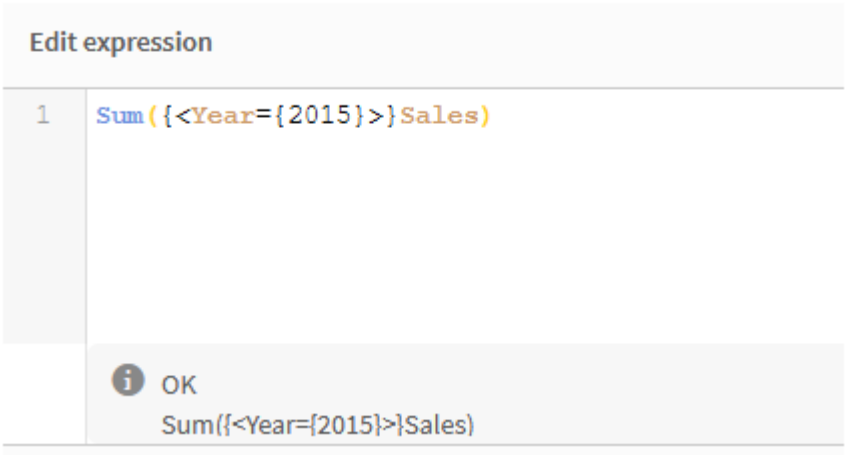


4. Скопируйте и вставьте KPI, чтобы создать новый объект KPI.
5. Щелкните созданный KPI выберите **Продажи** в области **Меры** и затем щелкните **Открыть редактор выражения**.

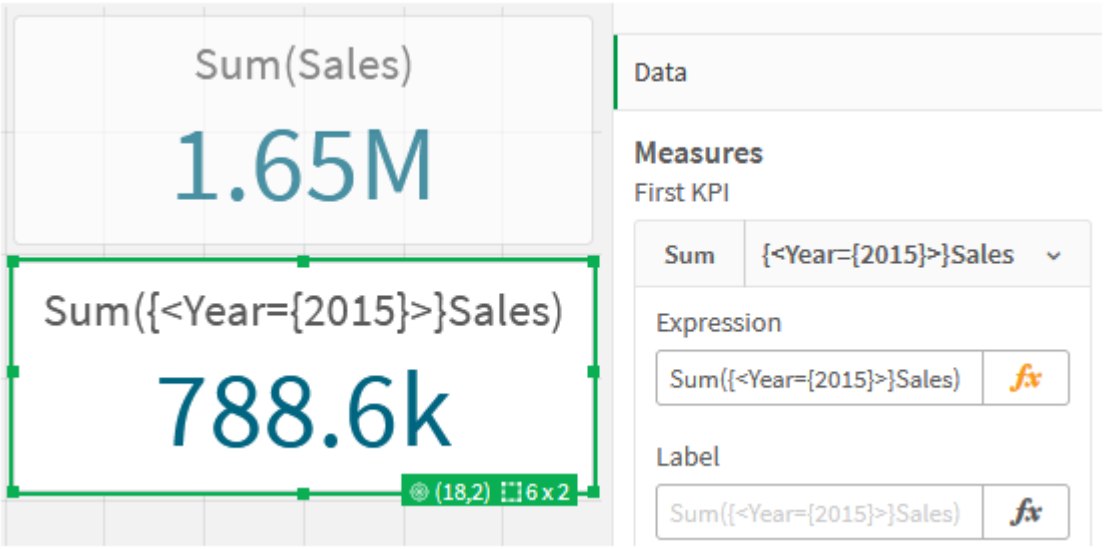


Редактор выражения открывается с агрегацией `Sum(Sales)`.

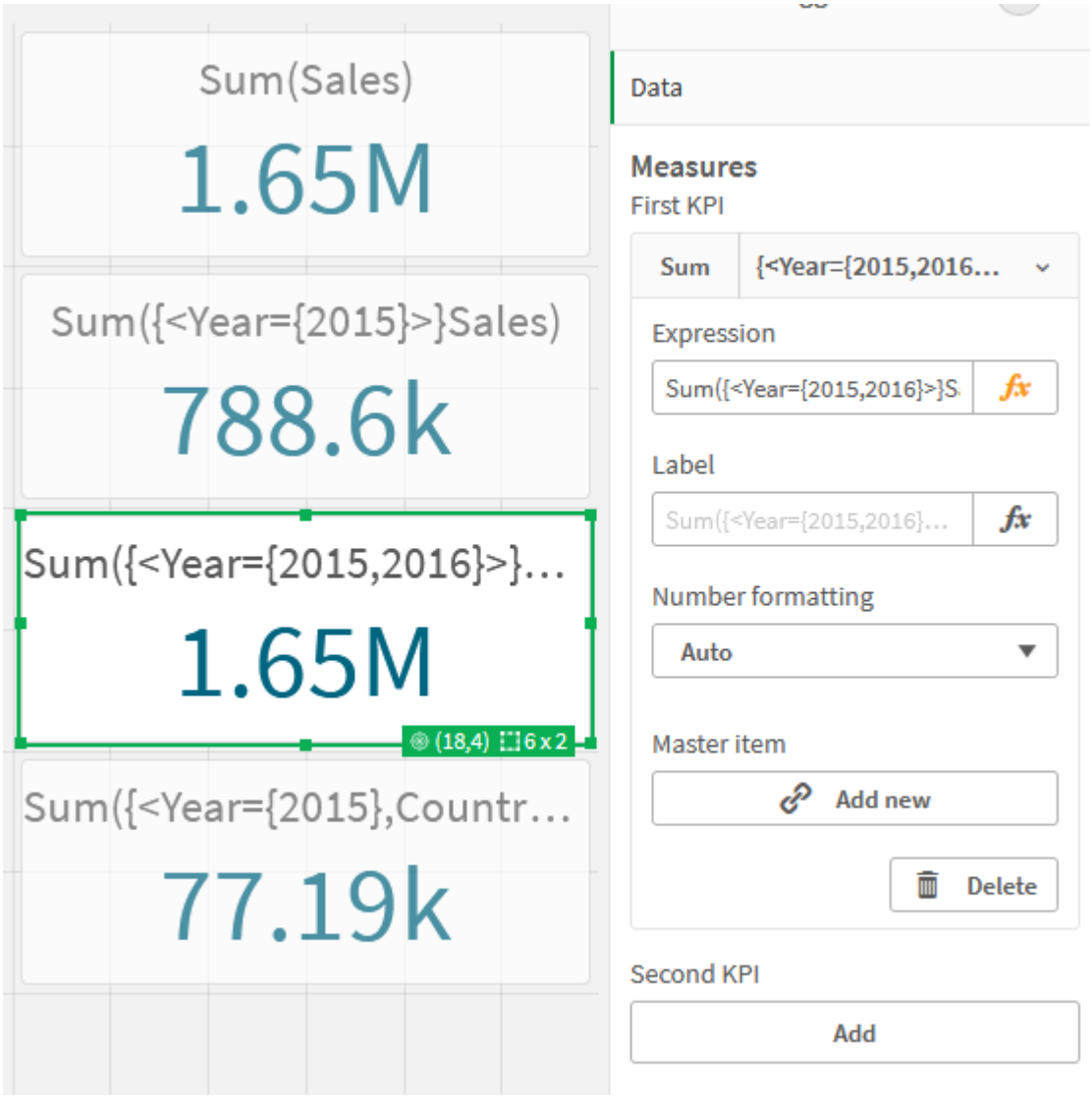
6. В редакторе выражения создайте выражение, чтобы суммировать Sales только за 2015 год.
   i. Добавьте фигурные скобки, чтобы обозначить выражение множества: `Sum({}Sales)`
   i. Добавьте угловые скобки, чтобы обозначить модификатор множества: `Sum({<>}Sales)`
   ii. В угловых скобках добавьте поле, которое необходимо выбрать, в данном случае `Year`, а затем знак «равно». Затем заключите 2015 в еще одну пару фигурных скобок. Полученный модификатор множества: `{<Year={2015}>}`.
   Выражение полностью:
   `Sum({<Year={2015}>}Sales)`



   iii. Нажмите кнопку **Применить** чтобы сохранить выражение и закрыть редактор выражения. Сумма Sales за 2015 год отображается в KPI.



7. Создайте еще два ключевых показателя эффективности, используя следующие выражения:
   `Sum({<Year={2015,2016}>}Sales)`
   В вышеприведенном примере модификатором является `<Year={2015,2016}>`. Выражение вернет сумму Sales за 2015 и 2016 гг.
   `Sum({<Year={2015},Country={'Germany'}>} Sales)`
   В вышеприведенном примере модификатором является `<Year={2015}, Country={'Germany'}>`. Выражение вернет сумму Sales за 2015 г., где этот год пересекается с Germany.
   Ключевые показатели эффективности, использующие модификаторы множества



Добавление идентификаторов множества

В приведенных выше выражениях множества за основу будут взяты текущие выборки, так как не использовался идентификатор. На следующем этапе добавьте идентификаторы, чтобы задать поведение при выполнении выборки.

Выполните следующие действия.

На листе составьте или скопируйте следующие выражения множества:

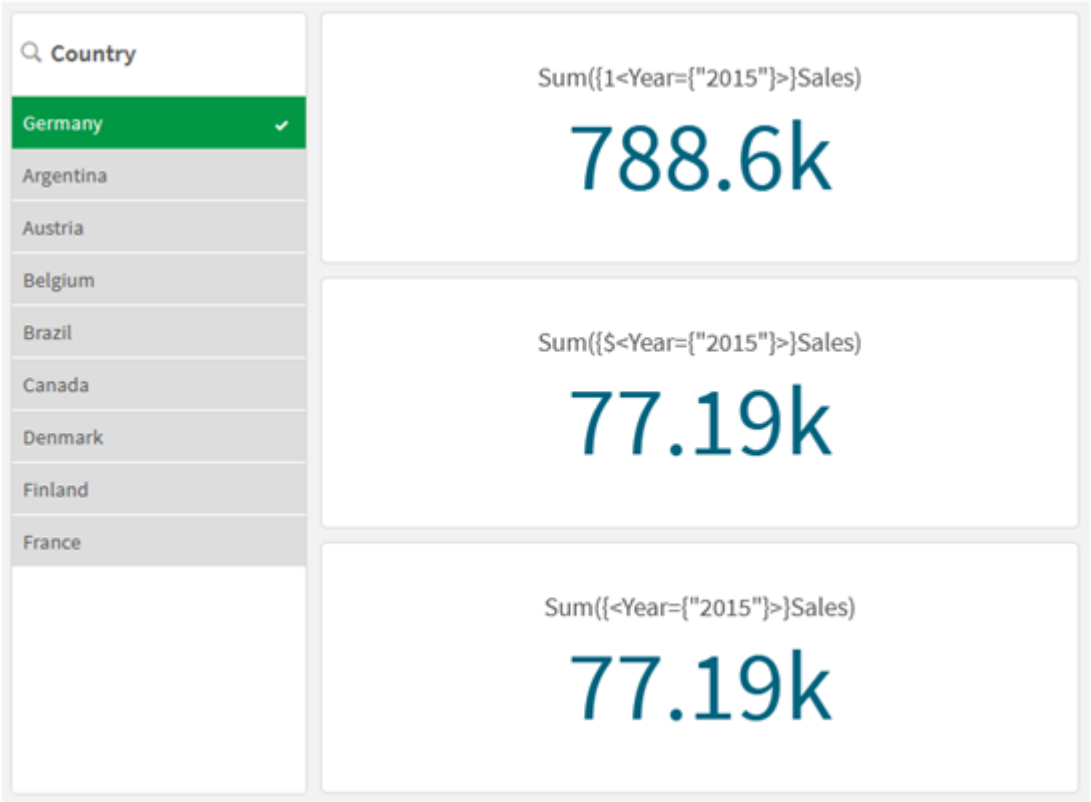
```
Sum({$<Year={"2015"}>}Sales
```

Идентификатор `$` создаст выражение множества на основе текущих выборок данных. Функция работает таким же образом, когда идентификатор не используется.

```
Sum({1<Year={"2015"}>}Sales)
```

Использование идентификатора `1` приведет к тому, что агрегирование `Sum(Sales)` по 2015 г. будет игнорировать текущую выборку. Значение агрегирования не изменится, если пользователь сделает другие выборки. Например, если выбрано значение Germany ниже, значение агрегирования по сумме для 2015 г. не изменяется.

Ключевые показатели эффективности, использующие модификаторы и идентификаторы множества



Добавление операторов

Операторы множества используются для включения, исключения или пересечения множеств данных. Все операторы используют множества в качестве операндов и в результате возвращают множество.

Можно использовать операторы множеств в двух различных ситуациях:

- Выполнение операции множества с идентификаторами множества, представляет множества записей в данных.
- Выполнение операции множества с множествами элементов, значениями полей или внутри модификатора множества.

Выполните следующие действия.

На листе составьте или скопируйте следующее выражение множества:

```
Sum({$<Year={2015}>+1<Country={'Germany'}>}Sales)
```

Здесь оператор «плюс» (`(+)`) объединяет наборы данных для 2015 и Germany. Как объяснялось для идентификаторов множества выше, идентификатор со знаком доллара (`($)`) означает, что для первого операнда, `<Year={2015}>`, будут учитываться текущие выборки. Идентификатор `1` означает, что выборка будет игнорироваться для второго операнда, `<Country={'Germany'}>`.

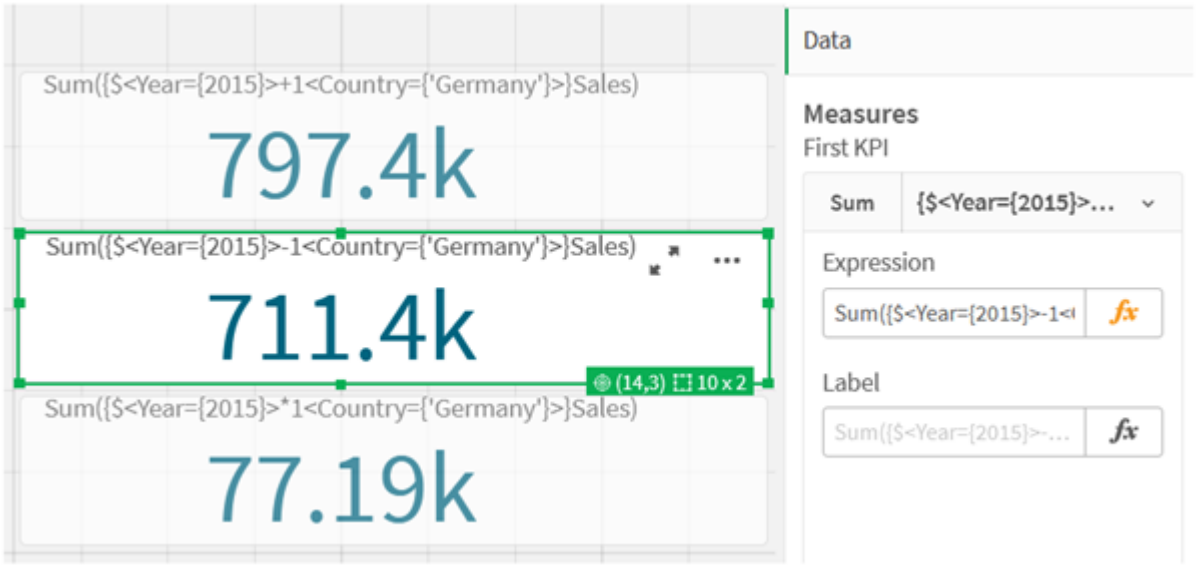
Ключевой показатель эффективности с использованием оператора «плюс» (+)

Также можно использовать знак «минус» (`(-)`), чтобы вернуть набор данных, включающий записи, которые относятся к 2015 году, но не к Germany. Или используйте звездочку (`(*)`), чтобы вернуть множество, содержащее записи, относящиеся к обоим множествам.

```
Sum({$<Year={2015}>-1<Country={'Germany'}>}Sales)
```

```
Sum({$<Year={2015}>*1<Country={'Germany'}>}Sales)
```

Ключевые показатели эффективности, использующие операторы



Данные учебного пособия по выражениям множества

Скрипт загрузки

Загрузите следующие данные через встроенную загрузку, чтобы создать выражения диаграммы в учебном пособии.

```
//Create table SalesByCountry SalesByCountry: Load * Inline [ Country, Year, Sales Argentina,
2016, 66295.03 Argentina, 2015, 140037.89 Austria, 2016, 54166.09 Austria, 2015, 182739.87
Belgium, 2016, 182766.87 Belgium, 2015, 178042.33 Brazil, 2016, 174492.67 Brazil, 2015,
2104.22 Canada, 2016, 101801.33 Canada, 2015, 40288.25 Denmark, 2016, 45273.25 Denmark, 2015,
106938.41 Finland, 2016, 107565.55 Finland, 2015, 30583.44 France, 2016, 115644.26 France,
2015, 30696.98 Germany, 2016, 8775.18 Germany, 2015, 77185.68 ];
```

Синтаксис выражений множества

Полный синтаксис (не включая дополнительное использование стандартных скобок для определения последовательности) описан с помощью формы Backus-Naur:

```
set_expression ::= { set_entity { set_operator set_entity } }
set_entity ::= set_identifier [ set_modifier ] | set_modifier
set_identifier ::= 1 | $ | $N | $_N | bookmark_id | bookmark_name
set_operator ::= + | - | * | /
set_modifier ::= < field_selection {, field_selection } >
field_selection ::= field_name [ = | += | -= | *= | /= ] element_set_
expression
element_set_expression ::= [ - ] element_set { set_operator element_set }
element_set ::= [ field_name ] | { element_list } | element_function
element_list ::= element { , element }
element_function ::= ( P | E ) ( [set_expression] [field_name] )
element ::= field_value | " search_mask "
```

3.3 Общий синтаксис выражений диаграммы

Следующая структура общего синтаксиса может использоваться для выражений диаграммы со многими дополнительными параметрами:

```
expression ::= ( constant | expressionname | operator1 expression | expression operator2
expression | function | aggregation function | (expression ) )
```

где:

элемент **constant** – строка (текст, дата или время), заключенная в одиночные прямые кавычки, или число. Константы записываются без разделителя тысяч, а в качестве разделителя десятичной части используется десятичный разделитель.

элемент **expressionname** – имя (метка) другого выражения в той же диаграмме.

элемент **operator1** – унарный оператор (работающий над одним выражением, справа).

элемент **operator2** – бинарный оператор (работающий над двумя выражениями, по одному с каждой стороны).

```
function ::= functionname ( parameters )
parameters ::= expression { , expression }
```

Число и типы параметров не являются произвольными. Они зависят от используемой функции.

```
aggregationfunction ::= aggregationfunctionname ( parameters2 )
parameters2 ::= aggrexpression { , aggrexpression }
```

Число и типы параметров не являются произвольными. Они зависят от используемой функции.

3.4 Общий синтаксис для агрегирования

Следующая структура общего синтаксиса может использоваться для агрегирований со многими дополнительными параметрами:

```
aggrexpression ::= ( fieldref | operator1 aggrexpression | aggrexpression operator2
aggrexpression | functioninaggr | ( aggrexpression ) )
```

Элемент **fieldref** является именем поля.

```
functionaggr ::= functionname ( parameters2 )
```

Выражения и функции, следовательно, могут свободно размещаться до тех пор, пока элемент **fieldref** включен в одну определенную функцию агрегирования, при условии, что Qlik Sense не выдаст сообщений об ошибках, когда выражение возвращает интерпретируемое значение.

4 Операторы

В этом разделе описаны операторы, которые можно использовать в программе Qlik Sense. Существует два типа операторов:

- унарные операторы (принимают только один операнд);
- бинарные операторы (принимают два операнда).

Большинство операторов являются бинарными.

Можно определить следующие операторы:

- Побитовые операторы
- Логические операторы
- Числовые операторы
- Реляционные операторы
- Строковые операторы

4.1 Побитовые операторы

Все побитовые операторы преобразуют (усекают) операнды в целые (32-разрядные) числа со знаком и возвращают результат тем же способом. Все операции выполняются поразрядно (бит за битом). Если операнд не может быть интерпретирован как число, операция возвратит значение NULL.

Побитовые операторы

Operator	Полное имя	Описание
bitnot	Побитовое отрицание.	Унарный оператор. Операция применяет логическое отрицание к каждому биту операнда. **Пример:** Элемент `bitnot 17` возвращает -18
bitand	Побитовое И.	Операция применяет логическое И к каждому биту операндов. **Пример:** Элемент `17 bitand 7` возвращает 1
bitor	Побитовое ИЛИ.	Операция применяет логическое ИЛИ к каждому биту операндов. **Пример:** Элемент `17 bitor 7` возвращает 23

Operator	Полное имя	Описание
bitxor	Побитовое исключающее ИЛИ.	Операция применяет логическое исключающее ИЛИ к каждому биту операндов. **Пример:** Элемент `17 bitxor 7` возвращает 22
>>	Битовый сдвиг вправо.	Операция возвращает первый операнд, сдвинутый вправо. Количество шагов определяется во втором операнде. **Пример:** Элемент `8 >> 2` возвращает 2
<<	Битовый сдвиг влево.	Операция возвращает первый операнд, сдвинутый влево. Количество шагов определяется во втором операнде. **Пример:** Элемент `8 << 2` возвращает 32

4.2 Логические операторы

Все логические операторы интерпретируют операнды в соответствии с определенной логикой и выдают результат True (-1) или False (0).

Логические операторы

Operator	Описание
not	Логическое отрицание. Один из нескольких унарных операторов. Операция возвращает логическое отрицание операнда.
and	Логическое И. Операция применяет логическое И к операндам.
or	Логическое ИЛИ. Операция возвращает логическое ИЛИ операндов.
Xor	Логическое исключающее ИЛИ. Операция возвращает результат операции логического исключающее ИЛИ операндов. Т. е. операция подобна логическому ИЛИ за исключением того, что, если оба операнда имеют значение True, результат имеет значение False.

4.3 Числовые операторы

Все числовые операторы используют числовые значения операндов и возвращают числовое значение в качестве результата.

Числовые операторы

Operator	Описание
+	Знак положительного числа (унарный оператор) или арифметического сложения. Бинарная операция возвращает сумму двух операндов.
-	Знак отрицательного числа (унарный оператор) или арифметического вычитания. Унарная операция возвращает операнд, умноженный на -1, а бинарная операция – разницу двух операндов.
*	Арифметическое умножение. Операция возвращает произведение двух операндов.
/	Арифметическое деление. Операция возвращает частное двух операндов.

4.4 Реляционные операторы

Все реляционные операторы сравнивают значения операндов и возвращают в качестве результата значения True (-1) или False (0). Все реляционные операторы являются бинарными.

Реляционные операторы

Operator	Описание
<	Меньше. Числовое сравнение выполняется, если оба операнда можно интерпретировать в числовом виде. Операция возвращает логическое значение оценки результата сравнения.
<=	Меньше или равно. Числовое сравнение выполняется, если оба операнда можно интерпретировать в числовом виде. Операция возвращает логическое значение оценки результата сравнения.
>	Больше. Числовое сравнение выполняется, если оба операнда можно интерпретировать в числовом виде. Операция возвращает логическое значение оценки результата сравнения.
>=	Больше или равно. Числовое сравнение выполняется, если оба операнда можно интерпретировать в числовом виде. Операция возвращает логическое значение оценки результата сравнения.
=	Равно. Числовое сравнение выполняется, если оба операнда можно интерпретировать в числовом виде. Операция возвращает логическое значение оценки результата сравнения.

Operator	Описание
<>	Не равно. Числовое сравнение выполняется, если оба операнда можно интерпретировать в числовом виде. Операция возвращает логическое значение оценки результата сравнения.
precedes	В отличие от оператора **<**, перед сравнением не предпринимается попытка выполнить числовую интерпретацию значений аргументов. Операция возвращает значение true, если значение слева от оператора имеет текстовое представление, которое предшествует текстовому представлению значения справа в сравнении строк. **Пример:** `'1 ' precedes ' 2'` возвращает FALSE `' 1' precedes ' 2'` возвращает TRUE в качестве значения пробела ASCII (' '), который имеет меньшее значение, чем значение числа ASCII. Сравните с: `'1 ' < ' 2'` возвращает TRUE `' 1' < ' 2'` возвращает TRUE
follows	В отличие от оператора **>**, перед сравнением не предпринимается попытка выполнить числовую интерпретацию значений аргументов. Операция возвращает значение true, если значение слева от оператора имеет текстовое представление, которое находится после текстового представления значения справа в сравнении строк. **Пример:** `' 2' follows '1'` возвращает FALSE ' 2' follows ' 1' возвращает TRUE в качестве значения пробела ASCII (' '), который имеет меньшее значение, чем значение числа ASCII. Сравните с: `' 2' > ' 1'` возвращает TRUE `' 2' > '1 '` возвращает TRUE

4.5 Строковые операторы

Существует два строковых оператора. Один из них использует строковые значения операндов и возвращает строку в качестве результата. Другой сравнивает операнды и возвращает булево значение, указывающее на совпадение.

&

Сцепление строк. В результате операции возвращается текстовая строка, состоящая из двух последовательно идущих строк операндов.

Пример:

`'abc' & 'xyz'` возвращает 'abcxyz'

like

Сравнение строки со знаками подстановки. В результате операции возвращается булево значение True (-1), если строка перед оператором совпадает со строкой после оператора. Во второй строке могут использоваться знаки подстановки * (любое количество произвольных символов) или ? (один произвольный символ).

Пример:

Элемент `'abc' like 'a*'` возвращает True (-1)

Элемент `'abcd' like 'a?c*'` возвращает True (-1)

Элемент `'abc' like 'a??bc'` возвращает False (0)

5 Функции скрипта и диаграммы

Преобразование и агрегирование данных с использованием функций в скриптах загрузки данных и выражений диаграммы.

Многие функции можно использовать таким же образом как в скриптах загрузки данных, так и в выражениях диаграмм, но есть несколько исключений:

- Некоторые функции можно использовать только в скриптах загрузки данных, это функции скрипта.
- Некоторые функции можно использовать только в выражениях диаграммы, это функции диаграммы.
- Некоторые функции можно использовать как в скриптах загрузки данных, так и в выражениях диаграмм, но существуют различия в параметрах и применении. Это описывается в отдельных темах, которые называются «Функции скрипта» или «Функции диаграммы».

5.1 Аналитические подключения для серверных расширений (SSE)

Функции, реализующиеся при поддержке аналитических подключений, будут видны только после настройки аналитических подключений и запуска Qlik Sense.

Сведения о порядке настройки аналитических подключений в QMC см. в разделе «Создание аналитического подключения» в руководстве Управление сайтами Qlik Sense.

Для настройки аналитических подключений в Qlik Sense Desktop необходимо изменить файл *Settings.ini*, см. тему «Настройка аналитических подключений в Qlik Sense Desktop» в руководстве Qlik Sense Desktop.

5.2 Функции агрегирования

Семейство функций, известных как функции агрегирования, состоит из функций, для которых несколько значений поля являются вводимым значением и которые возвращают один результат на группу. В данных функциях группирование определяется измерением диаграммы или предложением **group by** в операторе скрипта.

В число функций агрегирования входят функции **Sum()**, **Count()**, **Min()**, **Max()** и многие другие.

Большинство функций агрегирования можно использовать как в скрипте загрузки данных, так и в выражениях диаграмм, но синтаксис имеет различия.

Ограничения:

Параметр функции агрегирования не должен содержать другие функции агрегирования, кроме внутреннего агрегирования, содержащего классификатор **TOTAL**. Для получения более расширенных вложенных агрегирований необходимо использовать расширенную функцию **Aggr** вместе с указанным измерением.

Использование функций агрегирования в скрипте загрузки данных

Функции агрегирования могут использоваться только внутри операторов **LOAD** и **SELECT**.

Использование функций агрегирования в выражениях диаграмм

Параметр функции агрегирования не должен содержать другие функции агрегирования, кроме внутреннего агрегирования, содержащего классификатор **TOTAL**. Для получения более расширенных вложенных агрегирований необходимо использовать расширенную функцию **Aggr** вместе с указанным измерением.

Функция агрегирования агрегирует набор возможных записей, определенных выборкой. Однако альтернативное множество записей может быть определено выражением множества в анализе множеств.

Как вычисляются агрегирования

Агрегирование циклически проходит по записям определенной таблицы, агрегируя их. Например, **Count**(<Field>) будет считать количество записей в таблице, в которой находится <Field>. Если необходимо агрегировать только уникальные значения поля, необходимо использовать предложение **distinct**, например **Count**(**distinct** <Field>).

Если функция агрегирования содержит поля из различных таблиц, она циклически выполнится для записей векторного произведения таблиц из составляющих полей. Это снижает производительность, и поэтому таких агрегирований нужно избегать, особенно при большом объеме данных.

Агрегирование ключевых полей

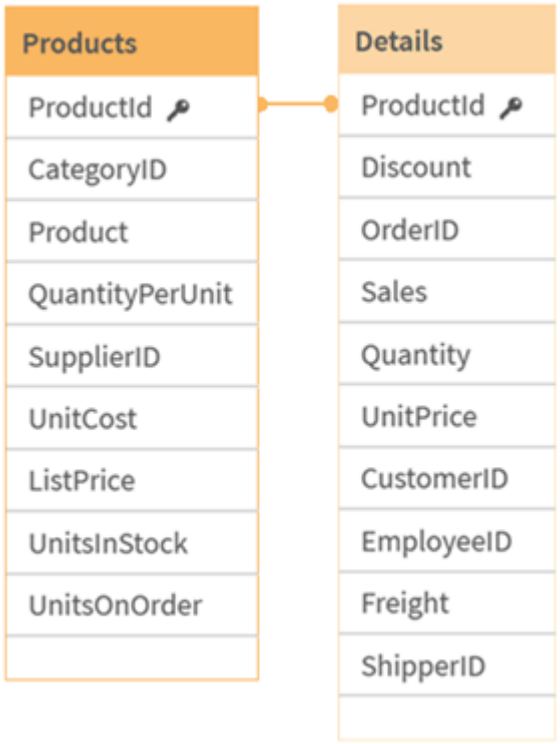
Способ вычисления агрегирований определяет, что нельзя агрегировать ключевые поля, потому что не ясно, какая таблица должна использоваться для агрегирования. Например, если поле <Key> связывает две таблицы, не ясно, возвратит ли **Count**(<Key>) количество записей в первой или второй таблице.

Однако при использовании предложения **distinct** агрегирование четко определено и может быть вычислено.

Итак, если ключевое поле используется в функции агрегирования без предложения **distinct**, Qlik Sense вернет количество, которое может не иметь смысла. Решение – использовать предложение **distinct** или копию ключа – то есть копию, которая находится только в одной таблице.

Например, в следующих таблицах ProductID представляет собой ключ между таблицами.

Ключ ProductID между таблицами Products (Продукты) и Details (Сведения)



Count(ProductID) может подсчитываться либо в таблице Products (которая содержит только по одной записи для каждого продукта – здесь ProductID является первичным ключом), либо в таблице Details (которая чаще всего содержит по несколько записей для каждого продукта). Если требуется подсчитать количество разных продуктов, следует использовать Count(distinct ProductID). Если требуется подсчитывать количество строк в конкретной таблице, не следует использовать ключ.

Базовые функции агрегирования

Обзор базовых функций агрегирования

Базовые функции агрегирования – это наиболее часто используемые функции агрегирования.

Каждая функция подробно описана после обзора. Также можно щелкнуть имя функции в синтаксисе, чтобы получить немедленный доступ к подробной информации об этой конкретной функции.

Базовые функции агрегирования в скрипте загрузки данных

FirstSortedValue

Параметр **FirstSortedValue()** возвращает значение из выражения, указанного в элементе **value**. Значение элемента соответствует результату сортировки по аргументу **sort_weight**, например, названию продукта с самой низкой стоимостью единицы. N-ное значение в порядке сортировки

можно указать в **rank**. Если в результате больше одного значения имеют один и тот же элемент **sort_weight** для указанного элемента **rank**, функция возвращает значение NULL. Сортированные значения повторяются в количестве записей, как указано в предложении **group by**, или агрегируются во всем наборе данных, если предложение **group by** не указано.

```
FirstSortedValue ([ distinct ] expression, sort_weight [, rank ])
```

Max

Функция **Max()** находит наибольшее числовое значение агрегированных данных в выражении, как определено предложением**group by**. Если указать **rank** n, можно найти наибольшее n-ное значение.

```
Max ( expression[, rank])
```

Min

Функция **Min()** возвращает наименьшее числовое значение агрегированных данных в выражении, как определено предложением **group by**. Если указать **rank** n, можно найти наименьшее n-ное значение.

```
Min ( expression[, rank])
```

Mode

Функция **Mode()** возвращает наиболее часто встречающееся значение, значение режима, агрегированных данных в выражении, как определено предложением **group by**. Функция **Mode()** может возвращать как числовые, так и текстовые значения.

```
Mode (expression )
```

Only

Only() возвращает значение, если есть только один возможный результат, который может быть получен из агрегированных данных. Если запись содержит только одно значение, возвращается это значение. В противном случае возвращается значение NULL. Используйте предложение **group by**, чтобы оценить множество записей. Функция **Only()** может возвращать числовые и текстовые значения.

```
Only (expression )
```

Sum

Функция **Sum()** вычисляет итоговое значение значений, агрегированных в выражении, как определено предложением **group by**.

```
Sum ([distinct]expression)
```

Базовые функции агрегирования в выражениях диаграмм

Функции агрегирования диаграммы могут использоваться только в полях выражений диаграммы. Выражение аргумента одной функции агрегирования не должно содержать другую функцию агрегирования.

FirstSortedValue

Параметр **FirstSortedValue()** возвращает значение из выражения, указанного в элементе **value**. Значение элемента соответствует результату сортировки по аргументу **sort_weight**, например, названию продукта с самой низкой стоимостью единицы. N-ное значение в порядке сортировки можно указать в **rank**. Если в результате больше одного значения имеют один и тот же элемент **sort_weight** для указанного элемента **rank**, функция возвращает значение NULL.

```
FirstSortedValue — функция диаграммы([{SetExpression}] [DISTINCT] [TOTAL
[<fld {,fld}>]] value, sort_weight [,rank])
```

Max

Max() находит наибольшее значение агрегированных данных. Если указать **rank** n, можно найти наибольшее n-ное значение.

```
Max — функция диаграммыMax() находит наибольшее значение агрегированных
данных. Если указать rank n, можно найти наибольшее n-ное значение. Давайте
также посмотрим на элементы FirstSortedValue и rangemax, которые имеют
одинаковую функциональность в отношении функции Max. Max([{SetExpression}]
[TOTAL [<fld {,fld}>]] expr [,rank]) числовое
значение АргументыАргументОписаниеexprВыражение или поле, содержащее данные
для измерения.rankПо умолчанию значение rank — 1, что соответствует
наибольшему значению. При указании для rank значения 2 будет возвращено
второе наибольшее значение. Если rank имеет значение 3, будет возвращено
третье наибольшее значение, и т. д.SetExpressionПо умолчанию функция
агрегирования агрегирует множество возможных записей, определенных выборкой.
Альтернативный набор записей может быть определен выражением анализа
множества. TOTALЕсли слово TOTAL стоит перед аргументами функции, вычисление
выполняется по всем возможным значениям, указанным в текущих выборках, а не
только в тех, которые относятся к значению текущего измерения, т. е.
измерения диаграммы игнорируются. При использовании выражения TOTAL [<fld
{.fld}>], где префикс TOTAL предшествует списку из одного или нескольких имен
полей, выступающих в качестве подмножества переменных измерения диаграммы,
создается подмножество всех возможных
значений. ДанныеCustomerProductUnitSalesUnitPrice
AstridaAA416AstridaAA1015AstridaBB99BetacabBB510BetacabCC220BetacabDD-
25CanutilityAA815CanutilityCC-19Примеры и результатыПримерыРезультатыMax
(UnitSales)Значение 10, поскольку это наибольшее значение в элементе
UnitSales.Значение порядка вычисляется из числа проданных единиц в элементе
(UnitSales), умноженного на стоимость единицы.Max
(UnitSales*UnitPrice)Значение 150, поскольку это наибольшее значение,
полученное в результате вычисления всех возможных значений элементов
(UnitSales)*(UnitPrice).Max(UnitSales, 2)Значение 9, которое является вторым
наибольшим значением.Max(TOTAL UnitSales)Значение 10, поскольку префикс TOTAL
означает, что обнаружено наибольшее возможное значение без учета измерений
диаграммы. Для диаграммы с элементом Customer в качестве измерения префикс
TOTAL обеспечит возврат максимального значения по всему набору данных вместо
максимального значения UnitSales для каждого клиента.Выполните выборку
Customer B.Max({1} TOTAL UnitSales)Значение 10, независимо от сделанной
```

```
выборки, поскольку выражение Set Analysis {1} определяет порядок записей для оценки в качестве элемента ALL, независимо от выборки.Данные, используемые в примерах:ProductData:LOAD * inline [Customer|Product|UnitSales|UnitPriceAstrida|AA|4|16Astrida|AA|10|15Astrida|BB|9|9Betacab|BB|5|10Betacab|CC|2|20Betacab|DD||25Canutility|AA|8|15Canutility|CC||19] (delimiter is '|'); FirstSortedValue RangeMax ([{SetExpression}] [DISTINCT] [TOTAL [<fld {,fld}>]] expr [,rank])
```

Min

Min() находит наименьшее значение агрегированных данных. Если указать **rank** n, можно найти наименьшее n-ное значение.

```
Min — функция диаграммы([{SetExpression}] [DISTINCT] [TOTAL [<fld {,fld}>]] expr [,rank])
```

Mode

Mode() находит наиболее часто встречающееся значение, значение режима, в агрегированных данных. Функция **Mode()** может обрабатывать как числовые, так и текстовые значения.

```
Mode — функция диаграммы ({[SetExpression] [TOTAL [<fld {,fld}>]]} expr)
```

Only

Only() возвращает значение, если есть только один возможный результат, который может быть получен из агрегированных данных. Например, при поиске одного продукта, где стоимость единицы = 9, будет возвращено значение NULL, если стоимость единицы 9 есть у нескольких продуктов.

```
Only — функция диаграммы([{SetExpression}] [DISTINCT] [TOTAL [<fld {,fld}>]] expr)
```

Sum

Sum() вычисляет итоговое значение агрегированных данных, выданное выражением или полем.

```
Sum — функция диаграммы([{SetExpression}] [DISTINCT] [TOTAL [<fld {,fld}>]] expr])
```

FirstSortedValue

Параметр **FirstSortedValue()** возвращает значение из выражения, указанного в элементе **value**. Значение элемента соответствует результату сортировки по аргументу **sort_weight**, например, названию продукта с самой низкой стоимостью единицы. N-ное значение в порядке сортировки можно указать в **rank**. Если в результате больше одного значения имеют один и тот же элемент **sort_weight** для указанного элемента **rank**, функция возвращает значение NULL. Сортированные значения повторяются в количестве записей, как указано в предложении **group by**, или агрегируются во всем наборе данных, если предложение **group by** не указано.

Синтаксис:

```
FirstSortedValue ([ distinct ] value, sort-weight [, rank ])
```

Возвращаемые типы данных: двойное значение

Аргументы:

Аргументы

Аргумент	Описание
value Expression	С помощью функции можно найти значение выражения **value**, которое соответствует результату сортировки поля **sort_weight**.
sort-weight Expression	Выражение, содержащее данные для сортировки. Обнаружено первое (нижнее) значение элемента **sort_weight**, на основе которого определяется соответствующее значение выражения **value**. Если указать знак минуса перед элементом **sort_weight**, функция вернет последнее (самое высокое) отсортированное значение.
rank Expression	При указании для элемента **rank** значения «n» выше 1 будет получено n-ое отсортированное значение.
distinct	Если слово **DISTINCT** указывается до аргументов функции, все дубликаты, возникшие в результате оценки аргументов функции, будут проигнорированы.

Примеры и результаты:

Добавьте образец скрипта в свое приложение и запустите. Затем добавьте на лист приложения как минимум поля, указанные в столбце с результатами, чтобы увидеть результаты.

Чтобы столбец с результатами выглядел так же, как столбец с результатами ниже, на панели свойств в разделе «Сортировка» переключите параметр с «Авто» на «Пользовательский», а затем отмените выбор числовой сортировки и сортировки в алфавитном порядке.

Примеры написания скриптов

Пример	Результат
Temp: LOAD * inline [Customer|Product|OrderNumber|UnitSales|CustomerID Astrida|AA|1|10|1 Astrida|AA|7|18|1 Astrida|BB|4|9|1 Astrida|CC|6|2|1 Betacab|AA|5|4|2 Betacab|BB|2|5|2 Betacab|DD|12|25|2 Canutility|AA|3|8|3 Canutility|CC|13|19|3 Divadip|AA|9|16|4 Divadip|AA|10|16|4 Divadip|DD|11|10|4] (delimiter is '|'); FirstSortedValue: LOAD Customer,FirstSortedValue(Product, UnitSales) as MyProductWithSmallestOrderByCustomer Resident Temp Group By Customer;	Customer MyProductWithSmallestOrderByCustomer Astrida CC Betacab AA Canutility AA Divadip DD Функция выполняет сортировку значений UnitSales от наименьшего к наибольшему элементу и ищет значение Customer, которому соответствует наименьшее значение UnitSales, то есть наименьший заказ. В связи с этим элемент CC соответствует значению наименьшего заказа (значение параметра UnitSales=2) для клиента Astrida. Элемент AA соответствует наименьшему заказу (4) для клиента Betacab, элемент AA соответствует наименьшему заказу (8) для клиента Canutility, а элемент DD соответствует наименьшему заказу (10) для клиента Divadip..

Пример	Результат

При условии, что таблица **Temp** загружается, как в предыдущем примере:
`LOAD Customer,FirstSortedValue(Product, -UnitSales) as MyProductWithLargestOrderByCustomer Resident Temp Group By Customer;`
	`Customer MyProductWithLargestOrderByCustomer`
	`Astrida AA`
	`Betacab DD`
	`Canutility CC`
	`Divadip -`
	Аргументу sort_weight предшествует знак минуса, поэтому с помощью функции элементы будут отсортированы от наибольших к наименьшим.
	Поскольку элемент AA соответствует наибольшему заказу (значение UnitSales:18) для клиента Astrida, элемент DD соответствует наибольшему заказу (12) для клиента Betacab, и элемент CC соответствует наибольшему заказу (13) для клиента Canutility. Существуют два одинаковых значения для наибольшего заказа (16) клиента Divadip, поэтому будет сформирован нулевой результат.

При условии, что таблица **Temp** загружается, как в предыдущем примере:
`LOAD Customer,FirstSortedValue(distinct Product, -UnitSales) as MyProductWithSmallestOrderByCustomer Resident Temp Group By Customer;`
	`Customer MyProductWithLargestOrderByCustomer`
	`Astrida AA`
	`Betacab DD`
	`Canutility CC`
	`Divadip AA`
	Все действия будут выполняться так же, как и в предыдущем примере, но будет использоваться префикс distinct. При этом результат дубликата для Divadip будет проигнорирован, что позволит вернуть ненулевое значение.

FirstSortedValue – функция диаграммы

Параметр **FirstSortedValue()** возвращает значение из выражения, указанного в элементе **value**. Значение элемента соответствует результату сортировки по аргументу **sort_weight**, например, названию продукта с самой низкой стоимостью единицы. N-ное значение в порядке сортировки можно указать в **rank**. Если в результате больше одного значения имеют один и тот же элемент **sort_weight** для указанного элемента **rank**, функция возвращает значение NULL.

Синтаксис:

```
FirstSortedValue([{SetExpression}] [DISTINCT] [TOTAL [<fld {,fld}>]] value, sort_weight [,rank])
```

Возвращаемые типы данных: двойное значение

Аргументы:

Аргументы

Аргумент	Описание
value	Поле вывода. С помощью функции можно найти значение выражения **value**, которое соответствует результату сортировки поля **sort_weight**.
sort_weight	Поле ввода. Выражение, содержащее данные для сортировки. Обнаружено первое (нижнее) значение элемента **sort_weight**, на основе которого определяется соответствующее значение выражения **value**. Если указать знак минуса перед элементом **sort_weight**, функция вернет последнее (самое высокое) отсортированное значение.
rank	При указании для элемента **rank** значения «n» выше 1 будет получено n-ое отсортированное значение.
SetExpression	По умолчанию функция агрегирования агрегирует множество возможных записей, определенных выборкой. Альтернативный набор записей может быть определен выражением анализа множества.
DISTINCT	Если слово **DISTINCT** указывается до аргументов функции, все дубликаты, возникшие в результате оценки аргументов функции, будут проигнорированы.
TOTAL	Если слово **TOTAL** стоит перед аргументами функции, вычисление выполняется по всем возможным значениям, указанным в текущих выборках, а не только в тех, которые относятся к значению текущего измерения, т. е. измерения диаграммы игнорируются. При использовании выражения **TOTAL [<fld {.fld}>]**, где префикс **TOTAL** предшествует списку из одного или нескольких имен полей, выступающих в качестве подмножества переменных измерения диаграммы, создается подмножество всех возможных значений.

Примеры и результаты:

Данные

Customer	Product	UnitSales	UnitPrice
Astrida	AA	4	16
Astrida	AA	10	15
Astrida	BB	9	9
Betacab	BB	5	10

Customer	Product	UnitSales	UnitPrice
Betacab	CC	2	20
Betacab	DD	-	25
Canutility	AA	8	15
Canutility	CC	-	19

Примеры и результаты

Пример	Результат
`firstsortedvalue (Product, UnitPrice)`	Элемент BB, который является элементом `Product`с наименьшим значением `UnitPrice`(9).
`firstsortedvalue (Product, UnitPrice, 2)`	Элемент BB, который является элементом `Product` со вторым наименьшим значением `UnitPrice`(10).
`firstsortedvalue (Customer, -UnitPrice, 2)`	Элемент Betacab, который является `Customer` с `Product` со вторым наибольшим значением `UnitPrice`(20).
`firstsortedvalue (Customer, UnitPrice, 3)`	Значение NULL, поскольку существуют два значения элемента `Customer` (Astrida и Canutility) с одинаковым значениемrank (третьим наименьшим) `UnitPrice`(15). Используйте префикс distinct, чтобы убедиться, что не возникнет нулевой результат.
`firstsortedvalue (Customer, -UnitPrice*UnitSales, 2)`	Значение Canutility, которое является элементом `Customer` со вторым наибольшим значением порядка продажи `UnitPrice`, умноженным на элемент `UnitSales` (120).

Данные, используемые в примерах:

```
ProductData:
LOAD * inline [
Customer|Product|UnitSales|UnitPrice
Astrida|AA|4|16
Astrida|AA|10|15
Astrida|BB|9|9
Betacab|BB|5|10
Betacab|CC|2|20
Betacab|DD||25
Canutility|AA|8|15
Canutility|CC||19
] (delimiter is '|');
```

Max

Функция **Max()** находит наибольшее числовое значение агрегированных данных в выражении, как определено предложением**group by**. Если указать **rank** n, можно найти наибольшее n-ное значение.

Синтаксис:

```
Max ( expr [, rank])
```

Возвращаемые типы данных: числовое значение

Аргументы:

Аргументы

Аргумент	Описание
expr Expression	Выражение или поле, содержащее данные для измерения.
rank Expression	По умолчанию значение **rank** – 1, что соответствует наибольшему значению. При указании для **rank** значения 2 будет возвращено второе наибольшее значение. Если **rank** имеет значение 3, будет возвращено третье наибольшее значение, и т. д.

Примеры и результаты:

Добавьте образец скрипта в свое приложение и запустите. Затем добавьте на лист приложения как минимум поля, указанные в столбце с результатами, чтобы увидеть результаты.

Чтобы столбец с результатами выглядел так же, как столбец с результатами ниже, на панели свойств в разделе «Сортировка» переключите параметр с «Авто» на «Пользовательский», а затем отмените выбор числовой сортировки и сортировки в алфавитном порядке.

Пример:

```
Temp:
LOAD * inline [
Customer|Product|OrderNumber|UnitSales|CustomerID
Astrida|AA|1|10|1
Astrida|AA|7|18|1
Astrida|BB|4|9|1
Astrida|CC|6|2|1
Betacab|AA|5|4|2
Betacab|BB|2|5|2
Betacab|DD
Canutility|DD|3|8
Canutility|CC
] (delimiter is '|');

Max:
LOAD Customer, Max(UnitSales) as MyMax Resident Temp Group By Customer;
```

Результирующая таблица

Клиент	MyMax
Astrida	18
Betacab	5
Canutility	8

Пример:

При условии, что таблица **Temp** загружается, как в предыдущем примере:

```
LOAD Customer, Max(UnitSales,2) as MyMaxRank2 Resident Temp Group By Customer;
```

Результирующая таблица

Клиент	MyMaxRank2
Astrida	10
Betacab	4
Canutility	-

Max – функция диаграммы

Max() находит наибольшее значение агрегированных данных. Если указать **rank** n, можно найти наибольшее n-ное значение.

*Давайте также посмотрим на элементы **FirstSortedValue** и **rangemax**, которые имеют одинаковую функциональность в отношении функции **Max**.*

Синтаксис:

```
Max([{SetExpression}] [TOTAL [<fld {,fld}>]] expr [,rank])
```

Возвращаемые типы данных: числовое значение

Аргументы:

Аргументы

Аргумент	Описание
expr	Выражение или поле, содержащее данные для измерения.
rank	По умолчанию значение **rank** – 1, что соответствует наибольшему значению. При указании для **rank** значения 2 будет возвращено второе наибольшее значение. Если **rank** имеет значение 3, будет возвращено третье наибольшее значение, и т. д.

Аргумент	Описание
SetExpression	По умолчанию функция агрегирования агрегирует множество возможных записей, определенных выборкой. Альтернативный набор записей может быть определен выражением анализа множества.
TOTAL	Если слово **TOTAL** стоит перед аргументами функции, вычисление выполняется по всем возможным значениям, указанным в текущих выборках, а не только в тех, которые относятся к значению текущего измерения, т. е. измерения диаграммы игнорируются. При использовании выражения **TOTAL [<fld {.fld}>]**, где префикс **TOTAL** предшествует списку из одного или нескольких имен полей, выступающих в качестве подмножества переменных измерения диаграммы, создается подмножество всех возможных значений.

Примеры и результаты:

Данные

Customer	Product	UnitSales	UnitPrice
Astrida	AA	4	16
Astrida	AA	10	15
Astrida	BB	9	9
Betacab	BB	5	10
Betacab	CC	2	20
Betacab	DD	-	25
Canutility	AA	8	15
Canutility	CC	-	19

Примеры и результаты

Примеры	Результаты
`Max(UnitSales)`	Значение 10, поскольку это наибольшее значение в элементе `UnitSales`.

Примеры	Результаты
Значение порядка вычисляется из числа проданных единиц в элементе (UnitSales), умноженного на стоимость единицы. `Max (UnitSales*UnitPrice)`	Значение 150, поскольку это наибольшее значение, полученное в результате вычисления всех возможных значений элементов (UnitSales)*(UnitPrice).
`Max(UnitSales, 2)`	Значение 9, которое является вторым наибольшим значением.
`Max(TOTAL UnitSales)`	Значение 10, поскольку префикс TOTAL означает, что обнаружено наибольшее возможное значение без учета измерений диаграммы. Для диаграммы с элементом Customer в качестве измерения префикс TOTAL обеспечит возврат максимального значения по всему набору данных вместо максимального значения UnitSales для каждого клиента.
Выполните выборку `Customer B`. `Max({1} TOTAL UnitSales)`	Значение 10, независимо от сделанной выборки, поскольку выражение Set Analysis {1} определяет порядок записей для оценки в качестве элемента ALL, независимо от выборки.

Данные, используемые в примерах:

```
ProductData:
LOAD * inline [
Customer|Product|UnitSales|UnitPrice
Astrida|AA|4|16
Astrida|AA|10|15
Astrida|BB|9|9
Betacab|BB|5|10
Betacab|CC|2|20
Betacab|DD||25
Canutility|AA|8|15
Canutility|CC||19
] (delimiter is '|');
```

См. также:

- *FirstSortedValue – функция диаграммы (page 231)*
- *RangeMax (page 757)*

Min

Функция **Min()** возвращает наименьшее числовое значение агрегированных данных в выражении, как определено предложением **group by**. Если указать **rank** n, можно найти наименьшее n-ное значение.

Синтаксис:

```
Min ( expr [, rank])
```

Возвращаемые типы данных: числовое значение

Аргументы:

Аргументы

Аргумент	Описание
expr Expression	Выражение или поле, содержащее данные для измерения.
rank Expression	Значение **rank** по умолчанию равно 1, что соответствует наименьшему значению. При указании для **rank** значения 2 будет возвращено второе наименьшее значение. Если **rank** имеет значение 3, будет возвращено третье наименьшее значение и т. д.

Примеры и результаты:

Добавьте образец скрипта в свое приложение и запустите. Затем добавьте на лист приложения как минимум поля, указанные в столбце с результатами, чтобы увидеть результаты.

Чтобы столбец с результатами выглядел так же, как столбец с результатами ниже, на панели свойств в разделе «Сортировка» переключите параметр с «Авто» на «Пользовательский», а затем отмените выбор числовой сортировки и сортировки в алфавитном порядке.

Пример:

```
Temp:
LOAD * inline [
Customer|Product|OrderNumber|UnitSales|CustomerID
Astrida|AA|1|10|1
Astrida|AA|7|18|1
Astrida|BB|4|9|1
Astrida|CC|6|2|1
Betacab|AA|5|4|2
Betacab|BB|2|5|2
Betacab|DD
Canutility|DD|3|8
Canutility|CC
] (delimiter is '|');
Min:
LOAD Customer, Min(UnitSales) as MyMin Resident Temp Group By Customer;
```

Результирующая таблица

Клиент	MyMin
Astrida	2

Клиент	MyMin
Betacab	4
Canutility	8

Пример:

При условии, что таблица **Temp** загружается, как в предыдущем примере:

```
LOAD Customer, Min(UnitSales,2) as MyMinRank2 Resident Temp Group By Customer;
```

Результирующая таблица

Клиент	MyMinRank2
Astrida	9
Betacab	5
Canutility	-

Min – функция диаграммы

Min() находит наименьшее значение агрегированных данных. Если указать **rank** n, можно найти наименьшее n-ное значение.

*Давайте также посмотрим на элементы **FirstSortedValue** и **rangemin**, которые имеют одинаковую функциональность в отношении функции **Min**.*

Синтаксис:

```
Min({[SetExpression] [TOTAL [<fld {,fld}>]]} expr [,rank])
```

Возвращаемые типы данных: числовое значение

Аргументы:

Аргументы

Аргумент	Описание
expr	Выражение или поле, содержащее данные для измерения.
rank	Значение **rank** по умолчанию равно 1, что соответствует наименьшему значению. При указании для **rank** значения 2 будет возвращено второе наименьшее значение. Если **rank** имеет значение 3, будет возвращено третье наименьшее значение и т. д.
SetExpression	По умолчанию функция агрегирования агрегирует множество возможных записей, определенных выборкой. Альтернативный набор записей может быть определен выражением анализа множества.

Аргумент	Описание
TOTAL	Если слово **TOTAL** стоит перед аргументами функции, вычисление выполняется по всем возможным значениям, указанным в текущих выборках, а не только в тех, которые относятся к значению текущего измерения, т. е. измерения диаграммы игнорируются. При использовании выражения **TOTAL [<fld {.fld}>]**, где префикс **TOTAL** предшествует списку из одного или нескольких имен полей, выступающих в качестве подмножества переменных измерения диаграммы, создается подмножество всех возможных значений.

Примеры и результаты:

Данные

Customer	Product	UnitSales	UnitPrice
Astrida	AA	4	16
Astrida	AA	10	15
Astrida	BB	9	9
Betacab	BB	5	10
Betacab	CC	2	20
Betacab	DD	-	25
Canutility	AA	8	15
Canutility	CC	-	19

Функция Min() должна возвращать значение, не являющееся NULL, из диапазона значений, обеспеченных выражением, если таковое имеется. Таким образом, поскольку в данных имеются значения NULL, функция возвращает первое значение, не являющееся NULL, оцененное из выражения.

Примеры и результаты

Примеры	Результаты
`Min(UnitSales)`	Значение 2, поскольку это наименьшее значение, не являющееся NULL, в элементе `UnitSales`.

Примеры	Результаты
Значение порядка вычисляется из числа проданных единиц в элементе (UnitSales), умноженного на стоимость единицы. `Min (UnitSales*UnitPrice)`	Значение 40, поскольку это наименьшее значение, не являющееся NULL, полученное в результате вычисления всех возможных значений элементов (UnitSales)*(UnitPrice).
`Min(UnitSales, 2)`	Значение 4, которое является вторым наименьшим значением (после значений NULL).
`Min(TOTAL UnitSales)`	Значение 2, поскольку классификатор TOTAL означает, что обнаружено наименьшее возможное значение без учета измерений диаграммы. Для диаграммы с элементом Customer в качестве измерения классификатор TOTAL обеспечит возврат минимального значения по всему набору данных вместо минимального значения UnitSales для каждого клиента.
Выполните выборку Customer B. `Min({1} TOTAL UnitSales)`	Значение 2, независимое от выборки Customer B. Выражение Set Analysis {1} определяет порядок записей для оценки в качестве элемента ALL, независимо от выборки.

Данные, используемые в примерах:

```
ProductData:
LOAD * inline [
Customer|Product|UnitSales|UnitPrice
Astrida|AA|4|16
Astrida|AA|10|15
Astrida|BB|9|9
Betacab|BB|5|10
Betacab|CC|2|20
Betacab|DD||25
Canutility|AA|8|15
Canutility|CC||19
] (delimiter is '|');
```

См. также:

- *FirstSortedValue – функция диаграммы (page 231)*
- *RangeMin (page 761)*

Mode

Функция **Mode()** возвращает наиболее часто встречающееся значение, значение режима, агрегированных данных в выражении, как определено предложением **group by**. Функция **Mode()** может возвращать как числовые, так и текстовые значения.

Синтаксис:

```
Mode ( expr)
```

Возвращаемые типы данных: двойное значение

Аргументы

Аргумент	Описание
expr Expression	Выражение или поле, содержащее данные для измерения.

Ограничения:

Если одинаково часто встречаются несколько значений, возвращается значение NULL.

Примеры и результаты:

Добавьте образец скрипта в свое приложение и запустите. Затем добавьте на лист приложения как минимум поля, указанные в столбце с результатами, чтобы увидеть результаты.

Чтобы столбец с результатами выглядел так же, как столбец с результатами ниже, на панели свойств в разделе «Сортировка» переключите параметр с «Авто» на «Пользовательский», а затем отмените выбор числовой сортировки и сортировки в алфавитном порядке.

Примеры написания скриптов

Пример	Результат
Temp: LOAD * inline [Customer|Product|OrderNumber|UnitSales|CustomerID Astrida|AA|1|10|1 Astrida|AA|7|18|1 Astrida|BB|4|9|1 Astrida|CC|6|2|1 Betacab|AA|5|4|2 Betacab|BB|2|5|2 Betacab|DD Canutility|DD|3|8 Canutility|CC] (delimiter is '|'); Mode: LOAD Customer, Mode(Product) as MyMostOftenSoldProduct Resident Temp Group By Customer;	MyMostOftenSoldProduct AA поскольку AA – это единственный продукт, проданный несколько раз.

Mode – функция диаграммы

Mode() находит наиболее часто встречающееся значение, значение режима, в агрегированных данных. Функция **Mode()** может обрабатывать как числовые, так и текстовые значения.

Синтаксис:

```
Mode({[SetExpression] [TOTAL [<fld {,fld}>]]} expr)
```

Возвращаемые типы данных: двойное значение

Аргументы:

Аргументы

Аргумент	Описание
expr	Выражение или поле, содержащее данные для измерения.
SetExpression	По умолчанию функция агрегирования агрегирует множество возможных записей, определенных выборкой. Альтернативный набор записей может быть определен выражением анализа множества.
TOTAL	Если слово **TOTAL** стоит перед аргументами функции, вычисление выполняется по всем возможным значениям, указанным в текущих выборках, а не только в тех, которые относятся к значению текущего измерения, т. е. измерения диаграммы игнорируются. При использовании выражения **TOTAL [<fld {,fld}>]**, где префикс **TOTAL** предшествует списку из одного или нескольких имен полей, выступающих в качестве подмножества переменных измерения диаграммы, создается подмножество всех возможных значений.

Примеры и результаты:

Данные

Customer	Product	UnitSales	UnitPrice
Astrida	AA	4	16
Astrida	AA	10	15
Astrida	BB	9	9
Betacab	BB	5	10
Betacab	CC	2	20
Betacab	DD	-	25
Canutility	AA	8	15
Canutility	CC	-	19

Примеры и результаты

Примеры	Результаты
`Mode(UnitPrice)` Выполните выборку `Customer A`.	Значение 15, поскольку это наиболее часто встречающееся значение в элементе `UnitSales`. Возвращает NULL (-). Одно значение встречается не чаще, чем другое.
`Mode(Product)` Выполните выборку `Customer A`.	Значение AA, поскольку это наиболее часто встречающееся значение в элементе `Product`. Возвращает NULL (-). Одно значение встречается не чаще, чем другое.
`Mode` `(TOTAL UnitPrice)`	Значение 15, поскольку классификатор TOTAL означает, что наиболее часто встречающимся значением все еще является 15, без учета измерений диаграммы.

Выполните выборку
`Customer B`.
 	
`Mode({1}`
`TOTAL UnitPrice)`
	Значение 15, независимо от сделанной выборки, поскольку выражение Set Analysis {1} определяет порядок записей для оценки в качестве элемента ALL, независимо от выборки.

Данные, используемые в примерах:

```
ProductData:
LOAD * inline [
Customer|Product|UnitSales|UnitPrice
Astrida|AA|4|16
Astrida|AA|10|15
Astrida|BB|9|9
Betacab|BB|5|10
Betacab|CC|2|20
Betacab|DD||25
Canutility|AA|8|15
Canutility|CC||19
] (delimiter is '|');
```

См. также:

- *Avg – функция диаграммы (page 290)*
- *Median – функция диаграммы (page 330)*

Only

Only() возвращает значение, если есть только один возможный результат, который может быть получен из агрегированных данных. Если запись содержит только одно значение, возвращается это значение. В противном случае возвращается значение NULL. Используйте предложение **group by**, чтобы оценить множество записей. Функция **Only()** может возвращать числовые и текстовые значения.

Синтаксис:

```
Only ( expr )
```

Возвращаемые типы данных: двойное значение

Аргументы

Аргумент	Описание
expr Expression	Выражение или поле, содержащее данные для измерения.

Примеры и результаты:

Добавьте образец скрипта в свое приложение и запустите. Затем добавьте на лист приложения как минимум поля, указанные в столбце с результатами, чтобы увидеть результаты.

Чтобы столбец с результатами выглядел так же, как столбец с результатами ниже, на панели свойств в разделе «Сортировка» переключите параметр с «Авто» на «Пользовательский», а затем отмените выбор числовой сортировки и сортировки в алфавитном порядке.

```
Temp:
LOAD * inline [
Customer|Product|OrderNumber|UnitSales|CustomerID
Astrida|AA|1|10|1
Astrida|AA|7|18|1
Astrida|BB|4|9|1
Astrida|CC|6|2|1
Betacab|AA|5|4|2
Betacab|BB|2|5|2
Betacab|DD
Canutility|DD|3|8
Canutility|CC
] (delimiter is '|');
Only:
LOAD Customer, Only(CustomerID) as MyUniqIDCheck Resident Temp Group By Customer;
```

Результирующая таблица

Клиент	MyUniqIDCheck
Astrida	1 поскольку только у клиента Astrida записи заполнены и включают элемент CustomerID.

Only – функция диаграммы

Only() возвращает значение, если есть только один возможный результат, который может быть получен из агрегированных данных. Например, при поиске одного продукта, где стоимость единицы = 9, будет возвращено значение NULL, если стоимость единицы 9 есть у нескольких продуктов.

Синтаксис:

```
Only([{SetExpression}] [TOTAL [<fld {,fld}>]] expr)
```

Возвращаемые типы данных: двойное значение

Аргументы:

Аргументы

Аргумент	Описание
expr	Выражение или поле, содержащее данные для измерения.
SetExpression	По умолчанию функция агрегирования агрегирует множество возможных записей, определенных выборкой. Альтернативный набор записей может быть определен выражением анализа множества.
TOTAL	Если слово **TOTAL** стоит перед аргументами функции, вычисление выполняется по всем возможным значениям, указанным в текущих выборках, а не только в тех, которые относятся к значению текущего измерения, т. е. измерения диаграммы игнорируются. При использовании выражения **TOTAL [<fld {.fld}>]**, где префикс **TOTAL** предшествует списку из одного или нескольких имен полей, выступающих в качестве подмножества переменных измерения диаграммы, создается подмножество всех возможных значений.

Используйте функцию Only(), если необходимо получить значение NULL в случае нескольких возможных значений в данных образца.

Примеры и результаты:

Данные

Customer	Product	UnitSales	UnitPrice
Astrida	AA	4	16
Astrida	AA	10	15
Astrida	BB	9	9
Betacab	BB	5	10
Betacab	CC	2	20
Betacab	DD	-	25
Canutility	AA	8	15
Canutility	CC	-	19

Примеры и результаты

Примеры	Результаты
`Only ({<UnitPrice={9}>} Product)`	Значение BB, поскольку это единственный элемент `Product`, у которого элемент `UnitPrice` равен 9.
`Only({<Product={DD}>} Customer)`	Значение Betacab, поскольку единственный элемент `Customer`, продающий `Product`, называется «DD».
`Only ({<UnitPrice={20}>} UnitSales)`	Число элементов `UnitSales`, где элемент `UnitPrice`, равный 20, составляет 2, поскольку есть только одно значение элемента `UnitSales`, где `UnitPrice` = 20.
`Only ({<UnitPrice={15}>} UnitSales)`	Значение NULL, поскольку существуют два значения элемента `UnitSales`, где `UnitPrice` = 15.

Данные, используемые в примерах:

```
ProductData:
LOAD * inline [
Customer|Product|UnitSales|UnitPrice
Astrida|AA|4|16
Astrida|AA|10|15
Astrida|BB|9|9
Betacab|BB|5|10
Betacab|CC|2|20
Betacab|DD||25
Canutility|AA|8|15
Canutility|CC||19
] (delimiter is '|');
```

Sum

Функция **Sum()** вычисляет итоговое значение значений, агрегированных в выражении, как определено предложением **group by**.

Синтаксис:

```
sum ( [ distinct] expr)
```

Возвращаемые типы данных: числовое значение

Аргументы:

Аргументы

Аргумент	Описание
distinct	Если слово **distinct** указано перед выражением, все дубликаты будут проигнорированы.
expr Expression	Выражение или поле, содержащее данные для измерения.

Примеры и результаты:

Добавьте образец скрипта в свое приложение и запустите. Затем добавьте на лист приложения как минимум поля, указанные в столбце с результатами, чтобы увидеть результаты.

Чтобы столбец с результатами выглядел так же, как столбец с результатами ниже, на панели свойств в разделе «Сортировка» переключите параметр с «Авто» на «Пользовательский», а затем отмените выбор числовой сортировки и сортировки в алфавитном порядке.

```
Temp:
LOAD * inline [
Customer|Product|OrderNumber|UnitSales|CustomerID
Astrida|AA|1|10|1
Astrida|AA|7|18|1
Astrida|BB|4|9|1
Astrida|CC|6|2|1
Betacab|AA|5|4|2
Betacab|BB|2|5|2
Betacab|DD
Canutility|DD|3|8
Canutility|CC
] (delimiter is '|');
Sum:
LOAD Customer, Sum(UnitSales) as MySum Resident Temp Group By Customer;
```

Результирующая таблица

Клиент	MySum
Astrida	39
Betacab	9
Canutility	8

Sum – функция диаграммы

Sum() вычисляет итоговое значение агрегированных данных, выданное выражением или полем.

Синтаксис:

```
Sum([{SetExpression}] [DISTINCT] [TOTAL [<fld {,fld}>]] expr)
```

Возвращаемые типы данных: числовое значение

Аргументы:

Аргументы

Аргумент	Описание
expr	Выражение или поле, содержащее данные для измерения.

Аргумент	Описание
SetExpression	По умолчанию функция агрегирования агрегирует множество возможных записей, определенных выборкой. Альтернативный набор записей может быть определен выражением анализа множества.
DISTINCT	Если слово **DISTINCT** указывается до аргументов функции, все дубликаты, возникшие в результате оценки аргументов функции, будут проигнорированы. *Несмотря на то, что классификатор DISTINCT поддерживается, используйте его чрезвычайно осторожно, поскольку его использование может ввести в заблуждение – читатель может подумать, что показано итоговое значение, в то время как некоторые данные опущены.*
TOTAL	Если слово **TOTAL** стоит перед аргументами функции, вычисление выполняется по всем возможным значениям, указанным в текущих выборках, а не только в тех, которые относятся к значению текущего измерения, т. е. измерения диаграммы игнорируются. При использовании выражения **TOTAL [<fld {.fld}>]**, где префикс **TOTAL** предшествует списку из одного или нескольких имен полей, выступающих в качестве подмножества переменных измерения диаграммы, создается подмножество всех возможных значений.

Примеры и результаты:

Данные

Customer	Product	UnitSales	UnitPrice
Astrida	AA	4	16
Astrida	AA	10	15
Astrida	BB	9	9
Betacab	BB	5	10
Betacab	CC	2	20
Betacab	DD	-	25
Canutility	AA	8	15
Canutility	CC	-	19

Примеры и результаты

Примеры	Результаты
`Sum(UnitSales)`	38. Итого значений в элементе `UnitSales`.
`Sum(UnitSales*UnitPrice)`	505. Итого элемента `UnitPrice`, умноженное на агрегированный элемент `UnitSales`.
`Sum (TOTAL UnitSales*UnitPrice)`	Значение 505 для всех строк в таблице, а также итоговое значение, поскольку классификатор TOTAL означает, что сумма по-прежнему равна 505, без учета измерений диаграммы.
Выполните выборку `Customer` в. `Sum({1} TOTAL UnitSales*UnitPrice)`	Значение 505, независимо от сделанной выборки, поскольку выражение Set Analysis {1} определяет порядок записей для оценки в качестве элемента ALL, независимо от выборки.

Данные, используемые в примерах:

```
ProductData:
LOAD * inline [
Customer|Product|UnitSales|UnitPrice
Astrida|AA|4|16
Astrida|AA|10|15
Astrida|BB|9|9
Betacab|BB|5|10
Betacab|CC|2|20
Betacab|DD||25
Canutility|AA|8|15
Canutility|CC||19
] (delimiter is '|');
```

Функции агрегирования счетчика

Функции агрегирования счетчика возвращают различные типы счетчиков выражения для ряда записей в скрипте загрузки данных или ряда значений в измерении диаграммы.

Каждая функция подробно описана после обзора. Также можно щелкнуть имя функции в синтаксисе, чтобы получить немедленный доступ к подробной информации об этой конкретной функции.

Функции агрегирования счетчика в скрипте загрузки данных

Count

Функция **Count()** возвращает число значений, агрегированных в выражении, как определено предложением **group by**.

```
Count ([distinct ] expression | * )
```

MissingCount

Функция **MissingCount()** возвращает число отсутствующих значений, агрегированных в выражении, как определено предложением **group by**.

```
MissingCount ([ distinct ] expression)
```

NullCount

Функция **NullCount()** возвращает число значений NULL, агрегированных в выражении, как определено предложением **group by**.

```
NullCount ([ distinct ] expression)
```

NumericCount

Функция **NumericCount()** возвращает число числовых значений, найденных в выражении, как определено предложением **group by**.

```
NumericCount ([ distinct ] expression)
```

TextCount

Функция **TextCount()** возвращает число нечисловых значений поля, агрегированных в выражении, как определено предложением **group by**.

```
TextCount ([ distinct ] expression)
```

Функции агрегирования счетчика в выражениях диаграмм

Следующие функции агрегирования счетчика можно использовать в диаграммах:

Count

Count() используется для агрегирования текстовых и числовых значений в каждом измерении диаграммы.

```
Count — функция диаграммы({[SetExpression] [DISTINCT] [TOTAL [<fld {,fld}>]]}
expr)
```

MissingCount

MissingCount() используется для агрегирования отсутствующих значений в каждом измерении диаграммы. Отсутствующие значения – это все нечисловые значения.

```
MissingCount — функция диаграммы({[SetExpression] [DISTINCT] [TOTAL [<fld
{,fld}>]] expr)
```

NullCount

NullCount() используется для агрегирования значений NULL в каждом измерении диаграммы.

```
NullCount — функция диаграммы({[SetExpression][DISTINCT] [TOTAL [<fld
{,fld}>]]} expr)
```

NumericCount

NumericCount() используется для агрегирования числовых значений в каждом измерении диаграммы.

```
NumericCount – функция диаграммы({[SetExpression] [DISTINCT] [TOTAL [<fld
{,fld}>]]} expr)
```

TextCount

TextCount() используется для агрегирования нечисловых значений поля в каждом измерении диаграммы.

```
TextCount – функция диаграммы({[SetExpression] [DISTINCT] [TOTAL [<fld
{,fld}>]]} expr)
```

Count

Функция **Count()** возвращает число значений, агрегированных в выражении, как определено предложением **group by**.

Синтаксис:

```
Count( [distinct ] expr)
```

Возвращаемые типы данных: целое

Аргументы:

Аргументы

Аргумент	Описание
expr	Выражение или поле, содержащее данные для измерения.
distinct	Если слово **distinct** указано перед выражением, все дубликаты будут проигнорированы.

Примеры и результаты:

Добавьте образец скрипта в свое приложение и запустите. Затем добавьте на лист приложения как минимум поля, указанные в столбце с результатами, чтобы увидеть результаты.

Чтобы столбец с результатами выглядел так же, как столбец с результатами ниже, на панели свойств в разделе «Сортировка» переключите параметр с «Авто» на «Пользовательский», а затем отмените выбор числовой сортировки и сортировки в алфавитном порядке.

Примеры написания скриптов

Пример	Результат
Temp: LOAD * inline [Customer|Product|OrderNumber|UnitSales|UnitPrice Astrida|AA|1|4|16 Astrida|AA|7|10|15 Astrida|BB|4|9|9 Betacab|CC|6|5|10 Betacab|AA|5|2|20 Betacab|BB|1|25| 25 Canutility|AA|3|8|15 Canutility|CC|||19 Divadip|CC|2|4|16 Divadip|DD|3|1|25] (delimiter is '|'); Count1: LOAD Customer,Count(OrderNumber) as OrdersByCustomer Resident Temp Group By Customer;	Customer OrdersByCustomer Astrida 3 Betacab 3 Canutility 2 Divadip 2 При условии, что измерение Customer включено в таблицу на листе, в противном случае результатом для OrdersByCustomer будет 3, 2.
При условии, что таблица **Temp** загружается, как в предыдущем примере: LOAD Count(OrderNumber) as TotalOrderNumber Resident Temp;	TotalOrderNumber 10
При условии, что таблица **Temp** загружается, как в первом примере: LOAD Count(distinct OrderNumber) as TotalOrderNumber Resident Temp;	TotalOrderNumber 8 Поскольку имеется два значения OrderNumber с одинаковым значением, а также 1 и «null».

Count – функция диаграммы

Count() используется для агрегирования текстовых и числовых значений в каждом измерении диаграммы.

Синтаксис:

```
Count({[SetExpression] [DISTINCT] [TOTAL [<fld {,fld}>]]} expr)
```

Возвращаемые типы данных: целое

Аргументы:

Аргументы

Аргумент	Описание
expr	Выражение или поле, содержащее данные для измерения.

Аргумент	Описание
SetExpression	По умолчанию функция агрегирования агрегирует множество возможных записей, определенных выборкой. Альтернативный набор записей может быть определен выражением анализа множества.
DISTINCT	Если слово **DISTINCT** указывается до аргументов функции, все дубликаты, возникшие в результате оценки аргументов функции, будут проигнорированы.
TOTAL	Если слово **TOTAL** стоит перед аргументами функции, вычисление выполняется по всем возможным значениям, указанным в текущих выборках, а не только в тех, которые относятся к значению текущего измерения, т. е. измерения диаграммы игнорируются. При использовании выражения **TOTAL [<fld {.fld}>]**, где префикс **TOTAL** предшествует списку из одного или нескольких имен полей, выступающих в качестве подмножества переменных измерения диаграммы, создается подмножество всех возможных значений.

Примеры и результаты:

Data

Customer	Product	OrderNumber	UnitSales	Unit Price
Astrida	AA	1	4	16
Astrida	AA	7	10	15
Astrida	BB	4	9	9
Betacab	BB	6	5	10
Betacab	CC	5	2	20
Betacab	DD	1	25	25
Canutility	AA	3	8	15
Canutility	CC			19
Divadip	AA	2	4	16
Divadip	DD	3		25

В следующих примерах считается, что все клиенты выбраны, если не указано иначе.

Примеры и результаты

Пример	Результат
`Count(OrderNumber)`	Значение 10, поскольку существует 10 полей, которые могут иметь значение для элемента OrderNumber, а также учитываются все записи, даже пустые. *«0» считается значением, а не пустой ячейкой. Тем не менее, если мера агрегирует значение для измерения до 0, это измерение не будет включено в диаграммы.*
`Count(Customer)`	Значение 10, поскольку элемент Count оценивает число вхождений во всех полях.
`Count(DISTINCT [Customer])`	Значение 4, поскольку при использовании классификатора Distinct элемент Count оценивает только уникальные вхождения.
При условии выбора клиента Canutility `Count (OrderNumber)/Count ({1} TOTAL OrderNumber)`	Значение 0,2, поскольку выражение возвращает число заказов выбранного клиента в виде процентного соотношения заказов всех клиентов. В этом случае 2/10.
При условии выбора клиентов Astrida и Canutility `Count(TOTAL <Product> OrderNumber)`	Значение 5, поскольку это число заказов, размещенных для продуктов только выбранных клиентов, пустые ячейки учитываются.

Данные, используемые в примерах:

```
Temp:
LOAD * inline [
Customer|Product|OrderNumber|UnitSales|UnitPrice
Astrida|AA|1|4|16
Astrida|AA|7|10|15
Astrida|BB|4|9|9
Betacab|CC|6|5|10
Betacab|AA|5|2|20
Betacab|BB|1|25| 25
Canutility|AA|3|8|15
Canutility|CC|||19
Divadip|CC|2|4|16
Divadip|DD|3|1|25
] (delimiter is '|');
```

MissingCount

Функция **MissingCount()** возвращает число отсутствующих значений, агрегированных в выражении, как определено предложением **group by**.

Синтаксис:

```
MissingCount ( [ distinct ] expr)
```

Возвращаемые типы данных: целое

Аргументы:

Аргументы

Аргумент	Описание
expr Expression	Выражение или поле, содержащее данные для измерения.
distinct	Если слово **distinct** указано перед выражением, все дубликаты будут проигнорированы.

Примеры и результаты:

Добавьте образец скрипта в свое приложение и запустите. Затем добавьте на лист приложения как минимум поля, указанные в столбце с результатами, чтобы увидеть результаты.

Чтобы столбец с результатами выглядел так же, как столбец с результатами ниже, на панели свойств в разделе «Сортировка» переключите параметр с «Авто» на «Пользовательский», а затем отмените выбор числовой сортировки и сортировки в алфавитном порядке.

Примеры написания скриптов

Пример	Результат
Temp: LOAD * inline [Customer|Product|OrderNumber|UnitSales|UnitPrice Astrida|AA|1|4|16 Astrida|AA|7|10|15 Astrida|BB|4|9|9 Betacab|CC|6|5|10 Betacab|AA|5|2|20 Betacab|BB||| 25 Canutility|AA|||15 Canutility|CC| ||19 Divadip|CC|2|4|16 Divadip|DD|3|1|25] (delimiter is '|'); MissCount1: LOAD Customer,MissingCount(OrderNumber) as MissingOrdersByCustomer Resident Temp Group By Customer; Load MissingCount(OrderNumber) as TotalMissingCount Resident Temp;	Customer MissingOrdersByCustomer Astrida 0 Betacab 1 Canutility 2 Divadip 0 Второй оператор дает следующее: TotalMissingCount 3 в таблице с этим измерением.
При условии, что таблица **Temp** загружается, как в предыдущем примере: LOAD MissingCount(distinct OrderNumber) as TotalMissingCountDistinct Resident Temp;	TotalMissingCountDistinct 1 Поскольку одним отсутствующим значением является только один элемент OrderNumber.

MissingCount – функция диаграммы

MissingCount() используется для агрегирования отсутствующих значений в каждом измерении диаграммы. Отсутствующие значения – это все нечисловые значения.

Синтаксис:

```
MissingCount({[SetExpression] [DISTINCT] [TOTAL [<fld {,fld}>]]} expr)
```

Возвращаемые типы данных: целое

Аргументы:

Аргументы

Аргумент	Описание
expr	Выражение или поле, содержащее данные для измерения.
SetExpression	По умолчанию функция агрегирования агрегирует множество возможных записей, определенных выборкой. Альтернативный набор записей может быть определен выражением анализа множества.

Аргумент	Описание
DISTINCT	Если слово **DISTINCT** указывается до аргументов функции, все дубликаты, возникшие в результате оценки аргументов функции, будут проигнорированы.
TOTAL	Если слово **TOTAL** стоит перед аргументами функции, вычисление выполняется по всем возможным значениям, указанным в текущих выборках, а не только в тех, которые относятся к значению текущего измерения, т. е. измерения диаграммы игнорируются. При использовании выражения **TOTAL [<fld {.fld}>]**, где префикс **TOTAL** предшествует списку из одного или нескольких имен полей, выступающих в качестве подмножества переменных измерения диаграммы, создается подмножество всех возможных значений.

Примеры и результаты:

Data

Customer	Product	OrderNumber	UnitSales	Unit Price
Astrida	AA	1	4	16
Astrida	AA	7	10	15
Astrida	BB	4	9	9
Betacab	BB	6	5	10
Betacab	CC	5	2	20
Betacab	DD			25
Canutility	AA			15
Canutility	CC			19
Divadip	AA	2	4	16
Divadip	DD	3		25

Примеры и результаты

Пример	Результат
`MissingCount([OrderNumber])`	Значение 3, поскольку 3 из 10 полей OrderNumber являются пустыми *«0» считается значением, а не пустой ячейкой. Тем не менее, если мера агрегирует значение для измерения до 0, это измерение не будет включено в диаграммы.*
`MissingCount ([OrderNumber])/MissingCount ({1} Total [OrderNumber])`	Выражение возвращает число невыполненных заказов выбранного клиента в виде доли невыполненных заказов всех клиентов. Всего 3 отсутствующих значения для поля OrderNumber для всех клиентов. Таким образом, для каждого элемента Customer, имеющего отсутствующее значение для элемента Product, результатом будет 1/3.

Данные, используемые в примере:

```
Temp:
LOAD * inline [
Customer|Product|OrderNumber|UnitSales|UnitPrice
Astrida|AA|1|4|16
Astrida|AA|7|10|15
Astrida|BB|4|9|9
Betacab|CC|6|5|10
Betacab|AA|5|2|20
Betacab|BB||| 25
Canutility|AA|||15
Canutility|CC| ||19
Divadip|CC|2|4|16
Divadip|DD|3|1|25
] (delimiter is '|');
```

NullCount

Функция **NullCount()** возвращает число значений NULL, агрегированных в выражении, как определено предложением **group by**.

Синтаксис:

```
NullCount ( [ distinct ] expr)
```

Возвращаемые типы данных: целое

Аргументы:

Аргументы

Аргумент	Описание
expr Expression	Выражение или поле, содержащее данные для измерения.
distinct	Если слово **distinct** указано перед выражением, все дубликаты будут проигнорированы.

Примеры и результаты:

Добавьте образец скрипта в свое приложение и запустите. Затем добавьте на лист приложения как минимум поля, указанные в столбце с результатами, чтобы увидеть результаты.

Чтобы столбец с результатами выглядел так же, как столбец с результатами ниже, на панели свойств в разделе «Сортировка» переключите параметр с «Авто» на «Пользовательский», а затем отмените выбор числовой сортировки и сортировки в алфавитном порядке.

Примеры написания скриптов

Пример	Результат

`Set NULLINTERPRET = NULL;`
`Temp:`
`LOAD * inline [`
`Customer|Product|OrderNumber|UnitSales|CustomerID`
`Astrida|AA|1|10|1`
`Astrida|AA|7|18|1`
`Astrida|BB|4|9|1`
`Astrida|CC|6|2|1`
`Betacab|AA|5|4|2`
`Betacab|BB|2|5|2`
`Betacab|DD||`
`Canutility|AA|3|8|`
`Canutility|CC|NULL||`
`] (delimiter is '|');`
`Set NULLINTERPRET=;`
`NullCount1:`
`LOAD Customer,NullCount(OrderNumber) as NullOrdersByCustomer Resident Temp Group By Customer;`
 	
`LOAD NullCount(OrderNumber) as TotalNullCount Resident Temp;`
	`Customer NullOrdersByCustomer`
	`Astrida 0`
	`Betacab 0`
	`Canutility 1`
	
	Второй оператор дает следующее:
	`TotalNullCount`
	`1`
	в таблице с этим измерением, поскольку единственная запись содержит нулевое значение.

NullCount – функция диаграммы

NullCount() используется для агрегирования значений NULL в каждом измерении диаграммы.

Синтаксис:

```
NullCount({[SetExpression][DISTINCT] [TOTAL [<fld {,fld}>]]} expr)
```

Возвращаемые типы данных: целое

Аргументы:

Аргументы

Аргумент	Описание
expr	Выражение или поле, содержащее данные для измерения.
set_ expression	По умолчанию функция агрегирования агрегирует множество возможных записей, определенных выборкой. Альтернативный набор записей может быть определен выражением анализа множества.
DISTINCT	Если слово **DISTINCT** указывается до аргументов функции, все дубликаты, возникшие в результате оценки аргументов функции, будут проигнорированы.
TOTAL	Если слово **TOTAL** стоит перед аргументами функции, вычисление выполняется по всем возможным значениям, указанным в текущих выборках, а не только в тех, которые относятся к значению текущего измерения, т. е. измерения диаграммы игнорируются. При использовании выражения **TOTAL [<fld {.fld}>]**, где префикс **TOTAL** предшествует списку из одного или нескольких имен полей, выступающих в качестве подмножества переменных измерения диаграммы, создается подмножество всех возможных значений.

Примеры и результаты:

Примеры и результаты

Пример	Результат
NullCount ([OrderNumber])	Значение 1, поскольку введено нулевое значение с помощью элемента NullInterpret во встроенном операторе **LOAD**.

Данные, используемые в примере:

```
Set NULLINTERPRET = NULL;
Temp:
LOAD * inline [
Customer|Product|OrderNumber|UnitSales|CustomerID
Astrida|AA|1|10|1
Astrida|AA|7|18|1
Astrida|BB|4|9|1
Astrida|CC|6|2|1
Betacab|AA|5|4|2
Betacab|BB|2|5|2
Betacab|DD|||
Canutility|AA|3|8|
Canutility|CC|NULL||
```

```
] (delimiter is '|');
Set NULLINTERPRET=;
```

NumericCount

Функция **NumericCount()** возвращает число числовых значений, найденных в выражении, как определено предложением **group by**.

Синтаксис:

```
NumericCount ( [ distinct ] expr)
```

Возвращаемые типы данных: целое

Аргументы:

Аргументы

Аргумент	Описание
expr Expression	Выражение или поле, содержащее данные для измерения.
distinct	Если слово **distinct** указано перед выражением, все дубликаты будут проигнорированы.

Примеры и результаты:

Добавьте образец скрипта в свое приложение и запустите. Затем добавьте на лист приложения как минимум поля, указанные в столбце с результатами, чтобы увидеть результаты.

Чтобы столбец с результатами выглядел так же, как столбец с результатами ниже, на панели свойств в разделе «Сортировка» переключите параметр с «Авто» на «Пользовательский», а затем отмените выбор числовой сортировки и сортировки в алфавитном порядке.

Пример написания скрипта

Пример	Результат
`LOAD NumericCount(OrderNumber) as TotalNumericCount Resident Temp;`	Второй оператор дает следующее: TotalNumericCount 7 в таблице с этим измерением.
При условии, что таблица **Temp** загружается, как в предыдущем примере: `LOAD NumericCount(distinct OrderNumber) as TotalNumeriCCountDistinct Resident Temp;`	TotalNumericCountDistinct 6 Поскольку существует один элемент OrderNumber, который дублирует другой элемент, результатом будет значение 6. Эти элементы не являются дубликатами.

Пример:

```
Temp:
LOAD * inline [
Customer|Product|OrderNumber|UnitSales|UnitPrice
Astrida|AA|1|4|16
Astrida|AA|7|10|15
Astrida|BB|4|9|9
Betacab|CC|6|5|10
Betacab|AA|5|2|20
Betacab|BB||| 25
Canutility|AA|||15
Canutility|CC| ||19
Divadip|CC|2|4|16
Divadip|DD|7|1|25
] (delimiter is '|');
NumCount1:
LOAD Customer,NumericCount(OrderNumber) as NumericCountByCustomer Resident Temp Group By
Customer;
```

Результирующая таблица

Клиент	NumericCountByCustomer
Astrida	3
Betacab	2
Canutility	0
Divadip	2

NumericCount – функция диаграммы

NumericCount() используется для агрегирования числовых значений в каждом измерении диаграммы.

Синтаксис:

```
NumericCount({[SetExpression] [DISTINCT] [TOTAL [<fld {,fld}>]]} expr)
```

Возвращаемые типы данных: целое

Аргументы:

Аргументы

Аргумент	Описание
expr	Выражение или поле, содержащее данные для измерения.
set_ expression	По умолчанию функция агрегирования агрегирует множество возможных записей, определенных выборкой. Альтернативный набор записей может быть определен выражением анализа множества.

Аргумент	Описание
DISTINCT	Если слово **DISTINCT** указывается до аргументов функции, все дубликаты, возникшие в результате оценки аргументов функции, будут проигнорированы.
TOTAL	Если слово **TOTAL** стоит перед аргументами функции, вычисление выполняется по всем возможным значениям, указанным в текущих выборках, а не только в тех, которые относятся к значению текущего измерения, т. е. измерения диаграммы игнорируются. При использовании выражения **TOTAL [<fld {.fld}>]**, где префикс **TOTAL** предшествует списку из одного или нескольких имен полей, выступающих в качестве подмножества переменных измерения диаграммы, создается подмножество всех возможных значений.

Примеры и результаты:

Data

Customer	Product	OrderNumber	UnitSales	Unit Price
Astrida	AA	1	4	16
Astrida	AA	7	10	15
Astrida	BB	4	9	1
Betacab	BB	6	5	10
Betacab	CC	5	2	20
Betacab	DD			25
Canutility	AA			15
Canutility	CC			19
Divadip	AA	2	4	16
Divadip	DD	3		25

В следующих примерах считается, что все клиенты выбраны, если не указано иначе.

Примеры и результаты

Пример	Результат
`NumericCount ([OrderNumber])`	Значение 7, поскольку 3 из 10 полей в элементе OrderNumber пустые. *«0» считается значением, а не пустой ячейкой. Тем не менее, если мера агрегирует значение для измерения до 0, это измерение не будет включено в диаграммы.*
`NumericCount ([Product])`	Значение 0, поскольку все имена продуктов указаны в тексте. Обычно данную операцию можно использовать, чтобы убедиться, что в текстовых полях нет числового содержимого.
`NumericCount (DISTINCT [OrderNumber])/Count (DISTINCT [OrderNumber)]`	Подсчитывается количество всех уникальных числовых заказов и делится по количеству числовых и не числовых заказов. Если все значения полей числовые, это значение будет равно 1. Обычно данный способ можно использовать, чтобы убедиться, что все значения в полях числовые. В этом примере имеется 7 уникальных числовых значений для элемента OrderNumber из 8 уникальных числовых и нечисловых значений, поэтому выражение возвращает 0,875.

Данные, используемые в примере:

```
Temp:
LOAD * inline [
Customer|Product|OrderNumber|UnitSales|UnitPrice
Astrida|AA|1|4|16
Astrida|AA|7|10|15
Astrida|BB|4|9|9
Betacab|CC|6|5|10
Betacab|AA|5|2|20
Betacab|BB||| 25
Canutility|AA|||15
Canutility|CC| ||19
Divadip|CC|2|4|16
Divadip|DD|3|1|25
] (delimiter is '|');
```

TextCount

Функция **TextCount()** возвращает число нечисловых значений поля, агрегированных в выражении, как определено предложением **group by**.

Синтаксис:

```
TextCount ( [ distinct ] expr)
```

Возвращаемые типы данных: целое

Аргументы:

Аргументы

Аргумент	Описание
expr Expression	Выражение или поле, содержащее данные для измерения.
distinct	Если слово **distinct** указано перед выражением, все дубликаты будут проигнорированы.

Примеры и результаты:

Добавьте образец скрипта в свое приложение и запустите. Затем добавьте на лист приложения как минимум поля, указанные в столбце с результатами, чтобы увидеть результаты.

Чтобы столбец с результатами выглядел так же, как столбец с результатами ниже, на панели свойств в разделе «Сортировка» переключите параметр с «Авто» на «Пользовательский», а затем отмените выбор числовой сортировки и сортировки в алфавитном порядке.

Пример:

```
Temp:
LOAD * inline [
Customer|Product|OrderNumber|UnitSales|UnitPrice
Astrida|AA|1|4|16
Astrida|AA|7|10|15
Astrida|BB|4|9|9
Betacab|CC|6|5|10
Betacab|AA|5|2|20
Betacab|BB||| 25
Canutility|AA|||15
Canutility|CC| ||19
Divadip|CC|2|4|16
Divadip|DD|3|1|25
] (delimiter is '|');
TextCount1:
LOAD Customer,TextCount(Product) as ProductTextCount Resident Temp Group By Customer;
```

Результирующая таблица

Клиент	ProductTextCount
Astrida	3
Betacab	3
Canutility	2
Divadip	2

Пример:

```
LOAD Customer,TextCount(OrderNumber) as OrderNumberTextCount Resident Temp Group By Customer;
```

Результирующая таблица

Клиент	OrderNumberTextCount
Astrida	0
Betacab	1
Canutility	2
Divadip	0

TextCount – функция диаграммы

TextCount() используется для агрегирования нечисловых значений поля в каждом измерении диаграммы.

Синтаксис:

```
TextCount({[SetExpression] [DISTINCT] [TOTAL [<fld {,fld}>]]} expr)
```

Возвращаемые типы данных: целое

Аргументы:

Аргументы

Аргумент	Описание
expr	Выражение или поле, содержащее данные для измерения.
SetExpression	По умолчанию функция агрегирования агрегирует множество возможных записей, определенных выборкой. Альтернативный набор записей может быть определен выражением анализа множества.
DISTINCT	Если слово **DISTINCT** указывается до аргументов функции, все дубликаты, возникшие в результате оценки аргументов функции, будут проигнорированы.
TOTAL	Если слово **TOTAL** стоит перед аргументами функции, вычисление выполняется по всем возможным значениям, указанным в текущих выборках, а не только в тех, которые относятся к значению текущего измерения, т. е. измерения диаграммы игнорируются. При использовании выражения **TOTAL [<fld {.fld}>]**, где префикс **TOTAL** предшествует списку из одного или нескольких имен полей, выступающих в качестве подмножества переменных измерения диаграммы, создается подмножество всех возможных значений.

Примеры и результаты:

Data

Customer	Product	OrderNumber	UnitSales	Unit Price
Astrida	AA	1	4	16
Astrida	AA	7	10	15
Astrida	BB	4	9	1
Betacab	BB	6	5	10
Betacab	CC	5	2	20
Betacab	DD			25
Canutility	AA			15
Canutility	CC			19
Divadip	AA	2	4	16
Divadip	DD	3		25

Примеры и результаты

Пример	Результат
`TextCount ([Product])`	Значение 10, поскольку все из 10 полей в элементе Product текстовые. *«0» считается значением, а не пустой ячейкой. Тем не менее, если мера агрегирует значение для измерения до 0, это измерение не будет включено в диаграммы. Пустые ячейки оцениваются как не текстовые и не учитываются элементом TextCount.*
`TextCount ([OrderNumber])`	Значение 3, поскольку пустые ячейки учитываются. Обычно используется, чтобы убедиться, что в числовых полях нет текстовых или не нулевых значений.
`TextCount (DISTINCT [Product])/Count ([Product)]`	Подсчитывается количество уникальных текстовых значений Product (4) и делится по итоговому количеству значений в элементе Product (10). Результат – 0,4.

Данные, используемые в примере:

```
Temp:
LOAD * inline [
Customer|Product|OrderNumber|UnitSales|UnitPrice
Astrida|AA|1|4|16
```

```
Astrida|AA|7|1|15
Astrida|BB|4|9|9
Betacab|CC|6|5|10
Betacab|AA|5|2|20
Betacab|BB|||| 25
Canutility|AA|||15
Canutility|CC|||19
Divadip|CC|2|4|16
Divadip|DD|3|1|25
] (delimiter is '|');
```

Функции финансового агрегирования

В этом разделе описаны функции агрегирования для финансовых операций в отношении платежей и денежного потока.

Каждая функция подробно описана после обзора. Также можно щелкнуть имя функции в синтаксисе, чтобы получить немедленный доступ к подробной информации об этой конкретной функции.

Функции финансового агрегирования в скрипте загрузки данных

IRR

Функция **IRR()** возвращает агрегированную внутреннюю ставку доходов для серии потоков денежных средств, представленных числами выражений, повторяемых в нескольких записях так, как это определено предложением group by.

```
IRR (expression)
```

XIRR

Функция **XIRR()** возвращает агрегированную внутреннюю ставку доходов для графика потоков денежных средств (не обязательно регулярных), представленных парными числами в элементах **pmt** и **date**, повторяемых в нескольких записях так, как это определено предложением group by. Все платежи учитываются на основе года с 365 днями.

```
XIRR (valueexpression, dateexpression )
```

NPV

NPV() возвращает агрегированную чистую текущую стоимость вложения на основе **discount_rate** за период и ряда будущих платежей (отрицательные значения) и поступлений (положительные значения), представленных числами в элементе **value**, повторяемом в нескольких записях так, как это определено предложением group by. Предполагается, что платежи и поступления происходят в конце каждого периода.

```
NPV (rate, expression)
```

XNPV

Функция **XNPV()** возвращает агрегированную чистую текущую стоимость для графика потоков денежных средств (не обязательно регулярных), представленных парными числами в элементах **pmt** и **date**, повторяемых в нескольких записях так, как это определено предложением group by.

Rate – это процентная ставка за период. Все платежи учитываются на основе года с 365 днями.

```
XNPV (rate, valueexpression, dateexpression)
```

Функции финансового агрегирования в выражениях диаграмм

Эти функции финансового агрегирования можно использовать в диаграммах.

IRR

IRR() возвращает агрегированную внутреннюю ставку доходов для серии потоков денежных средств, представленных числами в выражении, выданном элементом **value**, повторяемом в измерениях диаграммы.

```
IRR – функция диаграммы[TOTAL [<fld {,fld}>]] value)
```

NPV

Функция **NPV()** возвращает агрегированную чистую стоимость инвестиций на основе скидки **discount_rate** за период, серии будущих платежей (отрицательные значения) и дохода (положительные значения), представленных числами в элементе **value**, повторяемом в измерениях диаграммы. Предполагается, что платежи и поступления происходят в конце каждого периода.

```
NPV – функция диаграммы([TOTAL [<fld {,fld}>]] discount_rate, value)
```

XIRR

XIRR() возвращает агрегированную внутреннюю ставку доходов для графика потоков денежных средств (не обязательно периодических), представленных парными числами в выражениях, выданных элементами **pmt** и **date**, повторяемыми в измерениях диаграммы. Все платежи учитываются на основе года с 365 днями.

```
XIRR – функция диаграммы (page 277)([TOTAL [<fld {,fld}>]] pmt, date)
```

XNPV

XNPV() возвращает агрегированную чистую стоимость для графика потоков денежных средств (не обязательно периодических), представленных парными числами в выражениях, выданных элементами **pmt** и **date**, повторяемыми в измерениях диаграммы. Все платежи учитываются на основе года с 365 днями.

```
XNPV – функция диаграммы([TOTAL [<fld{,fld}>]] discount_rate, pmt, date)
```

IRR

Функция **IRR()** возвращает агрегированную внутреннюю ставку доходов для серии потоков денежных средств, представленных числами выражений, повторяемых в нескольких записях так, как это определено предложением group by.

Эти потоки денежных средств не обязаны быть равномерными, как ежегодные платежи. Однако потоки денежных средств должны осуществляться с регулярными интервалами, например ежемесячно или ежегодно. Внутренняя ставка доходов является процентной ставкой, полученной

по вложению и состоящей из платежей (отрицательные значения) и поступлений (положительные значения), которые происходят в равные промежутки. Для вычисления функции необходимо не менее одного отрицательного и одного положительного значений.

Синтаксис:

```
IRR(value)
```

Возвращаемые типы данных: числовое значение

Аргументы:

Аргументы

Аргумент	Описание
value	Выражение или поле, содержащее данные для измерения.

Ограничения:

Текстовые значения, значения NULL и отсутствующие значения игнорируются.

Примеры и результаты:

Добавьте образец скрипта в свое приложение и запустите. Чтобы увидеть результаты, добавьте поля, перечисленные в столбце результатов, в лист приложения.

Примеры и результаты:

Примеры и результаты

Пример	Год	IRR2013
Cashflow: LOAD 2013 as Year, * inline [Date|Discount|Payments 2013-01-01|0.1|-10000 2013-03-01|0.1|3000 2013-10-30|0.1|4200 2014-02-01|0.2|6800] (delimiter is '|'); Cashflow1: LOAD Year,IRR(Payments) as IRR2013 Resident Cashflow Group By Year;	2013	0.1634

IRR – функция диаграммы

IRR() возвращает агрегированную внутреннюю ставку доходов для серии потоков денежных средств, представленных числами в выражении, выданном элементом **value**, повторяемом в измерениях диаграммы.

Эти потоки денежных средств не обязаны быть равномерными, как ежегодные платежи. Однако потоки денежных средств должны осуществляться с регулярными интервалами, например ежемесячно или ежегодно. Внутренняя ставка доходов – это процентная ставка для инвестиций,

состоящих из платежей (отрицательные значения) и дохода (положительные значения), осуществляемых регулярно. Для вычисления этой функции необходимо по крайней мере одно положительное и одно отрицательное значение.

Синтаксис:

```
IRR([TOTAL [<fld {,fld}>]] value)
```

Возвращаемые типы данных: числовое значение

Аргументы:

Аргументы

Аргумент	Описание
value	Выражение или поле, содержащее данные для измерения.
TOTAL	Если слово **TOTAL** стоит перед аргументами функции, вычисление выполняется по всем возможным значениям, указанным в текущих выборках, а не только в тех, которые относятся к значению текущего измерения, т. е. измерения диаграммы игнорируются. При использовании выражения **TOTAL [<fld {.fld}>]**, где префикс **TOTAL** предшествует списку из одного или нескольких имен полей, выступающих в качестве подмножества переменных измерения диаграммы, создается подмножество всех возможных значений.

Ограничения:

Параметр функции агрегирования не должен содержать другие функции агрегирования, кроме внутреннего агрегирования, содержащего классификатор **TOTAL**. Для получения более расширенных вложенных агрегирований необходимо использовать расширенную функцию **Aggr** вместе с указанным измерением.

Текстовые значения, значения NULL и отсутствующие значения игнорируются.

Примеры и результаты:

Примеры и результаты

Пример	Результат
`IRR (Payments)`	0,1634 Предполагается, что платежи являются периодическими, например ежемесячными. *Поле Date используется в примере XIRR, где платежи могут быть не периодическими, если указываются даты, в которые совершаются платежи.*

Данные, используемые в примерах:

```
Cashflow:
LOAD 2013 as Year, * inline [
Date|Discount|Payments
2013-01-01|0.1|-10000
2013-03-01|0.1|3000
2013-10-30|0.1|4200
2014-02-01|0.2|6800
] (delimiter is '|');
```

См. также:

- *XIRR – функция диаграммы (page 277)*
- *Aggr – функция диаграммы (page 438)*

NPV

NPV() возвращает агрегированную чистую текущую стоимость вложения на основе **discount_rate** за период и ряда будущих платежей (отрицательные значения) и поступлений (положительные значения), представленных числами в элементе **value**, повторяемом в нескольких записях так, как это определено предложением group by. Предполагается, что платежи и поступления происходят в конце каждого периода.

Синтаксис:

```
NPV(discount_rate, value)
```

Возвращаемые типы данных: числовой. Результат имеет числовой денежный формат по умолчанию.

Аргументы:

Аргументы

Аргумент	Описание
discount_rate	**discount_rate** – это льготный тариф за какой-либо период.
value	Выражение или поле, содержащее данные для измерения.

Ограничения:

Текстовые значения, значения NULL и отсутствующие значения игнорируются.

Примеры и результаты:

Добавьте образец скрипта в свое приложение и запустите. Чтобы увидеть результаты, добавьте поля, перечисленные в столбце результатов, в лист приложения.

Примеры и результаты

Пример	Год	NPV1_2013
Cashflow: LOAD 2013 as Year, * inline [Date|Discount|Payments 2013-01-01|0.1|-10000 2013-03-01|0.1|3000 2013-10-30|0.1|4200 2014-02-01|0.2|6800] (delimiter is '|'); Cashflow1: LOAD Year,NPV(0.2, Payments) as NPV1_2013 Resident Cashflow Group By Year;	2013	-$540.12

Примеры и результаты

Пример	Год	Скидка	NPV2_ 2013
При условии, что таблица **Cashflow** загружается, как в предыдущем примере: LOAD Year,NPV(Discount, Payments) as NPV2_2013 Resident Cashflow Group By Year, Discount; Обратите внимание, что предложение Group By сортирует результаты по элементам Year и Discount. Первый аргумент discount_rate дан в виде поля (Discount), а не специального номера, и, таким образом, требуется второй критерий сортировки. Поле может содержать различные значения, поэтому агрегированные записи должны быть отсортированы, чтобы обеспечить различные значения Year и Discount. ;	2013 2013	0,1 0,2	-$3456,05 $5666,67

NPV – функция диаграммы

Функция **NPV()** возвращает агрегированную чистую стоимость инвестиций на основе скидки **discount_rate** за период, серии будущих платежей (отрицательные значения) и дохода (положительные значения), представленных числами в элементе **value**, повторяемом в измерениях диаграммы. Предполагается, что платежи и поступления происходят в конце каждого периода.

Синтаксис:

```
NPV([TOTAL [<fld {,fld}>]] discount_rate, value)
```

Возвращаемые типы данных: числовое значение Результат имеет числовой денежный формат по умолчанию.

Аргументы:

Аргументы

Аргумент	Описание
discount_rate	**discount_rate** – это льготный тариф за какой-либо период.
value	Выражение или поле, содержащее данные для измерения.
TOTAL	Если слово **TOTAL** стоит перед аргументами функции, вычисление выполняется по всем возможным значениям, указанным в текущих выборках, а не только в тех, которые относятся к значению текущего измерения, т. е. измерения диаграммы игнорируются. При использовании выражения **TOTAL [<fld {.fld}>]**, где префикс **TOTAL** предшествует списку из одного или нескольких имен полей, выступающих в качестве подмножества переменных измерения диаграммы, создается подмножество всех возможных значений. После классификатора **TOTAL** может быть указан список, включающий одно или несколько имен полей в угловых скобках. Эти имена полей должны быть поднабором переменных измерений диаграммы. В этом случае при вычислении будут проигнорированы все переменные измерений диаграммы, кроме перечисленных, то есть одно значение возвращается для каждого сочетания значений полей в перечисленных полях измерений. Поля, которые в текущий момент не являются измерением в диаграмме, могут также включаться в список. Это может быть полезно для измерений группы, в которых поля измерений не фиксированы. Перечисление всех переменных в группе вызывает выполнение функции при изменении уровня детализации.

Ограничения:

Элементы **discount_rate** и **value** не должны содержать функции агрегирования, если только внутреннее агрегирование не содержит префикс **TOTAL**. Для получения более расширенных вложенных агрегирований необходимо использовать расширенную функцию **Aggr** вместе с указанным измерением.

Текстовые значения, значения NULL и отсутствующие значения игнорируются.

Примеры и результаты:

Примеры и результаты

Пример	Результат
`NPV(Discount, Payments)`	-$540.12

Данные, используемые в примерах:

```
Cashflow:
LOAD 2013 as Year, * inline [
Date|Discount|Payments
2013-01-01|0.1|-10000
2013-03-01|0.1|3000
2013-10-30|0.1|4200
2014-02-01|0.2|6800
] (delimiter is '|');
```

См. также:

- *XNPV – функция диаграммы (page 280)*
- *Aggr – функция диаграммы (page 438)*

XIRR

Функция **XIRR()** возвращает агрегированную внутреннюю ставку доходов для графика потоков денежных средств (не обязательно регулярных), представленных парными числами в элементах **pmt** и **date**, повторяемых в нескольких записях так, как это определено предложением group by. Все платежи учитываются на основе года с 365 днями.

Синтаксис:

```
XIRR(pmt, date )
```

Возвращаемые типы данных: числовое значение

Аргументы:

Аргументы

Аргумент	Описание
pmt	Платежи. Выражение или поле, содержащее потоки денежных средств, соответствующих графику платежей, представленному в элементе **date**.
date	Выражение или поле, содержащее график дат, соответствующих потоку денежных средств, представленному в элементе **pmt**.

Ограничения:

Текстовые, отсутствующие значения и значения NULL в какой-либо или обеих частях пары значений приводят к игнорированию всей пары значений.

Примеры и результаты:

Добавьте образец скрипта в свое приложение и запустите. Чтобы увидеть результаты, добавьте поля, перечисленные в столбце результатов, в лист приложения.

Примеры и результаты

Пример	Год	XIRR2013
Cashflow: LOAD 2013 as Year, * inline [Date|Discount|Payments 2013-01-01|0.1|-10000 2013-03-01|0.1|3000 2013-10-30|0.1|4200 2014-02-01|0.2|6800] (delimiter is '|'); Cashflow1: LOAD Year,XIRR(Payments, Date) as XIRR2013 Resident Cashflow Group By Year;	2013	0.5385

XIRR – функция диаграммы

XIRR() возвращает агрегированную внутреннюю ставку доходов для графика потоков денежных средств (не обязательно периодических), представленных парными числами в выражениях, выданных элементами **pmt** и **date**, повторяемыми в измерениях диаграммы. Все платежи учитываются на основе года с 365 днями.

Синтаксис:

```
XIRR([TOTAL [<fld {,fld}>]] pmt, date)
```

Возвращаемые типы данных: числовое значение

Аргументы:

Аргументы

Аргумент	Описание
pmt	Платежи. Выражение или поле, содержащее потоки денежных средств, соответствующих графику платежей, представленному в элементе **date**.
date	Выражение или поле, содержащее график дат, соответствующих потоку денежных средств, представленному в элементе **pmt**.
TOTAL	Если слово **TOTAL** стоит перед аргументами функции, вычисление выполняется по всем возможным значениям, указанным в текущих выборках, а не только в тех, которые относятся к значению текущего измерения, т. е. измерения диаграммы игнорируются. При использовании выражения **TOTAL [<fld {,fld}>]**, где префикс **TOTAL** предшествует списку из одного или нескольких имен полей, выступающих в качестве подмножества переменных измерения диаграммы, создается подмножество всех возможных значений.

Ограничения:

Элементы **pmt** и **date** не должны содержать функции агрегирования, если только внутреннее агрегирование не содержит префикс **TOTAL**. Для получения более расширенных вложенных агрегирований необходимо использовать расширенную функцию **Aggr** вместе с указанным измерением.

Текстовые значения, значения NULL и отсутствующие значения в какой-либо или обеих частях пары значений приводят к игнорированию всей пары значений.

Примеры и результаты:

Примеры и результаты

Пример	Результат
`XIRR(Payments, Date)`	0,5385

Данные, используемые в примерах:

```
Cashflow:
LOAD 2013 as Year, * inline [
Date|Discount|Payments
2013-01-01|0.1|-10000
2013-03-01|0.1|3000
2013-10-30|0.1|4200
2014-02-01|0.2|6800
] (delimiter is '|');
```

См. также:

- *IRR – функция диаграммы (page 271)*
- *Aggr – функция диаграммы (page 438)*

XNPV

Функция **XNPV()** возвращает агрегированную чистую текущую стоимость для графика потоков денежных средств (не обязательно регулярных), представленных парными числами в элементах **pmt** и **date**, повторяемых в нескольких записях так, как это определено предложением group by. Rate – это процентная ставка за период. Все платежи учитываются на основе года с 365 днями.

Синтаксис:

```
XNPV(discount_rate, pmt, date)
```

Возвращаемые типы данных: числовой. Результат имеет числовой денежный формат по умолчанию. .

Аргументы:

Аргументы

Аргумент	Описание
discount_rate	**discount_rate** – это льготный тариф за какой-либо период.
pmt	Выражение или поле, содержащее данные для измерения.
date	Выражение или поле, содержащее график дат, соответствующих потоку денежных средств, представленному в элементе **pmt**.

Ограничения:

Текстовые, отсутствующие значения и значения NULL в какой-либо или обеих частях пары значений приводят к игнорированию всей пары значений.

Примеры:

Добавьте образец скрипта в свое приложение и запустите. Чтобы увидеть результаты, добавьте поля, перечисленные в столбце результатов, в лист приложения.

Примеры и результаты

Пример	Год	XNPV1_2013
Cashflow: LOAD 2013 as Year, * inline [Date|Discount|Payments 2013-01-01|0.1|-10000 2013-03-01|0.1|3000 2013-10-30|0.1|4200 2014-02-01|0.2|6800] (delimiter is '|'); Cashflow1: LOAD Year,XNPV(0.2, Payments, Date) as XNPV1_2013 Resident Cashflow Group By Year;	2013	$2104.37

Примеры и результаты

Пример	Год	Скидка	XNPV2_2013

При условии, что таблица **Cashflow** загружается, как в предыдущем примере:
`LOAD Year,XNPV(Discount, Payments, Date) as XNPV2_2013`
`Resident Cashflow Group By Year, Discount;`
Обратите внимание, что предложение Group By сортирует результаты по элементам Year и Discount. Первый аргумент discount_rate дан в виде поля (Discount), а не специального номера, и, таким образом, требуется второй критерий сортировки. Поле может содержать различные значения, поэтому агрегированные записи должны быть отсортированы, чтобы обеспечить различные значения Year и Discount.
	2013
	2013
		0,1
		0,2
			-$3164,35
			$6800,00

XNPV – функция диаграммы

XNPV() возвращает агрегированную чистую стоимость для графика потоков денежных средств (не обязательно периодических), представленных парными числами в выражениях, выданных элементами **pmt** и **date**, повторяемыми в измерениях диаграммы. Все платежи учитываются на основе года с 365 днями.

Синтаксис:

```
XNPV([TOTAL [<fld{,fld}>]] discount_rate, pmt, date)
```

Возвращаемые типы данных: числовое значение Результат имеет числовой денежный формат по умолчанию.

Аргументы:

Аргументы

Аргумент	Описание
discount_rate	**discount_rate** – это льготный тариф за какой-либо период.
pmt	Платежи. Выражение или поле, содержащее потоки денежных средств, соответствующих графику платежей, представленному в элементе **date**.
date	Выражение или поле, содержащее график дат, соответствующих потоку денежных средств, представленному в элементе **pmt**.

Аргумент	Описание
TOTAL	Если слово **TOTAL** стоит перед аргументами функции, вычисление выполняется по всем возможным значениям, указанным в текущих выборках, а не только в тех, которые относятся к значению текущего измерения, т. е. измерения диаграммы игнорируются. При использовании выражения **TOTAL [<fld {.fld}>]**, где префикс **TOTAL** предшествует списку из одного или нескольких имен полей, выступающих в качестве подмножества переменных измерения диаграммы, создается подмножество всех возможных значений.

Ограничения:

Элементы **discount_rate**, **pmt** и **date** не должны содержать функции агрегирования, если только эти внутренние агрегирования не содержат префиксы **TOTAL** или **ALL**. Для получения более расширенных вложенных агрегирований необходимо использовать расширенную функцию **Aggr** вместе с указанным измерением.

Текстовые значения, значения NULL и отсутствующие значения в какой-либо или обеих частях пары значений приводят к игнорированию всей пары значений.

Примеры и результаты:

Примеры и результаты

Пример	Результат
`XNPV(Discount, Payments, Date)`	-$3164,35

Данные, используемые в примерах:

```
Cashflow:
LOAD 2013 as Year, * inline [
Date|Discount|Payments
2013-01-01|0.1|-10000
2013-03-01|0.1|3000
2013-10-30|0.1|4200
2014-02-01|0.2|6800
] (delimiter is '|');
```

См. также:

NPV – функция диаграммы (page 274)
Aggr – функция диаграммы (page 438)

Функции статистического агрегирования

Каждая функция подробно описана после обзора. Также можно щелкнуть имя функции в синтаксисе, чтобы получить немедленный доступ к подробной информации об этой конкретной функции.

Функции статистического агрегирования в скрипте загрузки данных

В скриптах можно использовать следующие статистические функции агрегирования.

Avg

Функция **Avg()** находит среднее значение агрегированных данных в выражении в нескольких записях, как это определено предложением **group by**.

```
Avg  ([distinct] expression)
```

Correl

Функция **Correl()** возвращает агрегированный коэффициент корреляции для серии координат, представленных парными числами в выражениях x-expression и y-expression, повторяемых в нескольких записях так, как это определено предложением **group by**.

```
Correl (x-expression, y-expression)
```

Fractile

Функция **Fractile()** находит среди записей, определяемых условием **group by**, значение, соответствующее квантилю агрегированных данных, задаваемых выражением (метод включения).

```
Fractile (expression, fractile)
```

FractileExc

Функция **FractileExc()** находит среди записей, определяемых условием **group by**, значение, соответствующее квантилю агрегированных данных, задаваемых выражением (метод исключения).

```
FractileExc (expression, fractile)
```

Kurtosis

Функция **Kurtosis()** возвращает эксцесс данных в выражении в нескольких записях, как это определено предложением **group by**.

```
Kurtosis  ([distinct ] expression )
```

LINEST_B

Функция **LINEST_B()** возвращает агрегированное значение b (отрезок на оси y) линейной регрессии, определенной уравнением y=mx+b для серии координат, представленных парными числами в выражениях x-expression и y-expression, повторяемых в нескольких записях так, как это определено предложением **group by**.

```
LINEST_B (y-expression, x-expression [, y0 [, x0 ]])
```

LINEST_df

Функция **LINEST_DF()** возвращает агрегированное значение степеней свободы линейной регрессии, определенной уравнением y=mx+b для серии координат, представленных парными числами в выражениях x-expression и y-expression, повторяемых в нескольких записях так, как это определено предложением **group by**.

```
LINEST_DF  (y-expression, x-expression [, y0 [, x0 ]])
```

LINEST_f

Эта функция скрипта возвращает агрегированную статистику F ($r^2/(1-r^2)$) линейной регрессии, определенной уравнением y=mx+b для серии координат, представленных парными числами в выражениях x-expression и y-expression, повторяемых в нескольких записях так, как это определено выражением **group by**.

```
LINEST_F (y-expression, x-expression [, y0 [, x0 ]])
```

LINEST_m

Функция **LINEST_M()** возвращает агрегированное значение m (пересечение) линейной регрессии, определенной уравнением y=mx+b для серии координат, представленных парными числами в выражениях x-expression и y-expression, повторяемых в нескольких записях так, как это определено предложением **group by**.

```
LINEST_M  (y-expression, x-expression [, y0 [, x0 ]])
```

LINEST_r2

LINEST_R2() возвращает агрегированное значение r^2 (коэффициент детерминации) линейной регрессии, определенной уравнением y=mx+b для серии координат, представленных парными числами в выражениях x-expression и y-expression, повторяемых в нескольких записях так, как это определено предложением **group by**.

```
LINEST_R2 (y-expression, x-expression [, y0 [, x0 ]])
```

LINEST_seb

Функция **LINEST_SEB()** возвращает агрегированную стандартную ошибку значения b линейной регрессии, определенной уравнением y=mx+b для серии координат, представленных парными числами в выражениях x-expression и y-expression, повторяемых в нескольких записях так, как это определено предложением **group by**.

```
LINEST_SEB  (y-expression, x-expression [, y0 [, x0 ]])
```

LINEST_sem

Функция **LINEST_SEM()** возвращает агрегированную стандартную ошибку значения m линейной регрессии, определенной уравнением y=mx+b для серии координат, представленных парными числами в выражениях x-expression и y-expression, повторяемых в нескольких записях так, как это определено предложением **group by**.

```
LINEST_SEM  (y-expression, x-expression [, y0 [, x0 ]])
```

LINEST_sey

Функция **LINEST_SEY()** возвращает агрегированную стандартную ошибку оценки y линейной регрессии, определенной уравнением y=mx+b для серии координат, представленных парными числами в выражениях x-expression и y-expression, повторяемых в нескольких записях так, как это определено предложением **group by**.

```
LINEST_SEY (y-expression, x-expression [, y0 [, x0 ]])
```

LINEST_ssreg

LINEST_SSREG() возвращает агрегированную остаточную сумму квадратов линейной регрессии, определенной уравнением y=mx+b для серии координат, представленных парными числами в выражениях x-expression и y-expression, повторяемых в нескольких записях так, как это определено предложением **group by**.

```
LINEST_SSREG (y-expression, x-expression [, y0 [, x0 ]])
```

Linest_ssresid

Функция **LINEST_SSRESID()** возвращает агрегированную остаточную сумму квадратов линейной регрессии, определенной уравнением y=mx+b для серии координат, представленных парными числами в выражениях x-expression и y-expression, повторяемых в нескольких записях так, как это определено предложением **group by**.

```
LINEST_SSRESID (y-expression, x-expression [, y0 [, x0 ]])
```

Median

Функция **Median()** возвращает среднее значение агрегированных данных в выражении в нескольких записях, как это определено предложением **group by**.

```
Median (expression)
```

Skew

Функция **Skew()** возвращает асимметрию выражения в нескольких записях, как это определено предложением **group by**.

```
Skew ([ distinct] expression)
```

Stdev

Функция **Stdev()** возвращает стандартное отклонение значений в выражении в нескольких записях, как это определено предложением **group by**.

```
Stdev ([distinct] expression)
```

Sterr

Функция **Sterr()** возвращает агрегированную стандартную ошибку (stdev/sqrt(n)) для серии значений, представленных выражением, повторяемым в нескольких записях так, как это определено предложением **group by**.

```
Sterr ([distinct] expression)
```

STEYX

Функция **STEYX()** возвращает агрегированную стандартную ошибку предсказанного значения y для каждого значения x в регрессии для серии координат, представленных парными числами в выражениях x-expression и y-expression, повторяемых в нескольких записях так, как это определено предложением **group by**.

```
STEYX  (y-expression, x-expression)
```

Функции статистического агрегирования в выражениях диаграмм

Следующие функции статистического агрегирования можно использовать в диаграммах.

Avg

Функция **Avg()** возвращает агрегированное среднее значение выражения или поля, повторяемых в измерениях диаграммы.

```
Avg — функция диаграммы({[SetExpression] [DISTINCT] [TOTAL [<fld{, fld}>]]}
expr)
```

Correl

Функция **Correl()** возвращает агрегированный коэффициент корреляции для двух наборов данных. Функция корреляции – это мера отношений между наборами данных. Она агрегирована для пар значений (x,y), повторяемых в измерениях диаграммы.

```
Correl — функция диаграммы({[SetExpression] [TOTAL [<fld {, fld}>]]} value1,
value2 )
```

Fractile

Функция **Fractile()** находит значение, соответствующее квантилю агрегированных данных в диапазоне, полученном из выражения, выполняющего итерации по измерениям диаграммы (метод включения).

```
Fractile — функция диаграммы({[SetExpression] [TOTAL [<fld {, fld}>]]} expr,
fraction)
```

FractileExc

Функция **FractileExc()** находит значение, соответствующее квантилю агрегированных данных в диапазоне, полученном из выражения, выполняющего итерации по измерениям диаграммы (метод исключения).

```
FractileExc — функция диаграммы({[SetExpression] [TOTAL [<fld {, fld}>]]}
expr, fraction)
```

Kurtosis

Функция **Kurtosis()** находит эксцесс диапазона данных, агрегированных в выражении или поле, повторяемых в измерениях диаграммы.

```
Kurtosis — функция диаграммы({[SetExpression] [DISTINCT] [TOTAL [<fld{,
fld}>]]} expr)
```

LINEST_b

Функция **LINEST_B()** возвращает агрегированное значение b (отрезок на оси y) линейной регрессии, определенной уравнением y=mx+b для серии координат, представленных парными числами в выражениях **x_value** и **y_value**, повторяемых в измерениях диаграммы.

```
LINEST_R2 – функция диаграммы({[SetExpression] [TOTAL [<fld{ ,fld}>]] }y_
value, x_value[, y0_const[, x0_const]])
```

LINEST_df

Функция **LINEST_DF()** возвращает агрегированные степени свободы линейной регрессии, определенной уравнением y=mx+b для серии координат, представленных парными числами в выражениях **x_value** и **y_value**, повторяемых в измерениях диаграммы.

```
LINEST_DF – функция диаграммы({[SetExpression] [TOTAL [<fld{, fld}>]]} y_
value, x_value [, y0_const [, x0_const]])
```

LINEST_f

Функция **LINEST_F()** возвращает агрегированное статистическое F (r2/(1-r2)) линейной регрессии, определенной уравнением y=mx+b для серии координат, представленных парными числами в выражениях **x_value** и **y_value**, повторяемых в измерениях диаграммы.

```
LINEST_F – функция диаграммы({[SetExpression] [TOTAL[<fld{, fld}>]]} y_value,
x_value [, y0_const [, x0_const]])
```

LINEST_m

Функция **LINEST_M()** возвращает агрегированное значение m (пересечение) линейной регрессии, определенной уравнением y=mx+b для серии координат, представленных парными числами в выражениях **x_value** и **y_value**, повторяемых в измерениях диаграммы.

```
LINEST_M – функция диаграммы({[SetExpression] [TOTAL[<fld{, fld}>]]} y_value,
x_value [, y0_const [, x0_const]])
```

LINEST_r2

Функция **LINEST_R2()** возвращает агрегированное значение r2 (коэффициент детерминации) линейной регрессии, определенной уравнением y=mx+b для серии координат, представленных парными числами в выражениях **x_value** и **y_value**, повторяемых в измерениях диаграммы.

```
LINEST_R2 – функция диаграммы({[SetExpression] [TOTAL [<fld{ ,fld}>]] }y_
value, x_value[, y0_const[, x0_const]])
```

LINEST_seb

Функция **LINEST_SEB()** возвращает агрегированную стандартную ошибку значения b линейной регрессии, определенной уравнением y=mx+b для серии координат, представленных парными числами в выражениях **x_value** и **y_value**, повторяемых в измерениях диаграммы.

```
LINEST_SEB – функция диаграммы({[SetExpression] [TOTAL [<fld{ ,fld}>]] }y_
value, x_value[, y0_const[, x0_const]])
```

LINEST_sem

Функция **LINEST_SEM()** возвращает агрегированную стандартную ошибку значения m линейной регрессии, определенной уравнением y=mx+b для серии координат, представленных парными числами в выражениях **x_value** и **y_value**, повторяемых в измерениях диаграммы.

```
LINEST_SEM — функция диаграммы([{set_expression}][ distinct ] [total [<fld
{,fld}>] ] y-expression, x-expression [, y0 [, x0 ]] )
```

LINEST_sey

Функция **LINEST_SEY()** возвращает агрегированную стандартную ошибку значения y линейной регрессии, определенной уравнением y=mx+b для серии координат, представленных парными числами в выражениях **x_value** и **y_value**, повторяемых в измерениях диаграммы.

```
LINEST_SEY — функция диаграммы({[SetExpression] [TOTAL [<fld{ ,fld}>]] }y_
value, x_value[, y0_const[, x0_const]])
```

LINEST_ssreg

Функция **LINEST_SSREG()** возвращает агрегированную сумму регрессии площадей линейной регрессии, определенной уравнением y=mx+b для серии координат, представленных парными числами в выражениях **x_value** и **y_value**, повторяемых в измерениях диаграммы.

```
LINEST_SSREG — функция диаграммы({[SetExpression] [TOTAL [<fld{ ,fld}>]] }y_
value, x_value[, y0_const[, x0_const]])
```

LINEST_ssresid

Функция **LINEST_SSRESID()** возвращает агрегированную остаточную сумму площадей линейной регрессии, определенной уравнением y=mx+b для серии координат, представленных парными числами в выражениях **x_value** и **y_value**, повторяемых в измерениях диаграммы.

```
LINEST_SSRESID — функция диаграммы Функция LINEST_SSRESID() возвращает
агрегированную остаточную сумму площадей линейной регрессии, определенной
уравнением y=mx+b для серии координат, представленных парными числами в
выражениях x_value и y_value, повторяемых в измерениях диаграммы.  LINEST_
SSRESID([{SetExpression}] [DISTINCT] [TOTAL [<fld{, fld}>]] y_value, x_value
[, y0_const[, x0_const]]) числовое значение АргументыАргументОписаниеy_
valueВыражение или поле, содержащее диапазон значений y для измерения.x_
valueВыражение или поле, содержащее диапазон значений x для измерения.y0,
x0Дополнительное значение y0 можно указать путем принудительного прохождения
линии регрессии через ось y в определенной точке. Указав y0 и x0, можно
задать принудительное прохождение линии регрессии через одиночную
фиксированную координату. Если значения y0 и x0 не указаны, для вычисления
функции требуются хотя бы две допустимые пары значений. Если y0 и x0 указаны,
используется одна пара значений. SetExpressionПо умолчанию функция
агрегирования агрегирует множество возможных записей, определенных выборкой.
Альтернативный набор записей может быть определен выражением анализа
множества. DISTINCTЕсли слово DISTINCT указывается до аргументов функции, все
дубликаты, возникшие в результате оценки аргументов функции, будут
проигнорированы. TOTALЕсли слово TOTAL стоит перед аргументами функции,
```

вычисление выполняется по всем возможным значениям, указанным в текущих выборках, а не только в тех, которые относятся к значению текущего измерения, т. е. измерения диаграммы игнорируются. При использовании выражения TOTAL [<fld {.fld}>], где префикс TOTAL предшествует списку из одного или нескольких имен полей, выступающих в качестве подмножества переменных измерения диаграммы, создается подмножество всех возможных значений.Дополнительное значение y0 можно указать путем принудительного прохождения линии регрессии через ось y в определенной точке. Указав y0 и x0, можно задать принудительное прохождение линии регрессии через одиночную фиксированную координату. Параметр функции агрегирования не должен содержать другие функции агрегирования, кроме внутреннего агрегирования, содержащего классификатор TOTAL. Для получения более расширенных вложенных агрегирований необходимо использовать расширенную функцию Aggr вместе с указанным измерением. Текстовые значения, значения NULL и отсутствующие значения в какой-либо или обеих частях пары значений приводят к игнорированию всей пары значений. An example of how to use linest functionsavg({[SetExpression] [**TOTAL** [<fld{ ,fld}>]] }y_value, x_value[, y0_const[, x0_const]]**)

Median

Функция **Median()** возвращает значение медианы диапазона значений, агрегированных в выражении, повторяемом в измерениях диаграммы.

```
Median — функция диаграммы({[SetExpression] [TOTAL [<fld{, fld}>]]} expr)
```

MutualInfo

MutualInfo рассчитывает взаимную информацию (mutual information, MI) между двумя полями или агрегированными значениями в **Aggr()**.

```
MutualInfo — функция диаграммы (page 332){[SetExpression] [DISTINCT] [TOTAL
target, driver [, datatype [, breakdownbyvalue [, samplesize ]]])
```

Skew

Функция **Skew()** возвращает агрегированную асимметрию значений выражения или поля, повторяемых в измерениях диаграммы.

```
Skew — функция диаграммы{[SetExpression] [DISTINCT] [TOTAL [<fld{ ,fld}>]]}
expr)
```

Stdev

Функция **Stdev()** находит стандартное отклонение диапазона данных, агрегированных в выражении или поле, повторяемых в измерениях диаграммы.

```
Stdev — функция диаграммы({[SetExpression] [DISTINCT] [TOTAL [<fld{, fld}>]]}
expr)
```

Sterr

Функция **Sterr()** находит значение стандартной ошибки среднего значения (stdev/sqrt(n)) для серии значений, агрегированных в выражении, повторяемом в измерениях диаграммы.

```
Sterr — функция диаграммы({[SetExpression] [DISTINCT] [TOTAL[<fld{, fld}>]]}
expr)
```

STEYX

Функция **STEYX()** возвращает агрегированную стандартную ошибку во время предсказания значения y для каждого значения x в линейной регрессии, определенной серией координат, представленных парными числами в выражениях **y_value** и **x_value**.

```
STEYX — функция диаграммы{[SetExpression] [TOTAL [<fld{, fld}>]]} y_value, x_
value)
```

Avg

Функция **Avg()** находит среднее значение агрегированных данных в выражении в нескольких записях, как это определено предложением **group by**.

Синтаксис:

```
Avg([DISTINCT] expr)
```

Возвращаемые типы данных: числовое значение

Аргументы:

Аргументы

Аргумент	Описание
expr	Выражение или поле, содержащее данные для измерения.
DISTINCT	Если слово **distinct** указано перед выражением, все дубликаты будут проигнорированы.

Примеры и результаты:

Добавьте образец скрипта в свое приложение и запустите. Чтобы увидеть результаты, добавьте поля, перечисленные в столбце результатов, в лист приложения.

Результирующие данные

Пример	Результат
Temp: crosstable (Month, Sales) load * inline [Customer|Jan|Feb|Mar||Apr|May|Jun|Jul|Aug|Sep|Oct|Nov|Dec Astrida|46|60|70|13|78|20|45|65|78|12|78|22 Betacab|65|56|22|79|12|56|45|24|32|78|55|15 Canutility|77|68|34|91|24|68|57|36|44|90|67|27 Divadip|36|44|90|67|27|57|68|47|90|80|94] (delimiter is '|'); Avg1: LOAD Customer, Avg(Sales) as MyAverageSalesByCustomer Resident Temp Group By Customer;	Customer MyAverageSalesByCustomer Astrida 48.916667 Betacab 44.916667 Canutility 56.916667 Divadip 63.083333 Это можно проверить на листе путем создания таблицы, включая меру: Sum(Sales)/12
При условии, что таблица **Temp** загружается, как в предыдущем примере: LOAD Customer,Avg(DISTINCT Sales) as MyAvgSalesDistinct Resident Temp Group By Customer;	Customer MyAverageSalesByCustomer Astrida 43.1 Betacab 43.909091 Canutility 55.909091 Divadip 61 Учитываются только уникальные значения. Поделите итоговое значение на количество неповторяющихся значений.

Avg – функция диаграммы

Функция **Avg()** возвращает агрегированное среднее значение выражения или поля, повторяемых в измерениях диаграммы.

Синтаксис:

```
Avg([{SetExpression}] [DISTINCT] [TOTAL [<fld{, fld}>]] expr)
```

Возвращаемые типы данных: числовое значение

Аргументы:

Аргументы

Аргумент	Описание
expr	Выражение или поле, содержащее данные для измерения.
SetExpression	По умолчанию функция агрегирования агрегирует множество возможных записей, определенных выборкой. Альтернативный набор записей может быть определен выражением анализа множества.

Аргумент	Описание
DISTINCT	Если слово **DISTINCT** указывается до аргументов функции, все дубликаты, возникшие в результате оценки аргументов функции, будут проигнорированы.
TOTAL	Если слово **TOTAL** стоит перед аргументами функции, вычисление выполняется по всем возможным значениям, указанным в текущих выборках, а не только в тех, которые относятся к значению текущего измерения, т. е. измерения диаграммы игнорируются. При использовании выражения **TOTAL [<fld {.fld}>]**, где префикс **TOTAL** предшествует списку из одного или нескольких имен полей, выступающих в качестве подмножества переменных измерения диаграммы, создается подмножество всех возможных значений.

Ограничения:

Параметр функции агрегирования не должен содержать другие функции агрегирования, кроме внутреннего агрегирования, содержащего классификатор **TOTAL**. Для получения более расширенных вложенных агрегирований необходимо использовать расширенную функцию **Aggr** вместе с указанным измерением.

Примеры и результаты:

Example table

Customer	Jan	Feb	Mar	Apr	May	Jun	Jul	Aug	Sep	Oct	Nov	Dec
Astrida	46	60	70	13	78	20	45	65	78	12	78	22
Betacab	65	56	22	79	12	56	45	24	32	78	55	15
Canutility	77	68	34	91	24	68	57	36	44	90	67	27
Divadip	57	36	44	90	67	27	57	68	47	90	80	94

Примеры функции

Пример	Результат
`Avg(Sales)`	Для таблицы, включающей измерение `Customer` и меру `Avg([Sales])`, если показано значение **Итоги**, результат будет 2566.
`Avg([TOTAL (Sales))`	53,458333 для всех значений элемента `Customer`, поскольку классификатор TOTAL означает, что измерения игнорируются.
`Avg (DISTINCT (Sales))`	51,862069 для итогового значения, поскольку использование классификатора Distinct означает, что оцениваются только уникальные значения в поле `Sales` для каждого элемента `Customer`.

Данные, используемые в примерах:

```
Monthnames:
LOAD *, Dual(MonthText,MonthNumber) as Month INLINE [
```

```
MonthText, MonthNumber
Jan, 1
Feb, 2
Mar, 3
Apr, 4
May, 5
Jun, 6
Jul, 7
Aug, 8
Sep, 9
Oct, 10
Nov, 11
Dec, 12
];

Sales2013:
Crosstable (MonthText, Sales) LOAD * inline [
Customer|Jan|Feb|Mar|Apr|May|Jun|Jul|Aug|Sep|Oct|Nov|Dec
Astrida|46|60|70|13|78|20|45|65|78|12|78|22
Betacab|65|56|22|79|12|56|45|24|32|78|55|15
Canutility|77|68|34|91|24|68|57|36|44|90|67|27
Divadip|57|36|44|90|67|27|57|68|47|90|80|94
] (delimiter is '|');
```

См. также:

Aggr – функция диаграммы (page 438)

Correl

Функция **Correl()** возвращает агрегированный коэффициент корреляции для серии координат, представленных парными числами в выражениях x-expression и y-expression, повторяемых в нескольких записях так, как это определено предложением **group by**.

Синтаксис:

```
Correl(value1, value2)
```

Возвращаемые типы данных: числовое значение

Аргументы:

Аргументы

Аргумент	Описание
value1, value2	Выражения или поля, содержащие два образца множеств, для которых необходимо измерить коэффициент корреляции.

Ограничения:

Текстовые значения, значения NULL и отсутствующие значения в какой-либо или обеих частях пары значений приводят к игнорированию всей пары значений.

Примеры и результаты:

Добавьте образец скрипта в свое приложение и запустите. Чтобы увидеть результаты, добавьте поля, перечисленные в столбце результатов, в лист приложения.

Результирующие данные

Пример	Результат

`Salary:`
`Load *, 1 as Grp;`
`LOAD * inline [`
`"Employee name"|Gender|Age|Salary`
`Aiden Charles|Male|20|25000`
`Brenda Davies|Male|25|32000`
`Charlotte Edberg|Female|45|56000`
`Daroush Ferrara|Male|31|29000`
`Eunice Goldblum|Female|31|32000`
`Freddy Halvorsen|Male|25|26000`
`Gauri Indu|Female|36|46000`
`Harry Jones|Male|38|40000`
`Ian Underwood|Male|40|45000`
`Jackie Kingsley|Female|23|28000`
`] (delimiter is '|');`
 	
`Correl1:`
`LOAD Grp,`
`Correl(Age,Salary) as Correl_Salary Resident Salary Group By Grp;`
	В таблице с измерением `Correl_Salary`, результат вычисления Correl() в скрипте загрузки данных будет показан как: 0,9270611

Correl – функция диаграммы

Функция **Correl()** возвращает агрегированный коэффициент корреляции для двух наборов данных. Функция корреляции – это мера отношений между наборами данных. Она агрегирована для пар значений (x,y), повторяемых в измерениях диаграммы.

Синтаксис:

```
Correl([{SetExpression}] [DISTINCT] [TOTAL [<fld{, fld}>]] value1, value2 )
```

Возвращаемые типы данных: числовое значение

Аргументы:

Аргументы

Аргумент	Описание
value1, value2	Выражения или поля, содержащие два образца множеств, для которых необходимо измерить коэффициент корреляции.
SetExpression	По умолчанию функция агрегирования агрегирует множество возможных записей, определенных выборкой. Альтернативный набор записей может быть определен выражением анализа множества.
DISTINCT	Если слово **DISTINCT** указывается до аргументов функции, все дубликаты, возникшие в результате оценки аргументов функции, будут проигнорированы.
TOTAL	Если слово **TOTAL** стоит перед аргументами функции, вычисление выполняется по всем возможным значениям, указанным в текущих выборках, а не только в тех, которые относятся к значению текущего измерения, т. е. измерения диаграммы игнорируются. При использовании выражения **TOTAL [<fld {.fld}>]**, где префикс **TOTAL** предшествует списку из одного или нескольких имен полей, выступающих в качестве подмножества переменных измерения диаграммы, создается подмножество всех возможных значений.

Ограничения:

Параметр функции агрегирования не должен содержать другие функции агрегирования, кроме внутреннего агрегирования, содержащего классификатор **TOTAL**. Для получения более расширенных вложенных агрегирований необходимо использовать расширенную функцию **Aggr** вместе с указанным измерением.

Текстовые значения, значения NULL и отсутствующие значения в какой-либо или обеих частях пары значений приводят к игнорированию всей пары значений.

Примеры и результаты:

Примеры функции

Пример	Результат
`Correl (Age, Salary)`	Для таблицы, включающей измерение `Employee name` и меру `Correl(Age, Salary)`, результат будет 0,9270611. Результат отображается только для итоговой ячейки.

Пример	Результат
`Correl (TOTAL Age, Salary))`	0.927. Этот и следующие результаты показаны в формате с тремя знаками после десятичной запятой для удобства считывания. При создании фильтра с измерением Gender и выборками из него полученный результат составит 0,951, если выбран элемент Female, и 0,939, если выбран элемент Male. Это обусловлено тем, что выборка исключает все результаты, которые не принадлежат другому значению элемента Gender.
`Correl({1} TOTAL Age, Salary))`	0.927. Независимо от выборок. Это обусловлено тем, что выражение множества {1} игнорирует все выборки и измерения.
`Correl (TOTAL <Gender> Age, Salary))`	0,927 в итоговой ячейке, 0,939 для всех значений элемента Male и 0,951 для всех значений элемента Female. Это соответствует результатам при выполнении выборок в фильтре на основе элемента Gender.

Данные, используемые в примерах:

```
Salary:
LOAD * inline [
"Employee name"|Gender|Age|Salary
Aiden Charles|Male|20|25000
Brenda Davies|Male|25|32000
Charlotte Edberg|Female|45|56000
Daroush Ferrara|Male|31|29000
Eunice Goldblum|Female|31|32000
Freddy Halvorsen|Male|25|26000
Gauri Indu|Female|36|46000
Harry Jones|Male|38|40000
Ian Underwood|Male|40|45000
Jackie Kingsley|Female|23|28000
] (delimiter is '|');
```

См. также:

- *Aggr – функция диаграммы (page 438)*
- *Avg – функция диаграммы (page 290)*
- *RangeCorrel (page 748)*

Fractile

Функция **Fractile()** находит среди записей, определяемых условием **group by**, значение, соответствующее квантилю агрегированных данных, задаваемых выражением (метод включения).

Используйте функцию FractileExc (page 300) для расчета квантиля по методу исключения.

Синтаксис:

```
Fractile(expr, fraction)
```

Возвращаемые типы данных: числовое значение

Функция возвращает значение, соответствующее порядку, который определяется уравнением: `порядок = квантиль * (N+1)`, где `N` — набор значений, задаваемый выражением `expr`. Если `порядок` не является целым числом, выполняется интерполяция между двумя ближайшими целыми числами.

Аргументы:

Аргументы

Аргумент	Описание
expr	Выражение или поле, содержащие данные, которые используются для вычисления квантиля.
fraction	Число от 0 до 1, соответствующее квантилю (выраженному в дробном виде), которое подлежит вычислению.

Примеры и результаты:

Добавьте образец скрипта в свое приложение и запустите. Чтобы увидеть результаты, добавьте поля, перечисленные в столбце результатов, в лист приложения.

Результирующие данные

Пример	Результат
Table1: crosstable LOAD recno() as ID, * inline [Observation|Comparison 35|2 40|27 12|38 15|31 21|1 14|19 46|1 10|34 28|3 48|1 16|2 30|3 32|2 48|1 31|2 22|1 12|3 39|29 19|37 25|2] (delimiter is '|'); Fractile1: LOAD Type, Fractile(Value,0.75) as MyFractile Resident Table1 Group By Type;	В таблице с измерениями Type и MyFractile результаты вычислений Fractile() в скрипте загрузки данных будут показаны как: Type MyFractile Comparison 27.5 Observation 36

Fractile – функция диаграммы

Функция **Fractile()** находит значение, соответствующее квантилю агрегированных данных в диапазоне, полученном из выражения, выполняющего итерации по измерениям диаграммы (метод включения).

Используйте функцию FractileExc – функция диаграммы (page 301) для расчета квантиля по методу исключения.

Синтаксис:

```
Fractile([{SetExpression}] [DISTINCT] [TOTAL [<fld{, fld}>]] expr, fraction)
```

Возвращаемые типы данных: числовое значение

Функция возвращает значение, соответствующее порядку, который определяется уравнением: порядок = квантиль * (N+1), где N – набор значений, задаваемый выражением expr. Если порядок не является целым числом, выполняется интерполяция между двумя ближайшими целыми

числами.

Аргументы:

Аргументы

Аргумент	Описание
expr	Выражение или поле, содержащие данные, которые используются для вычисления квантиля.
fraction	Число от 0 до 1, соответствующее квантилю (выраженному в дробном виде), которое подлежит вычислению.
SetExpression	По умолчанию функция агрегирования агрегирует множество возможных записей, определенных выборкой. Альтернативный набор записей может быть определен выражением анализа множества.
DISTINCT	Если слово **DISTINCT** указывается до аргументов функции, все дубликаты, возникшие в результате оценки аргументов функции, будут проигнорированы.
TOTAL	Если слово **TOTAL** стоит перед аргументами функции, вычисление выполняется по всем возможным значениям, указанным в текущих выборках, а не только в тех, которые относятся к значению текущего измерения, т. е. измерения диаграммы игнорируются. При использовании выражения **TOTAL [<fld {.fld}>]**, где префикс **TOTAL** предшествует списку из одного или нескольких имен полей, выступающих в качестве подмножества переменных измерения диаграммы, создается подмножество всех возможных значений.

Ограничения:

Параметр функции агрегирования не должен содержать другие функции агрегирования, кроме внутреннего агрегирования, содержащего классификатор **TOTAL**. Для получения более расширенных вложенных агрегирований необходимо использовать расширенную функцию **Aggr** вместе с указанным измерением.

Примеры и результаты:

Example table

Customer	Jan	Feb	Mar	Apr	May	Jun	Jul	Aug	Sep	Oct	Nov	Dec
Astrida	46	60	70	13	78	20	45	65	78	12	78	22
Betacab	65	56	22	79	12	56	45	24	32	78	55	15
Canutility	77	68	34	91	24	68	57	36	44	90	67	27
Divadip	57	36	44	90	67	27	57	68	47	90	80	94

Примеры функции

Пример	Результат
`Fractile (Sales, 0.75)`	Для таблицы, включающей измерение `Customer` и меру `Fractile([Sales])`, если показано значение **Итоги**, результат будет 71,75. Это точка в распределении значений элемента `Sales`, ниже которой находится 75% значений.
`Fractile (TOTAL Sales, 0.75))`	71,75 для всех значений элемента `Customer`, поскольку классификатор TOTAL означает, что измерения игнорируются.
`Fractile (DISTINCT Sales, 0.75)`	70 для итогового значения, поскольку использование классификатора DISTINCT означает, что оцениваются только уникальные значения в поле `Sales` для каждого элемента `Customer`.

Данные, используемые в примерах:

```
Monthnames:
LOAD *, Dual(MonthText,MonthNumber) as Month INLINE [
MonthText, MonthNumber
Jan, 1
Feb, 2
Mar, 3
Apr, 4
May, 5
Jun, 6
Jul, 7
Aug, 8
Sep, 9
Oct, 10
Nov, 11
Dec, 12
];

Sales2013:
Crosstable (MonthText, Sales) LOAD * inline [
Customer|Jan|Feb|Mar|Apr|May|Jun|Jul|Aug|Sep|Oct|Nov|Dec
Astrida|46|60|70|13|78|20|45|65|78|12|78|22
Betacab|65|56|22|79|12|56|45|24|32|78|55|15
Canutility|77|68|34|91|24|68|57|36|44|90|67|27
Divadip|57|36|44|90|67|27|57|68|47|90|80|94
] (delimiter is '|');
```

См. также:

Aggr – функция диаграммы (page 438)

FractileExc

Функция **FractileExc()** находит среди записей, определяемых условием **group by**, значение, соответствующее квантилю агрегированных данных, задаваемых выражением (метод исключения).

Используйте функцию Fractile (page 295) для расчета квантиля по методу включения.

Синтаксис:

```
FractileExc(expr, квантиль)
```

Возвращаемые типы данных: числовое значение

Функция возвращает значение, соответствующее порядку, который определяется уравнением: `порядок = квантиль * (N+1)`, где `N` – набор значений, задаваемый выражением `expr`. Если `порядок` не является целым числом, выполняется интерполяция между двумя ближайшими целыми числами.

Аргументы:

Аргументы

Аргумент	Описание
expr	Выражение или поле, содержащие данные, которые используются для вычисления квантиля.
fraction	Число от 0 до 1, соответствующее квантилю (выраженному в дробном виде), которое подлежит вычислению.

Примеры и результаты:

Добавьте образец скрипта в свое приложение и запустите. Чтобы увидеть результаты, добавьте поля, перечисленные в столбце результатов, в лист приложения.

Результирующие данные

Пример	Результат
Table1: crosstable LOAD recno() as ID, * inline [Observation|Comparison 35|2 40|27 12|38 15|31 21|1 14|19 46|1 10|34 28|3 48|1 16|2 30|3 32|2 48|1 31|2 22|1 12|3 39|29 19|37 25|2] (delimiter is '|'); Fractile1: LOAD Type, FractileExc(Value,0.75) as MyFractile Resident Table1 Group By Type;	В таблице с измерениями `Type` и `MyFractile` результаты вычислений FractileExc() в скрипте загрузки данных будут показаны как: Type MyFractile Comparison 28.5 Observation 38

FractileExc – функция диаграммы

Функция **FractileExc()** находит значение, соответствующее квантилю агрегированных данных в диапазоне, полученном из выражения, выполняющего итерации по измерениям диаграммы (метод исключения).

Используйте функцию Fractile – функция диаграммы (page 297) для расчета квантиля по методу включения.

Синтаксис:

```
FractileExc([{SetExpression}] [DISTINCT] [TOTAL [<fld{, fld}>]] expr,
fraction)
```

Возвращаемые типы данных: числовое значение

Функция возвращает значение, соответствующее порядку, который определяется уравнением: `порядок = квантиль * (N+1)`, где `N` — набор значений, задаваемый выражением `expr`. Если `порядок` не является целым числом, выполняется интерполяция между двумя ближайшими целыми числами.

Аргументы:

Аргументы

Аргумент	Описание
expr	Выражение или поле, содержащие данные, которые используются для вычисления квантиля.
fraction	Число от 0 до 1, соответствующее квантилю (выраженному в дробном виде), которое подлежит вычислению.
SetExpression	По умолчанию функция агрегирования агрегирует множество возможных записей, определенных выборкой. Альтернативный набор записей может быть определен выражением анализа множества.
DISTINCT	Если слово **DISTINCT** указывается до аргументов функции, все дубликаты, возникшие в результате оценки аргументов функции, будут проигнорированы.
TOTAL	Если слово **TOTAL** стоит перед аргументами функции, вычисление выполняется по всем возможным значениям, указанным в текущих выборках, а не только в тех, которые относятся к значению текущего измерения, т. е. измерения диаграммы игнорируются. При использовании выражения **TOTAL [<fld {.fld}>]**, где префикс **TOTAL** предшествует списку из одного или нескольких имен полей, выступающих в качестве подмножества переменных измерения диаграммы, создается подмножество всех возможных значений.

Ограничения:

Параметр функции агрегирования не должен содержать другие функции агрегирования, кроме внутреннего агрегирования, содержащего классификатор **TOTAL**. Для получения более расширенных вложенных агрегирований необходимо использовать расширенную функцию **Aggr** вместе с указанным измерением.

Примеры и результаты:

Example table

Customer	Jan	Feb	Mar	Apr	May	Jun	Jul	Aug	Sep	Oct	Nov	Dec
Astrida	46	60	70	13	78	20	45	65	78	12	78	22

Customer	Jan	Feb	Mar	Apr	May	Jun	Jul	Aug	Sep	Oct	Nov	Dec
Betacab	65	56	22	79	12	56	45	24	32	78	55	15
Canutility	77	68	34	91	24	68	57	36	44	90	67	27
Divadip	57	36	44	90	67	27	57	68	47	90	80	94

Примеры функции

Пример	Результат
`FractileExc (Sales, 0.75)`	Для таблицы, включающей измерение `Customer` и меру `FractileExc([Sales])`, если показано значение **Итоги**, результат будет равен 75,25. Это точка в распределении значений элемента `Sales`, ниже которой находится 75% значений.
`FractileExc (TOTAL Sales, 0.75))`	75,25 для всех значений `Customer`, поскольку классификатор TOTAL означает, что измерения игнорируются.
`FractileExc (DISTINCT Sales, 0.75)`	73,50 для итогового значения, поскольку использование классификатора DISTINCT означает, что оцениваются только уникальные значения в поле `Sales` для каждого элемента `Customer`.

Данные, используемые в примерах:

```
Monthnames:
LOAD *, Dual(MonthText,MonthNumber) as Month INLINE [
MonthText, MonthNumber
Jan, 1
Feb, 2
Mar, 3
Apr, 4
May, 5
Jun, 6
Jul, 7
Aug, 8
Sep, 9
Oct, 10
Nov, 11
Dec, 12
];

Sales2013:
Crosstable (MonthText, Sales) LOAD * inline [
Customer|Jan|Feb|Mar|Apr|May|Jun|Jul|Aug|Sep|Oct|Nov|Dec
Astrida|46|60|70|13|78|20|45|65|78|12|78|22
Betacab|65|56|22|79|12|56|45|24|32|78|55|15
Canutility|77|68|34|91|24|68|57|36|44|90|67|27
Divadip|57|36|44|90|67|27|57|68|47|90|80|94
] (delimiter is '|');
```

См. также:

- *Aggr – функция диаграммы (page 438)*

Kurtosis

Функция **Kurtosis()** возвращает эксцесс данных в выражении в нескольких записях, как это определено предложением **group by**.

Синтаксис:

```
Kurtosis([distinct ] expr )
```

Возвращаемые типы данных: числовое значение

Аргументы:

Аргументы

Аргумент	Описание
expr	Выражение или поле, содержащее данные для измерения.
distinct	Если слово **distinct** указано перед выражением, все дубликаты будут проигнорированы.

Примеры и результаты:

Добавьте образец скрипта в свое приложение и запустите. Чтобы увидеть результаты, добавьте поля, перечисленные в столбце результатов, в лист приложения.

Результирующие данные

Пример	Результат
Table1: crosstable LOAD recno() as ID, * inline [Observation|Comparison 35|2 40|27 12|38 15|31 21|1 14|19 46|1 10|34 28|3 48|1 16|2 30|3 32|2 48|1 31|2 22|1 12|3 39|29 19|37 25|2] (delimiter is '|'); Kurtosis1: LOAD Type, Kurtosis(Value) as MyKurtosis1, Kurtosis(DISTINCT Value) as MyKurtosis2 Resident Table1 Group By Type;	В таблице с измерениями Type, MyKurtosis1 и MyKurtosis2 результаты вычислений Kurtosis() в скрипте загрузки данных будут показаны как: Type MyKurtosis1 MyKurtosis2 Comparison -1.1612957 -1.4982366 Observation -1.1148768 -0.93540144

Kurtosis – функция диаграммы

Функция **Kurtosis()** находит эксцесс диапазона данных, агрегированных в выражении или поле, повторяемых в измерениях диаграммы.

Синтаксис:

```
Kurtosis([{SetExpression}] [DISTINCT] [TOTAL [<fld{, fld}>]] expr)
```

Возвращаемые типы данных: числовое значение

Аргументы:

Аргументы

Аргумент	Описание
expr	Выражение или поле, содержащее данные для измерения.
SetExpression	По умолчанию функция агрегирования агрегирует множество возможных записей, определенных выборкой. Альтернативный набор записей может быть определен выражением анализа множества.
DISTINCT	Если слово **DISTINCT** указывается до аргументов функции, все дубликаты, возникшие в результате оценки аргументов функции, будут проигнорированы.
TOTAL	Если слово **TOTAL** стоит перед аргументами функции, вычисление выполняется по всем возможным значениям, указанным в текущих выборках, а не только в тех, которые относятся к значению текущего измерения, т. е. измерения диаграммы игнорируются. При использовании выражения **TOTAL [<fld {.fld}>]**, где префикс **TOTAL** предшествует списку из одного или нескольких имен полей, выступающих в качестве подмножества переменных измерения диаграммы, создается подмножество всех возможных значений.

Ограничения:

Параметр функции агрегирования не должен содержать другие функции агрегирования, кроме внутреннего агрегирования, содержащего классификатор **TOTAL**. Для получения более расширенных вложенных агрегирований необходимо использовать расширенную функцию **Aggr** вместе с указанным измерением.

Примеры и результаты:

Example table

Type	Value																			
Comparison	2	27	38	31	1	19	1	34	3	1	2	3	2	1	2	1	3	29	37	2
Observation	35	40	12	15	21	14	46	10	28	48	16	30	32	48	31	22	12	39	19	25

Примеры функции

Пример	Результат
`Kurtosis (Value)`	Для таблицы, включающей измерение `Type` и меру `Kurtosis(Value)`, показано значение **Итоги**, форматирование числа задастся на 3 значащие цифры, и результатом будет 1,252. Для элемента `Comparison` это будет 1,161, а для элемента `Observation` – 1,115.
`Kurtosis (TOTAL Value))`	1,252 для всех значений элемента `Type`, поскольку классификатор TOTAL означает, что измерения игнорируются.

Данные, используемые в примерах:

```
Table1:
crosstable LOAD recno() as ID, * inline [
Observation|Comparison
35|2
40|27
12|38
15|31
21|1
14|19
46|1
10|34
28|3
48|1
16|2
30|3
32|2
48|1
31|2
22|1
12|3
39|29
19|37
25|2 ] (delimiter is '|');
```

См. также:

Avg – функция диаграммы (page 290)

LINEST_B

Функция **LINEST_B()** возвращает агрегированное значение b (отрезок на оси y) линейной регрессии, определенной уравнением y=mx+b для серии координат, представленных парными числами в выражениях x-expression и y-expression, повторяемых в нескольких записях так, как это определено предложением **group by**.

Синтаксис:

```
LINEST_B (y_value, x_value[, y0 [, x0 ]])
```

Возвращаемые типы данных: числовое значение

Аргументы:

Аргументы

Аргумент	Описание
y_value	Выражение или поле, содержащее диапазон значений y для измерения.
x_value	Выражение или поле, содержащее диапазон значений x для измерения.
y(0), x(0)	Дополнительное значение y0 можно указать путем принудительного прохождения линии регрессии через ось y в определенной точке. Указав y0 и x0, можно задать принудительное прохождение линии регрессии через одиночную фиксированную координату. Если значения y0 и x0 не указаны, для вычисления функции требуются хотя бы две допустимые пары значений. Если y0 и x0 указаны, используется одна пара значений.

Ограничения:

Текстовые значения, значения NULL и отсутствующие значения в какой-либо или обеих частях пары значений приводят к игнорированию всей пары значений.

См. также:

Примеры использования функций linest (page 350)

LINEST_B – функция диаграммы

Функция **LINEST_B()** возвращает агрегированное значение b (отрезок на оси y) линейной регрессии, определенной уравнением y=mx+b для серии координат, представленных парными числами в выражениях **x_value** и **y_value**, повторяемых в измерениях диаграммы.

Синтаксис:

```
LINEST_B([{SetExpression}] [DISTINCT] [TOTAL [<fld{, fld}>]] y_value, x_value
[, y0_const [ , x0_const]])
```

Возвращаемые типы данных: числовое значение

Аргументы:

Аргументы

Аргумент	Описание
y_value	Выражение или поле, содержащее диапазон значений y для измерения.

Аргумент	Описание
x_value	Выражение или поле, содержащее диапазон значений x для измерения.
y0_const, x0_const	Дополнительное значение y0 можно указать путем принудительного прохождения линии регрессии через ось y в определенной точке. Указав y0 и x0, можно задать принудительное прохождение линии регрессии через одиночную фиксированную координату. *Если значения y0 и x0 не указаны, для вычисления функции требуются хотя бы две допустимые пары значений. Если y0 и x0 указаны, используется одна пара значений.*
SetExpression	По умолчанию функция агрегирования агрегирует множество возможных записей, определенных выборкой. Альтернативный набор записей может быть определен выражением анализа множества.
DISTINCT	Если слово **DISTINCT** указывается до аргументов функции, все дубликаты, возникшие в результате оценки аргументов функции, будут проигнорированы.
TOTAL	Если слово **TOTAL** стоит перед аргументами функции, вычисление выполняется по всем возможным значениям, указанным в текущих выборках, а не только в тех, которые относятся к значению текущего измерения, т. е. измерения диаграммы игнорируются. При использовании выражения **TOTAL [<fld {.fld}>]**, где префикс **TOTAL** предшествует списку из одного или нескольких имен полей, выступающих в качестве подмножества переменных измерения диаграммы, создается подмножество всех возможных значений.

Ограничения:

Параметр функции агрегирования не должен содержать другие функции агрегирования, кроме внутреннего агрегирования, содержащего классификатор **TOTAL**. Для получения более расширенных вложенных агрегирований необходимо использовать расширенную функцию **Aggr** вместе с указанным измерением.

Текстовые значения, значения NULL и отсутствующие значения в какой-либо или обеих частях пары значений приводят к игнорированию всей пары значений.

См. также:

- *Примеры использования функций linest (page 350)*
- *Avg – функция диаграммы (page 290)*

LINEST_DF

Функция **LINEST_DF()** возвращает агрегированное значение степеней свободы линейной регрессии, определенной уравнением y=mx+b для серии координат, представленных парными числами в выражениях x-expression и y-expression, повторяемых в нескольких записях так, как это определено предложением **group by**.

Синтаксис:

```
LINEST_DF (y_value, x_value[, y0 [, x0 ]])
```

Возвращаемые типы данных: числовое значение

Аргументы:

Аргументы

Аргумент	Описание
y_value	Выражение или поле, содержащее диапазон значений y для измерения.
x_value	Выражение или поле, содержащее диапазон значений x для измерения.
y(0), x(0)	Дополнительное значение y0 можно указать путем принудительного прохождения линии регрессии через ось y в определенной точке. Указав y0 и x0, можно задать принудительное прохождение линии регрессии через одиночную фиксированную координату. Если значения y0 и x0 не указаны, для вычисления функции требуются хотя бы две допустимые пары значений. Если y0 и x0 указаны, используется одна пара значений.

Ограничения:

Текстовые значения, значения NULL и отсутствующие значения в какой-либо или обеих частях пары значений приводят к игнорированию всей пары значений.

См. также:

Примеры использования функций linest (page 350)

LINEST_DF – функция диаграммы

Функция **LINEST_DF()** возвращает агрегированные степени свободы линейной регрессии, определенной уравнением y=mx+b для серии координат, представленных парными числами в выражениях **x_value** и **y_value**, повторяемых в измерениях диаграммы.

Синтаксис:

```
LINEST_DF([{SetExpression}] [DISTINCT] [TOTAL [<fld{, fld}>]] y_value, x_
value [, y0_const [, x0_const]])
```

Возвращаемые типы данных: числовое значение

Аргументы:

Аргументы

Аргумент	Описание
y_value	Выражение или поле, содержащее диапазон значений y для измерения.
x_value	Выражение или поле, содержащее диапазон значений x для измерения.
y0, x0	Дополнительное значение y0 можно указать путем принудительного прохождения линии регрессии через ось y в определенной точке. Указав y0 и x0, можно задать принудительное прохождение линии регрессии через одиночную фиксированную координату.  *Если значения y0 и x0 не указаны, для вычисления функции требуются хотя бы две допустимые пары значений. Если y0 и x0 указаны, используется одна пара значений.*
SetExpression	По умолчанию функция агрегирования агрегирует множество возможных записей, определенных выборкой. Альтернативный набор записей может быть определен выражением анализа множества.
DISTINCT	Если слово **DISTINCT** указывается до аргументов функции, все дубликаты, возникшие в результате оценки аргументов функции, будут проигнорированы.
TOTAL	Если слово **TOTAL** стоит перед аргументами функции, вычисление выполняется по всем возможным значениям, указанным в текущих выборках, а не только в тех, которые относятся к значению текущего измерения, т. е. измерения диаграммы игнорируются. При использовании выражения **TOTAL [<fld {.fld}>]**, где префикс **TOTAL** предшествует списку из одного или нескольких имен полей, выступающих в качестве подмножества переменных измерения диаграммы, создается подмножество всех возможных значений.

Ограничения:

Параметр функции агрегирования не должен содержать другие функции агрегирования, кроме внутреннего агрегирования, содержащего классификатор **TOTAL**. Для получения более расширенных вложенных агрегирований необходимо использовать расширенную функцию **Aggr** вместе с указанным измерением.

Текстовые значения, значения NULL и отсутствующие значения в какой-либо или обеих частях пары значений приводят к игнорированию всей пары значений.

См. также:

- *Примеры использования функций linest (page 350)*
- *Avg – функция диаграммы (page 290)*

LINEST_F

Эта функция скрипта возвращает агрегированную статистику F ($r^2/(1-r^2)$) линейной регрессии, определенной уравнением y=mx+b для серии координат, представленных парными числами в выражениях x-expression и y-expression, повторяемых в нескольких записях так, как это определено выражением **group by**.

Синтаксис:

```
LINEST_F (y_value, x_value[, y0 [, x0 ]])
```

Возвращаемые типы данных: числовое значение

Аргументы:

Аргументы

Аргумент	Описание
y_value	Выражение или поле, содержащее диапазон значений y для измерения.
x_value	Выражение или поле, содержащее диапазон значений x для измерения.
y(0), x(0)	Дополнительное значение y0 можно указать путем принудительного прохождения линии регрессии через ось y в определенной точке. Указав y0 и x0, можно задать принудительное прохождение линии регрессии через одиночную фиксированную координату. Если значения y0 и x0 не указаны, для вычисления функции требуются хотя бы две допустимые пары значений. Если y0 и x0 указаны, используется одна пара значений.

Ограничения:

Текстовые значения, значения NULL и отсутствующие значения в какой-либо или обеих частях пары значений приводят к игнорированию всей пары значений.

См. также:

- *Примеры использования функций linest (page 350)*

LINEST_F — функция диаграммы

Функция **LINEST_F()** возвращает агрегированное статистическое F (r2/(1-r2)) линейной регрессии, определенной уравнением y=mx+b для серии координат, представленных парными числами в выражениях **x_value** и **y_value**, повторяемых в измерениях диаграммы.

Синтаксис:

```
LINEST_F([{SetExpression}] [DISTINCT] [TOTAL [<fld{, fld}>]] y_value, x_value
[, y0_const [, x0_const]])
```

Возвращаемые типы данных: числовое значение

Аргументы:

Аргументы

Аргумент	Описание
y_value	Выражение или поле, содержащее диапазон значений y для измерения.
x_value	Выражение или поле, содержащее диапазон значений x для измерения.
y0, x0	Дополнительное значение y0 можно указать путем принудительного прохождения линии регрессии через ось y в определенной точке. Указав y0 и x0, можно задать принудительное прохождение линии регрессии через одиночную фиксированную координату. *Если значения y0 и x0 не указаны, для вычисления функции требуются хотя бы две допустимые пары значений. Если y0 и x0 указаны, используется одна пара значений.*
SetExpression	По умолчанию функция агрегирования агрегирует множество возможных записей, определенных выборкой. Альтернативный набор записей может быть определен выражением анализа множества.
DISTINCT	Если слово **DISTINCT** указывается до аргументов функции, все дубликаты, возникшие в результате оценки аргументов функции, будут проигнорированы.
TOTAL	Если слово **TOTAL** стоит перед аргументами функции, вычисление выполняется по всем возможным значениям, указанным в текущих выборках, а не только в тех, которые относятся к значению текущего измерения, т. е. измерения диаграммы игнорируются. При использовании выражения **TOTAL [<fld {.fld}>]**, где префикс **TOTAL** предшествует списку из одного или нескольких имен полей, выступающих в качестве подмножества переменных измерения диаграммы, создается подмножество всех возможных значений.

Ограничения:

Параметр функции агрегирования не должен содержать другие функции агрегирования, кроме внутреннего агрегирования, содержащего классификатор **TOTAL**. Для получения более расширенных вложенных агрегирований необходимо использовать расширенную функцию **Aggr** вместе с указанным измерением.

Текстовые значения, значения NULL и отсутствующие значения в какой-либо или обеих частях пары значений приводят к игнорированию всей пары значений.

См. также:

- *Примеры использования функций linest (page 350)*
- *Avg – функция диаграммы (page 290)*

LINEST_M

Функция **LINEST_M()** возвращает агрегированное значение m (пересечение) линейной регрессии, определенной уравнением y=mx+b для серии координат, представленных парными числами в выражениях x-expression и y-expression, повторяемых в нескольких записях так, как это определено предложением **group by**.

Синтаксис:

```
LINEST_M (y_value, x_value[, y0 [, x0 ]])
```

Возвращаемые типы данных: числовое значение

Аргументы:

Аргументы

Аргумент	Описание
y_value	Выражение или поле, содержащее диапазон значений y для измерения.
x_value	Выражение или поле, содержащее диапазон значений x для измерения.
y(0), x(0)	Дополнительное значение y0 можно указать путем принудительного прохождения линии регрессии через ось y в определенной точке. Указав y0 и x0, можно задать принудительное прохождение линии регрессии через одиночную фиксированную координату. Если значения y0 и x0 не указаны, для вычисления функции требуются хотя бы две допустимые пары значений. Если y0 и x0 указаны, используется одна пара значений.

Ограничения:

Текстовые значения, значения NULL и отсутствующие значения в какой-либо или обеих частях пары значений приводят к игнорированию всей пары значений.

См. также:

Примеры использования функций linest (page 350)

LINEST_M – функция диаграммы

Функция **LINEST_M()** возвращает агрегированное значение m (пересечение) линейной регрессии, определенной уравнением y=mx+b для серии координат, представленных парными числами в выражениях **x_value** и **y_value**, повторяемых в измерениях диаграммы.

Синтаксис:

```
LINEST_M([{SetExpression}] [DISTINCT] [TOTAL [<fld{, fld}>]] y_value, x_value
[, y0_const [, x0_const]])
```

Возвращаемые типы данных: числовое значение

Аргументы:

Аргументы

Аргумент	Описание
y_value	Выражение или поле, содержащее диапазон значений y для измерения.
x_value	Выражение или поле, содержащее диапазон значений x для измерения.
y0, x0	Дополнительное значение y0 можно указать путем принудительного прохождения линии регрессии через ось y в определенной точке. Указав y0 и x0, можно задать принудительное прохождение линии регрессии через одиночную фиксированную координату. *Если значения y0 и x0 не указаны, для вычисления функции требуются хотя бы две допустимые пары значений. Если y0 и x0 указаны, используется одна пара значений.*
SetExpression	По умолчанию функция агрегирования агрегирует множество возможных записей, определенных выборкой. Альтернативный набор записей может быть определен выражением анализа множества.
DISTINCT	Если слово **DISTINCT** указывается до аргументов функции, все дубликаты, возникшие в результате оценки аргументов функции, будут проигнорированы.

Аргумент	Описание
TOTAL	Если слово **TOTAL** стоит перед аргументами функции, вычисление выполняется по всем возможным значениям, указанным в текущих выборках, а не только в тех, которые относятся к значению текущего измерения, т. е. измерения диаграммы игнорируются. При использовании выражения **TOTAL [<fld {.fld}>]**, где префикс **TOTAL** предшествует списку из одного или нескольких имен полей, выступающих в качестве подмножества переменных измерения диаграммы, создается подмножество всех возможных значений.

Ограничения:

Параметр функции агрегирования не должен содержать другие функции агрегирования, кроме внутреннего агрегирования, содержащего классификатор **TOTAL**. Для получения более расширенных вложенных агрегирований необходимо использовать расширенную функцию **Aggr** вместе с указанным измерением.

Текстовые значения, значения NULL и отсутствующие значения в какой-либо или обеих частях пары значений приводят к игнорированию всей пары значений.

См. также:

- *Примеры использования функций linest (page 350)*
- *Avg – функция диаграммы (page 290)*

LINEST_R2

LINEST_R2() возвращает агрегированное значение r^2 (коэффициент детерминации) линейной регрессии, определенной уравнением y=mx+b для серии координат, представленных парными числами в выражениях x-expression и y-expression, повторяемых в нескольких записях так, как это определено предложением **group by**.

Синтаксис:

```
LINEST_R2 (y_value, x_value[, y0 [, x0 ]])
```

Возвращаемые типы данных: числовое значение

Аргументы:

Аргументы

Аргумент	Описание
y_value	Выражение или поле, содержащее диапазон значений y для измерения.
x_value	Выражение или поле, содержащее диапазон значений x для измерения.

Аргумент	Описание
y(0), x(0)	Дополнительное значение y0 можно указать путем принудительного прохождения линии регрессии через ось y в определенной точке. Указав y0 и x0, можно задать принудительное прохождение линии регрессии через одиночную фиксированную координату. Если значения y0 и x0 не указаны, для вычисления функции требуются хотя бы две допустимые пары значений. Если y0 и x0 указаны, используется одна пара значений.

Ограничения:

Текстовые значения, значения NULL и отсутствующие значения в какой-либо или обеих частях пары значений приводят к игнорированию всей пары значений.

См. также:

Примеры использования функций linest (page 350)

LINEST_R2 – функция диаграммы

Функция **LINEST_R2()** возвращает агрегированное значение r2 (коэффициент детерминации) линейной регрессии, определенной уравнением y=mx+b для серии координат, представленных парными числами в выражениях **x_value** и **y_value**, повторяемых в измерениях диаграммы.

Синтаксис:

```
LINEST_R2([{SetExpression}] [DISTINCT] [TOTAL [<fld{, fld}>]] y_value, x_
value[, y0_const[, x0_const]])
```

Возвращаемые типы данных: числовое значение

Аргументы:

Аргументы

Аргумент	Описание
y_value	Выражение или поле, содержащее диапазон значений y для измерения.
x_value	Выражение или поле, содержащее диапазон значений x для измерения.

Аргумент	Описание
y0, x0	Дополнительное значение y0 можно указать путем принудительного прохождения линии регрессии через ось y в определенной точке. Указав y0 и x0, можно задать принудительное прохождение линии регрессии через одиночную фиксированную координату. *Если значения y0 и x0 не указаны, для вычисления функции требуются хотя бы две допустимые пары значений. Если y0 и x0 указаны, используется одна пара значений.*
SetExpression	По умолчанию функция агрегирования агрегирует множество возможных записей, определенных выборкой. Альтернативный набор записей может быть определен выражением анализа множества.
DISTINCT	Если слово **DISTINCT** указывается до аргументов функции, все дубликаты, возникшие в результате оценки аргументов функции, будут проигнорированы.
TOTAL	Если слово **TOTAL** стоит перед аргументами функции, вычисление выполняется по всем возможным значениям, указанным в текущих выборках, а не только в тех, которые относятся к значению текущего измерения, т. е. измерения диаграммы игнорируются. При использовании выражения **TOTAL [<fld {.fld}>]**, где префикс **TOTAL** предшествует списку из одного или нескольких имен полей, выступающих в качестве подмножества переменных измерения диаграммы, создается подмножество всех возможных значений.

Ограничения:

Параметр функции агрегирования не должен содержать другие функции агрегирования, кроме внутреннего агрегирования, содержащего классификатор **TOTAL**. Для получения более расширенных вложенных агрегирований необходимо использовать расширенную функцию **Aggr** вместе с указанным измерением.

Текстовые значения, значения NULL и отсутствующие значения в какой-либо или обеих частях пары значений приводят к игнорированию всей пары значений.

См. также:

Примеры использования функций linest (page 350)
Avg – функция диаграммы (page 290)

LINEST_SEB

Функция **LINEST_SEB()** возвращает агрегированную стандартную ошибку значения b линейной регрессии, определенной уравнением y=mx+b для серии координат, представленных парными числами в выражениях x-expression и y-expression,

повторяемых в нескольких записях так, как это определено предложением **group by**.

Синтаксис:

```
LINEST_SEB (y_value, x_value[, y0 [, x0 ]])
```

Возвращаемые типы данных: числовое значение

Аргументы:

Аргументы

Аргумент	Описание
y_value	Выражение или поле, содержащее диапазон значений y для измерения.
x_value	Выражение или поле, содержащее диапазон значений x для измерения.
y(0), x(0)	Дополнительное значение y0 можно указать путем принудительного прохождения линии регрессии через ось y в определенной точке. Указав y0 и x0, можно задать принудительное прохождение линии регрессии через одиночную фиксированную координату. Если значения y0 и x0 не указаны, для вычисления функции требуются хотя бы две допустимые пары значений. Если y0 и x0 указаны, используется одна пара значений.

Ограничения:

Текстовые значения, значения NULL и отсутствующие значения в какой-либо или обеих частях пары значений приводят к игнорированию всей пары значений.

См. также:

Примеры использования функций linest (page 350)

LINEST_SEB – функция диаграммы

Функция **LINEST_SEB()** возвращает агрегированную стандартную ошибку значения b линейной регрессии, определенной уравнением y=mx+b для серии координат, представленных парными числами в выражениях **x_value** и **y_value**, повторяемых в измерениях диаграммы.

Синтаксис:

```
LINEST_SEB([{SetExpression}] [DISTINCT] [TOTAL [<fld{, fld}>]] y_value, x_
value[, y0_const[, x0_const]])
```

Возвращаемые типы данных: числовое значение

Аргументы:

Аргументы

Аргумент	Описание
y_value	Выражение или поле, содержащее диапазон значений y для измерения.
x_value	Выражение или поле, содержащее диапазон значений x для измерения.
y0, x0	Дополнительное значение y0 можно указать путем принудительного прохождения линии регрессии через ось y в определенной точке. Указав y0 и x0, можно задать принудительное прохождение линии регрессии через одиночную фиксированную координату.  *Если значения y0 и x0 не указаны, для вычисления функции требуются хотя бы две допустимые пары значений. Если y0 и x0 указаны, используется одна пара значений.*
SetExpression	По умолчанию функция агрегирования агрегирует множество возможных записей, определенных выборкой. Альтернативный набор записей может быть определен выражением анализа множества.
DISTINCT	Если слово **DISTINCT** указывается до аргументов функции, все дубликаты, возникшие в результате оценки аргументов функции, будут проигнорированы.
TOTAL	Если слово **TOTAL** стоит перед аргументами функции, вычисление выполняется по всем возможным значениям, указанным в текущих выборках, а не только в тех, которые относятся к значению текущего измерения, т. е. измерения диаграммы игнорируются. При использовании выражения **TOTAL [<fld {.fld}>]**, где префикс **TOTAL** предшествует списку из одного или нескольких имен полей, выступающих в качестве подмножества переменных измерения диаграммы, создается подмножество всех возможных значений.

Ограничения:

Параметр функции агрегирования не должен содержать другие функции агрегирования, кроме внутреннего агрегирования, содержащего классификатор **TOTAL**. Для получения более расширенных вложенных агрегирований необходимо использовать расширенную функцию **Aggr** вместе с указанным измерением.

Текстовые значения, значения NULL и отсутствующие значения в какой-либо или обеих частях пары значений приводят к игнорированию всей пары значений.

См. также:

- *Примеры использования функций linest (page 350)*
- *Avg – функция диаграммы (page 290)*

LINEST_SEM

Функция **LINEST_SEM()** возвращает агрегированную стандартную ошибку значения m линейной регрессии, определенной уравнением y=mx+b для серии координат, представленных парными числами в выражениях x-expression и y-expression, повторяемых в нескольких записях так, как это определено предложением **group by**.

Синтаксис:

```
LINEST_SEM (y_value, x_value[, y0 [, x0 ]])
```

Возвращаемые типы данных: числовое значение

Аргументы:

Аргумент	Описание
y_value	Выражение или поле, содержащее диапазон значений y для измерения.
x_value	Выражение или поле, содержащее диапазон значений x для измерения.
y(0), x(0)	Дополнительное значение y0 можно указать путем принудительного прохождения линии регрессии через ось y в определенной точке. Указав y0 и x0, можно задать принудительное прохождение линии регрессии через одиночную фиксированную координату. Если значения y0 и x0 не указаны, для вычисления функции требуются хотя бы две допустимые пары значений. Если y0 и x0 указаны, используется одна пара значений.

Ограничения:

Текстовые значения, значения NULL и отсутствующие значения в какой-либо или обеих частях пары значений приводят к игнорированию всей пары значений.

См. также:

- *Примеры использования функций linest (page 350)*

LINEST_SEM – функция диаграммы

Функция **LINEST_SEM()** возвращает агрегированную стандартную ошибку значения m линейной регрессии, определенной уравнением y=mx+b для серии координат, представленных парными числами в выражениях **x_value** и **y_value**, повторяемых в измерениях диаграммы.

Синтаксис:

```
LINEST_SEM([{SetExpression}] [DISTINCT] [TOTAL [<fld{, fld}>]] y_value, x_value[, y0_const[, x0_const]])
```

Возвращаемые типы данных: числовое значение

Аргументы:

Аргументы

Аргумент	Описание
y_value	Выражение или поле, содержащее диапазон значений y для измерения.
x_value	Выражение или поле, содержащее диапазон значений x для измерения.
y0, x0	Дополнительное значение y0 можно указать путем принудительного прохождения линии регрессии через ось y в определенной точке. Указав y0 и x0, можно задать принудительное прохождение линии регрессии через одиночную фиксированную координату. *Если значения y0 и x0 не указаны, для вычисления функции требуются хотя бы две допустимые пары значений. Если y0 и x0 указаны, используется одна пара значений.*
SetExpression	По умолчанию функция агрегирования агрегирует множество возможных записей, определенных выборкой. Альтернативный набор записей может быть определен выражением анализа множества.
DISTINCT	Если слово **DISTINCT** указывается до аргументов функции, все дубликаты, возникшие в результате оценки аргументов функции, будут проигнорированы.
TOTAL	Если слово **TOTAL** стоит перед аргументами функции, вычисление выполняется по всем возможным значениям, указанным в текущих выборках, а не только в тех, которые относятся к значению текущего измерения, т. е. измерения диаграммы игнорируются. При использовании выражения **TOTAL [<fld {.fld}>]**, где префикс **TOTAL** предшествует списку из одного или нескольких имен полей, выступающих в качестве подмножества переменных измерения диаграммы, создается подмножество всех возможных значений.

Ограничения:

Параметр функции агрегирования не должен содержать другие функции агрегирования, кроме внутреннего агрегирования, содержащего классификатор **TOTAL**. Для получения более расширенных вложенных агрегирований необходимо использовать расширенную функцию **Aggr** вместе с указанным измерением.

Текстовые значения, значения NULL и отсутствующие значения в какой-либо или обеих частях пары значений приводят к игнорированию всей пары значений.

См. также:

- *Примеры использования функций linest (page 350)*
- *Avg – функция диаграммы (page 290)*

LINEST_SEY

Функция **LINEST_SEY()** возвращает агрегированную стандартную ошибку оценки y линейной регрессии, определенной уравнением y=mx+b для серии координат, представленных парными числами в выражениях x-expression и y-expression, повторяемых в нескольких записях так, как это определено предложением **group by**.

Синтаксис:

```
LINEST_SEY (y_value, x_value[, y0 [, x0 ]])
```

Возвращаемые типы данных: числовое значение

Аргументы:

Аргумент	Описание
y_value	Выражение или поле, содержащее диапазон значений y для измерения.
x_value	Выражение или поле, содержащее диапазон значений x для измерения.
y(0), x(0)	Дополнительное значение y0 можно указать путем принудительного прохождения линии регрессии через ось y в определенной точке. Указав y0 и x0, можно задать принудительное прохождение линии регрессии через одиночную фиксированную координату. Если значения y0 и x0 не указаны, для вычисления функции требуются хотя бы две допустимые пары значений. Если y0 и x0 указаны, используется одна пара значений.

Ограничения:

Текстовые значения, значения NULL и отсутствующие значения в какой-либо или обеих частях пары значений приводят к игнорированию всей пары значений.

См. также:

- *Примеры использования функций linest (page 350)*

LINEST_SEY – функция диаграммы

Функция **LINEST_SEY()** возвращает агрегированную стандартную ошибку значения y линейной регрессии, определенной уравнением y=mx+b для серии координат, представленных парными числами в выражениях **x_value** и **y_value**, повторяемых в измерениях диаграммы.

Синтаксис:

```
LINEST_SEY([{SetExpression}] [DISTINCT] [TOTAL [<fld{, fld}>]] y_value, x_value[, y0_const[, x0_const]])
```

Возвращаемые типы данных: числовое значение

Аргументы:

Аргументы

Аргумент	Описание
y_value	Выражение или поле, содержащее диапазон значений y для измерения.
x_value	Выражение или поле, содержащее диапазон значений x для измерения.
y0, x0	Дополнительное значение y0 можно указать путем принудительного прохождения линии регрессии через ось y в определенной точке. Указав y0 и x0, можно задать принудительное прохождение линии регрессии через одиночную фиксированную координату. *Если значения y0 и x0 не указаны, для вычисления функции требуются хотя бы две допустимые пары значений. Если y0 и x0 указаны, используется одна пара значений.*
SetExpression	По умолчанию функция агрегирования агрегирует множество возможных записей, определенных выборкой. Альтернативный набор записей может быть определен выражением анализа множества.
DISTINCT	Если слово **DISTINCT** указывается до аргументов функции, все дубликаты, возникшие в результате оценки аргументов функции, будут проигнорированы.
TOTAL	Если слово **TOTAL** стоит перед аргументами функции, вычисление выполняется по всем возможным значениям, указанным в текущих выборках, а не только в тех, которые относятся к значению текущего измерения, т. е. измерения диаграммы игнорируются. При использовании выражения **TOTAL [<fld {.fld}>]**, где префикс **TOTAL** предшествует списку из одного или нескольких имен полей, выступающих в качестве подмножества переменных измерения диаграммы, создается подмножество всех возможных значений.

Ограничения:

Параметр функции агрегирования не должен содержать другие функции агрегирования, кроме внутреннего агрегирования, содержащего классификатор **TOTAL**. Для получения более расширенных вложенных агрегирований необходимо использовать расширенную функцию **Aggr** вместе с указанным измерением.

Текстовые значения, значения NULL и отсутствующие значения в какой-либо или обеих частях пары значений приводят к игнорированию всей пары значений.

См. также:

Примеры использования функций linest (page 350)
Avg – функция диаграммы (page 290)

LINEST_SSREG

LINEST_SSREG() возвращает агрегированную остаточную сумму квадратов линейной регрессии, определенной уравнением y=mx+b для серии координат, представленных парными числами в выражениях x-expression и y-expression, повторяемых в нескольких записях так, как это определено предложением **group by**.

Синтаксис:

```
LINEST_SSREG (y_value, x_value[, y0 [, x0 ]])
```

Возвращаемые типы данных: числовое значение

Аргументы:

Аргументы

Аргумент	Описание
y_value	Выражение или поле, содержащее диапазон значений y для измерения.
x_value	Выражение или поле, содержащее диапазон значений x для измерения.
y(0), x(0)	Дополнительное значение y0 можно указать путем принудительного прохождения линии регрессии через ось y в определенной точке. Указав y0 и x0, можно задать принудительное прохождение линии регрессии через одиночную фиксированную координату. Если значения y0 и x0 не указаны, для вычисления функции требуются хотя бы две допустимые пары значений. Если y0 и x0 указаны, используется одна пара значений.

Ограничения:

Текстовые значения, значения NULL и отсутствующие значения в какой-либо или обеих частях пары значений приводят к игнорированию всей пары значений.

См. также:

Примеры использования функций linest (page 350)

LINEST_SSREG – функция диаграммы

Функция **LINEST_SSREG()** возвращает агрегированную сумму регрессии площадей линейной регрессии, определенной уравнением y=mx+b для серии координат, представленных парными числами в выражениях **x_value** и **y_value**, повторяемых в измерениях диаграммы.

Синтаксис:

```
LINEST_SSREG([{SetExpression}] [DISTINCT] [TOTAL [<fld{, fld}>]] y_value, x_value[, y0_const[, x0_const]])
```

Возвращаемые типы данных: числовое значение

Аргументы:

Аргументы

Аргумент	Описание
y_value	Выражение или поле, содержащее диапазон значений y для измерения.
x_value	Выражение или поле, содержащее диапазон значений x для измерения.
y0, x0	Дополнительное значение y0 можно указать путем принудительного прохождения линии регрессии через ось y в определенной точке. Указав y0 и x0, можно задать принудительное прохождение линии регрессии через одиночную фиксированную координату. *Если значения y0 и x0 не указаны, для вычисления функции требуются хотя бы две допустимые пары значений. Если y0 и x0 указаны, используется одна пара значений.*
SetExpression	По умолчанию функция агрегирования агрегирует множество возможных записей, определенных выборкой. Альтернативный набор записей может быть определен выражением анализа множества.
DISTINCT	Если слово **DISTINCT** указывается до аргументов функции, все дубликаты, возникшие в результате оценки аргументов функции, будут проигнорированы.

Аргумент	Описание
TOTAL	Если слово **TOTAL** стоит перед аргументами функции, вычисление выполняется по всем возможным значениям, указанным в текущих выборках, а не только в тех, которые относятся к значению текущего измерения, т. е. измерения диаграммы игнорируются. При использовании выражения **TOTAL [<fld {.fld}>]**, где префикс **TOTAL** предшествует списку из одного или нескольких имен полей, выступающих в качестве подмножества переменных измерения диаграммы, создается подмножество всех возможных значений.

Ограничения:

Параметр функции агрегирования не должен содержать другие функции агрегирования, кроме внутреннего агрегирования, содержащего классификатор **TOTAL**. Для получения более расширенных вложенных агрегирований необходимо использовать расширенную функцию **Aggr** вместе с указанным измерением.

Текстовые значения, значения NULL и отсутствующие значения в какой-либо или обеих частях пары значений приводят к игнорированию всей пары значений.

См. также:

- *Примеры использования функций linest (page 350)*
- *Avg – функция диаграммы (page 290)*

LINEST_SSRESID

Функция **LINEST_SSRESID()** возвращает агрегированную остаточную сумму квадратов линейной регрессии, определенной уравнением y=mx+b для серии координат, представленных парными числами в выражениях x-expression и y-expression, повторяемых в нескольких записях так, как это определено предложением **group by**.

Синтаксис:

```
LINEST_SSRESID (y_value, x_value[, y0 [, x0 ]])
```

Возвращаемые типы данных: числовое значение

Аргументы:

Аргументы

Аргумент	Описание
y_value	Выражение или поле, содержащее диапазон значений y для измерения.

Аргумент	Описание
x_value	Выражение или поле, содержащее диапазон значений x для измерения.
y(0), x(0)	Дополнительное значение y0 можно указать путем принудительного прохождения линии регрессии через ось y в определенной точке. Указав y0 и x0, можно задать принудительное прохождение линии регрессии через одиночную фиксированную координату. Если значения y0 и x0 не указаны, для вычисления функции требуются хотя бы две допустимые пары значений. Если y0 и x0 указаны, используется одна пара значений.

Ограничения:

Текстовые значения, значения NULL и отсутствующие значения в какой-либо или обеих частях пары значений приводят к игнорированию всей пары значений.

См. также:

Примеры использования функций linest (page 350)

LINEST_SSRESID – функция диаграммы

Функция **LINEST_SSRESID()** возвращает агрегированную остаточную сумму площадей линейной регрессии, определенной уравнением y=mx+b для серии координат, представленных парными числами в выражениях **x_value** и **y_value**, повторяемых в измерениях диаграммы.

Синтаксис:

```
LINEST_SSRESID([{SetExpression}] [DISTINCT] [TOTAL [<fld{, fld}>]] y_value,
x_value[, y0_const[, x0_const]])
```

Возвращаемые типы данных: числовое значение

Аргументы:

Аргументы

Аргумент	Описание
y_value	Выражение или поле, содержащее диапазон значений y для измерения.
x_value	Выражение или поле, содержащее диапазон значений x для измерения.

Аргумент	Описание
y0, x0	Дополнительное значение y0 можно указать путем принудительного прохождения линии регрессии через ось y в определенной точке. Указав y0 и x0, можно задать принудительное прохождение линии регрессии через одиночную фиксированную координату. *Если значения y0 и x0 не указаны, для вычисления функции требуются хотя бы две допустимые пары значений. Если y0 и x0 указаны, используется одна пара значений.*
SetExpression	По умолчанию функция агрегирования агрегирует множество возможных записей, определенных выборкой. Альтернативный набор записей может быть определен выражением анализа множества.
DISTINCT	Если слово **DISTINCT** указывается до аргументов функции, все дубликаты, возникшие в результате оценки аргументов функции, будут проигнорированы.
TOTAL	Если слово **TOTAL** стоит перед аргументами функции, вычисление выполняется по всем возможным значениям, указанным в текущих выборках, а не только в тех, которые относятся к значению текущего измерения, т. е. измерения диаграммы игнорируются. При использовании выражения **TOTAL [<fld {.fld}>]**, где префикс **TOTAL** предшествует списку из одного или нескольких имен полей, выступающих в качестве подмножества переменных измерения диаграммы, создается подмножество всех возможных значений.

Дополнительное значение y0 можно указать путем принудительного прохождения линии регрессии через ось y в определенной точке. Указав y0 и x0, можно задать принудительное прохождение линии регрессии через одиночную фиксированную координату.

Ограничения:

Параметр функции агрегирования не должен содержать другие функции агрегирования, кроме внутреннего агрегирования, содержащего классификатор **TOTAL**. Для получения более расширенных вложенных агрегирований необходимо использовать расширенную функцию **Aggr** вместе с указанным измерением.

Текстовые значения, значения NULL и отсутствующие значения в какой-либо или обеих частях пары значений приводят к игнорированию всей пары значений.

См. также:

- *Примеры использования функций linest (page 350)*
- *Avg – функция диаграммы (page 290)*

Median

Функция **Median()** возвращает среднее значение агрегированных данных в выражении в нескольких записях, как это определено предложением **group by**.

Синтаксис:

```
Median (expr)
```

Возвращаемые типы данных: числовое значение

Аргументы:

Аргументы

Аргумент	Описание
expr	Выражение или поле, содержащее данные для измерения.

Пример: Выражение скрипта с использованием медианы

Пример: выражение скрипта

Скрипт загрузки

Загрузите следующие встроенные данные и выражение скрипта в редакторе загрузки данных, чтобы создать данный пример.

```
Table 1: Load RecNo() as RowNo, Letter, Number Inline [Letter, Number A,1 A,3 A,4 A,9 B,2 B,8
B,9];                                    Median: LOAD Letter, Median(Number) as MyMedian Resident Table1 Grou
```

Создать визуализацию

На листе Qlik Sense создайте визуализацию таблицы с измерениями **Letter** и **MyMedian**.

Результат

Letter	MyMedian
A	3.5
B	8

Объяснение

Медиана считается «средним» числом, когда числа сортируются в порядке от наименьшего до наибольшего. Если набор данных содержит четное число значений, функция возвращает среднее значение двух средних чисел. В данном примере медиана вычисляется для каждого набора значений **A** и **B**, то есть 3,5 и 8, соответственно.

Median — функция диаграммы

Функция **Median()** возвращает значение медианы диапазона значений, агрегированных в выражении, повторяемом в измерениях диаграммы.

Синтаксис:

```
Median([{SetExpression}] [DISTINCT] [TOTAL [<fld{, fld}>]] expr)
```

Возвращаемые типы данных: числовое значение

Аргументы:

Аргументы

Аргумент	Описание
expr	Выражение или поле, содержащее данные для измерения.
SetExpression	По умолчанию функция агрегирования агрегирует множество возможных записей, определенных выборкой. Альтернативный набор записей может быть определен выражением анализа множества.
DISTINCT	Если слово **DISTINCT** указывается до аргументов функции, все дубликаты, возникшие в результате оценки аргументов функции, будут проигнорированы.
TOTAL	Если слово **TOTAL** стоит перед аргументами функции, вычисление выполняется по всем возможным значениям, указанным в текущих выборках, а не только в тех, которые относятся к значению текущего измерения, т. е. измерения диаграммы игнорируются. При использовании выражения **TOTAL [<fld {.fld}>]**, где префикс **TOTAL** предшествует списку из одного или нескольких имен полей, выступающих в качестве подмножества переменных измерения диаграммы, создается подмножество всех возможных значений.

Ограничения:

Параметр функции агрегирования не должен содержать другие функции агрегирования, кроме внутреннего агрегирования, содержащего классификатор **TOTAL**. Для получения более расширенных вложенных агрегирований необходимо использовать расширенную функцию **Aggr** вместе с указанным измерением.

Пример: Выражение диаграммы с использованием медианы

Пример: выражение диаграммы

Скрипт загрузки

Загрузите следующие данные через встроенную загрузку в редакторе загрузки данных, чтобы создать пример с выражениями диаграммы, показанный ниже.

```
Load RecNo() as RowNo, Letter, Number Inline [Letter, Number A,1 A,3 A,4 A,9 B,2 B,8 B,9];
```

Создать визуализацию

Создайте на листе Qlik Sense визуализацию таблицы с измерением **Letter**.

Выражение диаграммы

Добавьте в таблицу следующее выражение в качестве меры:

```
Median(Number)
```

Результат

Letter	Median(Number)
Totals	4
A	3.5
B	8

Объяснение

Медиана считается «средним» числом, когда числа сортируются в порядке от наименьшего до наибольшего. Если набор данных содержит четное число значений, функция возвращает среднее значение двух средних чисел. В данном примере медиана вычисляется для каждого набора значений **A** и **B**, то есть 3,5 и 8, соответственно.

Медиана для **Totals** вычисляется на основе всех значений и равна 4.

См. также:

Avg – функция диаграммы (page 290)

MutualInfo – функция диаграммы

MutualInfo рассчитывает взаимную информацию (mutual information, MI) между двумя полями или агрегированными значениями в **Aggr()**.

MutualInfo возвращает агрегированную взаимную информацию для двух наборов данных. Это позволяет провести анализ ключевых факторов между полем и потенциальным фактором. Взаимная информация (MI) измеряет отношение между наборами данных. Она агрегирована для пар значений (x,y), повторяемых в измерениях диаграммы. Взаимная информация измеряется в диапазоне от 0 до 1 и может быть представлена в виде значения процентиля. **MutualInfo** определяется выборками или выражением множества.

MutualInfo позволяет выполнять различные виды анализа MI:

- Попарный MI: рассчитывается MI между полем фактора и целевым полем.
- Разбивка фактора по значению: MI рассчитывается между индивидуальными значениями в поле фактора и целевом поле.
- Выбор функции: используйте **MutualInfo** на сетчатой диаграмме для создания матрицы, где все поля сравниваются друг с другом на основе MI.

MutualInfo не обязательно указывает на причинно-следственную связь между полями, обменивающимися взаимной информацией. Два поля могут иметь взаимную информацию, но не могут быть равноценными факторами друг для друга. Например, при сравнении продаж мороженого и температуры наружного воздуха **MutualInfo** покажет взаимную информацию между ними. Он не покажет, является ли температура наружного воздуха фактором, определяющим продажи мороженого (что вполне вероятно), или продажи мороженого определяют температуру наружного воздуха (что маловероятно).

При расчете взаимной информации связи затрагивают соотношение между значениями из полей различных таблиц и частоту этих значений.

Возвращаемые значения для одних и тех же полей или выборок могут незначительно отличаться. Причина этого в том, что каждый вызов **MutualInfo** использует случайным образом выбранный образец и свойственную случайность алгоритма **MutualInfo**.

MutualInfo может быть применена к функции **Aggr()**.

Синтаксис:

```
MutualInfo({SetExpression}] [DISTINCT] [TOTAL] field1, field2 , datatype [, breakdownbyvalue [, samplesize ]])
```

Возвращаемые типы данных: числовое значение

Аргументы:

Аргументы

Аргумент	Описание
field1, field2	Выражения или поля, содержащие два образца множеств, для которых необходимо измерить взаимную информацию.
datatype	Типы данных, содержащиеся в целевом поле и в поле фактора: 1 или 'dd' для комбинации «дискретный:дискретный» 2 или 'cc' для комбинации «непрерывный:непрерывный» 3 или 'cd' для комбинации «непрерывный:дискретный» 4 или 'dc' для комбинации «дискретный:непрерывный» Типы данных не чувствительны к регистру.
breakdownbyvalue	Статическое значение, соответствующее значению в поле фактора. Если указано, рассчитывается доля MI в этом значении. Можно использовать **ValueList()** или **ValueLoop()**. Если добавляется **Null()**, рассчитывается общее значение MI для всех значений в факторе. Разбивка по значениям требует, чтобы фактор содержал дискретные данные.

Аргумент	Описание
samplesize	Количество значений для выборки из целевых полей и полей фактора. Выборка выполняется случайным образом. **MutualInfo** требует минимального объема выборки 80. По умолчанию **MutualInfo** делает выборку только до 10 000 пар данных, поэтому **MutualInfo** может быть ресурсоемким. Можно указать больше пар данных в объеме выборки. В случае тайм-аута **MutualInfo** уменьшите объем выборки.
SetExpression	По умолчанию функция агрегирования агрегирует множество возможных записей, определенных выборкой. Альтернативный набор записей может быть определен выражением анализа множества.
DISTINCT	Если слово **DISTINCT** указывается до аргументов функции, все дубликаты, возникшие в результате оценки аргументов функции, будут проигнорированы.
TOTAL	Если слово **TOTAL** стоит перед аргументами функции, вычисление выполняется по всем возможным значениям, указанным в текущих выборках, а не только в тех, которые относятся к значению текущего измерения, т. е. измерения диаграммы игнорируются. При использовании выражения **TOTAL [<fld {.fld}>]**, где префикс **TOTAL** предшествует списку из одного или нескольких имен полей, выступающих в качестве подмножества переменных измерения диаграммы, создается подмножество всех возможных значений.

Ограничения:

Текстовые значения, значения NULL и отсутствующие значения в какой-либо или обеих частях пары значений приводят к игнорированию всей пары значений.

Примеры и результаты:

Добавьте образец скрипта в свое приложение и запустите. Чтобы увидеть результаты, добавьте поля, перечисленные в столбце результатов, в лист приложения.

Примеры функции

Пример	Результат
`mutualinfo (Age, Salary, 1)`	Для таблицы, включающей измерение `Employee name` и меру `mutualinfo(Age, Salary, 1)`, результат будет 0.99820986. Результат отображается только для итоговой ячейки.
`mutualinfo (TOTAL Age, Salary, 1, null(), 81)`	При создании фильтра с измерением Gender и выборками из него полученный результат составит 0,99805677, если выбран элемент Female, и 0,99847373, если выбран элемент Male. Это обусловлено тем, что выборка исключает все результаты, которые не принадлежат другому значению элемента Gender.

Пример	Результат
`mutualinfo (TOTAL Age, Gender, 1, ValueLoop (25,35))`	0.68196996. Выбор любого значения из Gender изменит это значение на 0.
`mutualinfo ({1} TOTAL Age, Salary, 1, null())`	0.99820986. Независимо от выборок. Выражение множества {1} игнорирует все выборки и измерения.

Данные, используемые в примерах:

```
Salary:

LOAD * inline [

"Employee name"|Age|Gender|Salary

Aiden Charles|20|Male|25000

Ann Lindquist|69|Female|58000

Anna Johansen|37|Female|36000

Anna Karlsson|42|Female|23000

Antonio Garcia|20|Male|61000

Benjamin Smith|42|Male|27000

Bill Yang|49|Male|50000

Binh Protzmann|69|Male|21000

Bob Park|51|Male|54000

Brenda Davies|25|Male|32000

Celine Gagnon|48|Female|38000

Cezar Sandu|50|Male|46000

Charles Ingvar Jönsson|27|Male|58000

Charlotte Edberg|45|Female|56000

Cindy Lynn|69|Female|28000

Clark Wayne|63|Male|31000

Daroush Ferrara|31|Male|29000
```

```
David Cooper|37|Male|64000

David Leg|58|Male|57000

Eunice Goldblum|31|Female|32000

Freddy Halvorsen|25|Male|26000

Gauri Indu|36|Female|46000

George van Zaant|59|Male|47000

Glenn Brown|58|Male|40000

Harry Jones|38|Male|40000

Helen Brolin|52|Female|66000

Hiroshi Ito|24|Male|42000

Ian Underwood|40|Male|45000

Ingrid Hendrix|63|Female|27000

Ira Baumel|39|Female|39000

Jackie Kingsley|23|Female|28000

Jennica Williams|36|Female|48000

Jerry Tessel|31|Male|57000

Jim Bond|50|Male|58000

Joan Callins|60|Female|65000

Joan Cleaves|25|Female|61000

Joe Cheng|61|Male|41000

John Doe|36|Male|59000

John Lemon|43|Male|21000

Karen Helmkey|54|Female|25000

Karl Berger|38|Male|68000

Karl Straubaum|30|Male|40000

Kaya Alpan|32|Female|60000

Kenneth Finley|21|Male|25000
```

```
Leif Shine|63|Male|70000

Lennart Skoglund|63|Male|24000

Leona Korhonen|46|Female|50000

Lina André|50|Female|65000

Louis Presley|29|Male|36000

Luke Langston|50|Male|63000

Marcus Salvatori|31|Male|46000

Marie Simon|57|Female|23000

Mario Rossi|39|Male|62000

Markus Danzig|26|Male|48000

Michael Carlen|21|Male|45000

Michelle Tyson|44|Female|69000

Mike Ashkenaz|45|Male|68000

Miro Ito|40|Male|39000

Nina Mihn|62|Female|57000

Olivia Nguyen|35|Female|51000

Olivier Simenon|44|Male|31000

Östen Ärlig|68|Male|57000

Pamala Garcia|69|Female|29000

Paolo Romano|34|Male|45000

Pat Taylor|67|Female|69000

Paul Dupont|34|Male|38000

Peter Smith|56|Male|53000

Pierre Clouseau|21|Male|37000

Preben Jørgensen|35|Male|38000

Rey Jones|65|Female|20000

Ricardo Gucci|55|Male|65000
```

```
Richard Ranieri|30|Male|64000

Rob Carsson|46|Male|54000

Rolf Wesenlund|25|Male|51000

Ronaldo Costa|64|Male|39000

Sabrina Richards|57|Female|40000

Sato Hiromu|35|Male|21000

Sehoon Daw|57|Male|24000

Stefan Lind|67|Male|35000

Steve Cioazzi|58|Male|23000

Sunil Gupta|45|Male|40000

Sven Svensson|45|Male|55000

Tom Lindwall|46|Male|24000

Tomas Nilsson|27|Male|22000

Trinity Rizzo|52|Female|48000

Vanessa Lambert|54|Female|27000

] (delimiter is '|');
```

Skew

Функция **Skew()** возвращает асимметрию выражения в нескольких записях, как это определено предложением **group by**.

Синтаксис:

```
Skew([ distinct] expr)
```

Возвращаемые типы данных: числовое значение

Аргументы:

Аргументы

Аргумент	Описание
expr	Выражение или поле, содержащее данные для измерения.
DISTINCT	Если слово **distinct** указано перед выражением, все дубликаты будут проигнорированы.

Примеры и результаты:

Добавьте образец скрипта в свое приложение и запустите. Затем создайте прямую таблицу, используя `Type` и *MySkew* в качестве измерений.

Результирующие данные

Пример	Результат

`Table1:`
`crosstable LOAD recno() as ID, *`
`inline [`
`Observation|Comparison`
`35|2`
`40|27`
`12|38`
`15|31`
`21|1`
`14|19`
`46|1`
`10|34`
`28|3`
`48|1`
`16|2`
`30|3`
`32|2`
`48|1`
`31|2`
`22|1`
`12|3`
`39|29`
`19|37`
`25|2 ] (delimiter is '|');`
 	
`Skew1:`
`LOAD Type,`
`Skew(Value) as MySkew`
`Resident Table1 Group By Type;`
	Результаты вычисления Skew() выглядят следующим образом:
	• `Type` — `MySkew`
	• `Comparison` — 0.86414768
	• `Observation` — 0.32625351

Skew – функция диаграммы

Функция **Skew()** возвращает агрегированную асимметрию значений выражения или поля, повторяемых в измерениях диаграммы.

Синтаксис:

```
Skew([{SetExpression}] [DISTINCT] [TOTAL [<fld{, fld}>]] expr)
```

Возвращаемые типы данных: числовое значение

Аргументы:

Аргументы

Аргумент	Описание
expr	Выражение или поле, содержащее данные для измерения.
SetExpression	По умолчанию функция агрегирования агрегирует множество возможных записей, определенных выборкой. Альтернативный набор записей может быть определен выражением анализа множества.
DISTINCT	Если слово **DISTINCT** указывается до аргументов функции, все дубликаты, возникшие в результате оценки аргументов функции, будут проигнорированы.
TOTAL	Если слово **TOTAL** стоит перед аргументами функции, вычисление выполняется по всем возможным значениям, указанным в текущих выборках, а не только в тех, которые относятся к значению текущего измерения, т. е. измерения диаграммы игнорируются. При использовании выражения **TOTAL [<fld {.fld}>]**, где префикс **TOTAL** предшествует списку из одного или нескольких имен полей, выступающих в качестве подмножества переменных измерения диаграммы, создается подмножество всех возможных значений.

Ограничения:

Параметр функции агрегирования не должен содержать другие функции агрегирования, кроме внутреннего агрегирования, содержащего классификатор **TOTAL**. Для получения более расширенных вложенных агрегирований необходимо использовать расширенную функцию **Aggr** вместе с указанным измерением.

Примеры и результаты:

Добавьте образец скрипта в свое приложение и запустите. Затем создайте прямую таблицу. установив измерение `Type` и меру `Skew(Value)`.

Необходимо активировать элемент `Totals` в свойствах таблицы.

Пример	Результат
Table1: crosstable LOAD recno() as ID, * inline [Observation|Comparison 35|2 40|27 12|38 15|31 21|1 14|19 46|1 10|34 28|3 48|1 16|2 30|3 32|2 48|1 31|2 22|1 12|3 39|29 19|37 25|2] (delimiter is '|');	Результаты вычисления Skew(Value) выглядят следующим образом: • Total – 0.23522195 • Comparison – 0.86414768 • Observation – 0.32625351

См. также:

Avg – функция диаграммы (page 290)

Stdev

Функция **Stdev()** возвращает стандартное отклонение значений в выражении в нескольких записях, как это определено предложением **group by**.

Синтаксис:

```
Stdev([distinct] expr)
```

Возвращаемые типы данных: числовое значение

Аргументы:

Аргументы

Аргумент	Описание
expr	Выражение или поле, содержащее данные для измерения.
distinct	Если слово **distinct** указано перед выражением, все дубликаты будут проигнорированы.

Примеры и результаты:

Добавьте образец скрипта в свое приложение и запустите. Затем создайте прямую таблицу, используя `Type` и `MyStdev` в качестве измерений.

Результирующие данные

Пример	Результат

`Table1:`
`crosstable LOAD recno() as ID, *`
`inline [`
`Observation|Comparison`
`35|2`
`40|27`
`12|38`
`15|31`
`21|1`
`14|19`
`46|1`
`10|34`
`28|3`
`48|1`
`16|2`
`30|3`
`32|2`
`48|1`
`31|2`
`22|1`
`12|3`
`39|29`
`19|37`
`25|2 ] (delimiter is '|');`
 	
`Stdev1:`
`LOAD Type,`
`Stdev(Value) as MyStdev`
`Resident Table1 Group By Type;`
	Результаты вычисления Stdev() выглядят следующим образом:
	• `Type` — `MyStdev`
	• `Comparison` — 14.61245
	• `Observation` — 12.507997

Stdev – функция диаграммы

Функция **Stdev()** находит стандартное отклонение диапазона данных, агрегированных в выражении или поле, повторяемых в измерениях диаграммы.

Синтаксис:

```
Stdev([{SetExpression}] [DISTINCT] [TOTAL [<fld{, fld}>]] expr)
```

Возвращаемые типы данных: числовое значение

Аргументы:

Аргументы

Аргумент	Описание
expr	Выражение или поле, содержащее данные для измерения.
SetExpression	По умолчанию функция агрегирования агрегирует множество возможных записей, определенных выборкой. Альтернативный набор записей может быть определен выражением анализа множества.
DISTINCT	Если слово **DISTINCT** указывается до аргументов функции, все дубликаты, возникшие в результате оценки аргументов функции, будут проигнорированы.
TOTAL	Если слово **TOTAL** стоит перед аргументами функции, вычисление выполняется по всем возможным значениям, указанным в текущих выборках, а не только в тех, которые относятся к значению текущего измерения, т. е. измерения диаграммы игнорируются. При использовании выражения **TOTAL [<fld {,fld}>]**, где префикс **TOTAL** предшествует списку из одного или нескольких имен полей, выступающих в качестве подмножества переменных измерения диаграммы, создается подмножество всех возможных значений.

Ограничения:

Параметр функции агрегирования не должен содержать другие функции агрегирования, кроме внутреннего агрегирования, содержащего классификатор **TOTAL**. Для получения более расширенных вложенных агрегирований необходимо использовать расширенную функцию **Aggr** вместе с указанным измерением.

Примеры и результаты:

Добавьте образец скрипта в свое приложение и запустите. Затем создайте прямую таблицу. установив измерение `Type` и меру `Stdev(Value)`.

Необходимо активировать элемент `Totals` в свойствах таблицы.

Пример	Результат

`Stdev(Value)`
`Table1:`
`crosstable LOAD recno()`
`as ID, * inline [`
`Observation|Comparison`
`35|2`
`40|27`
`12|38`
`15|31`
`21|1`
`14|19`
`46|1`
`10|34`
`28|3`
`48|1`
`16|2`
`30|3`
`32|2`
`48|1`
`31|2`
`22|1`
`12|3`
`39|29`
`19|37`
`25|2 ] (delimiter is`
`'|');`
	Результаты вычисления Stdev(Value) выглядят следующим образом:
	• `Total` – 15.47529
	• `Comparison` – 14.61245
	• `Observation` – 12.507997

См. также:

- *Avg – функция диаграммы (page 290)*
- *STEYX – функция диаграммы (page 348)*

Sterr

Функция **Sterr()** возвращает агрегированную стандартную ошибку (stdev/sqrt(n)) для серии значений, представленных выражением, повторяемым в нескольких записях так, как это определено предложением **group by**.

Синтаксис:

```
Sterr ([distinct] expr)
```

Возвращаемые типы данных: числовое значение

Аргументы:

Аргументы

Аргумент	Описание
expr	Выражение или поле, содержащее данные для измерения.
distinct	Если слово **distinct** указано перед выражением, все дубликаты будут проигнорированы.

Ограничения:

Текстовые значения, значения NULL и отсутствующие значения игнорируются.

Примеры и результаты:

Добавьте образец скрипта в свое приложение и запустите. Чтобы увидеть результаты, добавьте поля, перечисленные в столбце результатов, в лист приложения.

Результирующие данные

Пример	Результат
Table1: crosstable LOAD recno() as ID, * inline [Observation|Comparison 35|2 40|27 12|38 15|31 21|1 14|19 46|1 10|34 28|3 48|1 16|2 30|3 32|2 48|1 31|2 22|1 12|3 39|29 19|37 25|2] (delimiter is '|'); Sterr1: LOAD Type, Sterr(Value) as MySterr Resident Table1 Group By Type;	В таблице с измерениями `Type` и `MySterr` результаты вычисления Sterr() в скрипте загрузки данных будут показаны как: Type MySterr Comparison 3.2674431 Observation 2.7968733

Sterr – функция диаграммы

Функция **Sterr()** находит значение стандартной ошибки среднего значения (stdev/sqrt(n)) для серии значений, агрегированных в выражении, повторяемом в измерениях диаграммы.

Синтаксис:

```
Sterr([{SetExpression}] [DISTINCT] [TOTAL [<fld{, fld}>]] expr)
```

Возвращаемые типы данных: числовое значение

Аргументы:

Аргументы

Аргумент	Описание
expr	Выражение или поле, содержащее данные для измерения.
SetExpression	По умолчанию функция агрегирования агрегирует множество возможных записей, определенных выборкой. Альтернативный набор записей может быть определен выражением анализа множества.
DISTINCT	Если слово **DISTINCT** указывается до аргументов функции, все дубликаты, возникшие в результате оценки аргументов функции, будут проигнорированы.
TOTAL	Если слово **TOTAL** стоит перед аргументами функции, вычисление выполняется по всем возможным значениям, указанным в текущих выборках, а не только в тех, которые относятся к значению текущего измерения, т. е. измерения диаграммы игнорируются. При использовании выражения **TOTAL [<fld {.fld}>]**, где префикс **TOTAL** предшествует списку из одного или нескольких имен полей, выступающих в качестве подмножества переменных измерения диаграммы, создается подмножество всех возможных значений.

Ограничения:

Параметр функции агрегирования не должен содержать другие функции агрегирования, кроме внутреннего агрегирования, содержащего классификатор **TOTAL**. Для получения более расширенных вложенных агрегирований необходимо использовать расширенную функцию **Aggr** вместе с указанным измерением.

Текстовые значения, значения NULL и отсутствующие значения игнорируются.

Примеры и результаты:

Добавьте образец скрипта в свое приложение и запустите. Затем создайте прямую таблицу. установив измерение `Type` и меру `Sterr(Value)`.

Необходимо активировать элемент `Totals` в свойствах таблицы.

Пример	Результат
Table1: crosstable LOAD recno() as ID, * inline [Observation|Comparison 35|2 40|27 12|38 15|31 21|1 14|19 46|1 10|34 28|3 48|1 16|2 30|3 32|2 48|1 31|2 22|1 12|3 39|29 19|37 25|2] (delimiter is '|');	Результаты вычисления Sterr(Value) выглядят следующим образом: • Total – 2.4468583 • Comparison – 3.2674431 • Observation – 2.7968733

См. также:

Avg – функция диаграммы (page 290)
STEYX – функция диаграммы (page 348)

STEYX

Функция **STEYX()** возвращает агрегированную стандартную ошибку предсказанного значения y для каждого значения x в регрессии для серии координат, представленных парными числами в выражениях x-expression и y-expression, повторяемых в нескольких записях так, как это определено предложением **group by**.

Синтаксис:

```
STEYX (y_value, x_value)
```

Возвращаемые типы данных: числовое значение

Аргументы:

Аргументы

Аргумент	Описание
y_value	Выражение или поле, содержащее диапазон значений y для измерения.
x_value	Выражение или поле, содержащее диапазон значений x для измерения.

Ограничения:

Текстовые значения, значения NULL и отсутствующие значения в какой-либо или обеих частях пары значений приводят к игнорированию всей пары значений.

Примеры и результаты:

Добавьте образец скрипта в свое приложение и запустите. Чтобы увидеть результаты, добавьте поля, перечисленные в столбце результатов, в лист приложения.

Результирующие данные

Пример	Результат

`Trend:`
`Load *, 1 as Grp;`
`LOAD * inline [`
`Month|KnownY|KnownX`
`Jan|2|6`
`Feb|3|5`
`Mar|9|11`
`Apr|6|7`
`May|8|5`
`Jun|7|4`
`Jul|5|5`
`Aug|10|8`
`Sep|9|10`
`Oct|12|14`
`Nov|15|17`
`Dec|14|16`
`] (delimiter is '|');`
 	
`STEYX1:`
`LOAD Grp,`
`STEYX(KnownY, KnownX)`
`as MySTEYX`
`Resident Trend Group`
`By Grp;`
	В таблице с измерением `MySTEYX` результат вычисления STEYX() в скрипте загрузки данных будет показан как 2,0714764.

STEYX – функция диаграммы

Функция **STEYX()** возвращает агрегированную стандартную ошибку во время предсказания значения y для каждого значения x в линейной регрессии, определенной серией координат, представленных парными числами в выражениях **y_value** и **x_value**.

Синтаксис:

```
STEYX([{SetExpression}] [DISTINCT] [TOTAL [<fld{, fld}>]] y_value, x_value)
```

Возвращаемые типы данных: числовое значение

Аргументы:

Аргументы

Аргумент	Описание
y_value	Выражение или поле, содержащее диапазон известных y-значений для измерения.
x_value	Выражение или поле, содержащее диапазон известных x-значений для измерения.
SetExpression	По умолчанию функция агрегирования агрегирует множество возможных записей, определенных выборкой. Альтернативный набор записей может быть определен выражением анализа множества.
DISTINCT	Если слово **DISTINCT** указывается до аргументов функции, все дубликаты, возникшие в результате оценки аргументов функции, будут проигнорированы.
TOTAL	Если слово **TOTAL** стоит перед аргументами функции, вычисление выполняется по всем возможным значениям, указанным в текущих выборках, а не только в тех, которые относятся к значению текущего измерения, т. е. измерения диаграммы игнорируются. При использовании выражения **TOTAL [<fld {.fld}>]**, где префикс **TOTAL** предшествует списку из одного или нескольких имен полей, выступающих в качестве подмножества переменных измерения диаграммы, создается подмножество всех возможных значений.

Ограничения:

Параметр функции агрегирования не должен содержать другие функции агрегирования, кроме внутреннего агрегирования, содержащего классификатор **TOTAL**. Для получения более расширенных вложенных агрегирований необходимо использовать расширенную функцию **Aggr** вместе с указанным измерением.

Текстовые значения, значения NULL и отсутствующие значения в какой-либо или обеих частях пары значений приводят к игнорированию всей пары значений.

Примеры и результаты:

Добавьте образец скрипта в свое приложение и запустите. Затем создайте прямую таблицу со значениями измерения `KnownY` и `KnownX` и мерой `Steyx(KnownY,KnownX)`.

Необходимо активировать элемент `Totals` в свойствах таблицы.

Пример	Результат
Trend: LOAD * inline [Month|KnownY|KnownX Jan|2|6 Feb|3|5 Mar|9|11 Apr|6|7 May|8|5 Jun|7|4 Jul|5|5 Aug|10|8 Sep|9|10 Oct|12|14 Nov|15|17 Dec|14|16] (delimiter is '|');	Результат вычисления STEYX(KnownY,KnownX) равен 2,071 (если форматирование числа установлено на значение "3 десятичных знака".)

См. также:

- *Avg – функция диаграммы (page 290)*
- *Sterr – функция диаграммы (page 345)*

Примеры использования функций linest

Функции linest используются для обнаружения значений, связанных с анализом линейной регрессии. В этом разделе описано, как построить визуализации с помощью данных образца, чтобы найти значения функций linest, доступных в программе Qlik Sense. Функции linest можно использовать как в скрипте загрузки данных, так и в выражениях диаграммы.

Описание синтаксиса и аргументов см. в отдельных разделах о функциях linest диаграммы и скрипта.

Выражения данных и скрипта, используемые в примерах

Загрузите следующие выражения данных и скрипта через встроенную загрузку в редакторе загрузки данных, чтобы создать примеры для функции linest(), показанные ниже.

```
T1: LOAD *, 1 as Grp; LOAD * inline [ X|Y 1|0 2|1 3|3 4|8 5|14 6|20 7|0 8|50 9|25 10|60 11|38
12|19 13|26 14|143 15|98 16|27 17|59 18|78 19|158 20|279 ] (delimiter is '|');                    R1: LOAD
Grp, linest_B(Y,X) as Linest_B, linest_DF(Y,X) as Linest_DF, linest_F(Y,X) as Linest_F,
linest_M(Y,X) as Linest_M, linest_R2(Y,X) as Linest_R2, linest_SEB(Y,X,1,1) as Linest_SEB,
linest_SEM(Y,X) as Linest_SEM, linest_SEY(Y,X) as Linest_SEY, linest_SSREG(Y,X) as Linest_
SSREG, linest_SSRESID(Y,X) as Linest_SSRESID resident T1 group by Grp;
```

Пример 1. Выражения скрипта с использованием linest

Пример: Выражения скрипта

Создайте визуализацию на основе вычислений скрипта загрузки данных

На листе Qlik Sense создайте визуализацию таблицы со следующими полями в качестве столбцов:

- Linest_B
- Linest_DF
- Linest_F
- Linest_M
- Linest_R2
- Linest_SEB
- Linest_SEM
- Linest_SEY
- Linest_SSREG
- Linest_SSRESID

Результат

Таблица, содержащая результаты вычислений linest, выполненных в скрипте загрузки данных, должна выглядеть так:

Результирующая таблица

Linest_B	Linest_DF	Linest_F	Linest_M	Linest_R2	Linest_SEB
-35.047	18	20.788	8.605	0.536	22.607

Результирующая таблица

Linest_SEM	Linest_SEY	Linest_SSREG	Linest_SSRESID
1.887	48.666	49235.014	42631.186

Пример 2. Выражения диаграммы с использованием linest

Пример: Выражения диаграммы

На листе Qlik Sense создайте визуализацию таблицы со следующими полями в качестве измерений:

```
ValueList('Linest_b', 'Linest_df','Linest_f', 'Linest_m','Linest_r2','Linest_SEB','Linest_
SEM','Linest_SEY','Linest_SSREG','Linest_SSRESID')
```

Данное выражение использует функцию синтетических измерений для создания меток для измерений с именами функций linest. Для экономии места метку можно изменить на **Linest functions**.

Добавьте следующее выражение в таблицу в качестве меры:

```
Pick(Match(ValueList('Linest_b', 'Linest_df','Linest_f', 'Linest_m','Linest_r2','Linest_
SEB','Linest_SEM','Linest_SEY','Linest_SSREG','Linest_SSRESID'),'Linest_b', 'Linest_
df','Linest_f', 'Linest_m','Linest_r2','Linest_SEB','Linest_SEM','Linest_SEY','Linest_
SSREG','Linest_SSRESID'),Linest_b(Y,X),Linest_df(Y,X),Linest_f(Y,X),Linest_m(Y,X),Linest_r2
(Y,X),Linest_SEB(Y,X,1,1),Linest_SEM(Y,X),Linest_SEY(Y,X),Linest_SSREG(Y,X),Linest_SSRESID
(Y,X) )
```

Данное выражение отображает значение результата каждой функции linest напротив соответствующего имени в синтетическом измерении. Результат функции `Linest_b(Y,X)` отображается рядом с **linest_b** и так далее.

Результат

Результирующая таблица

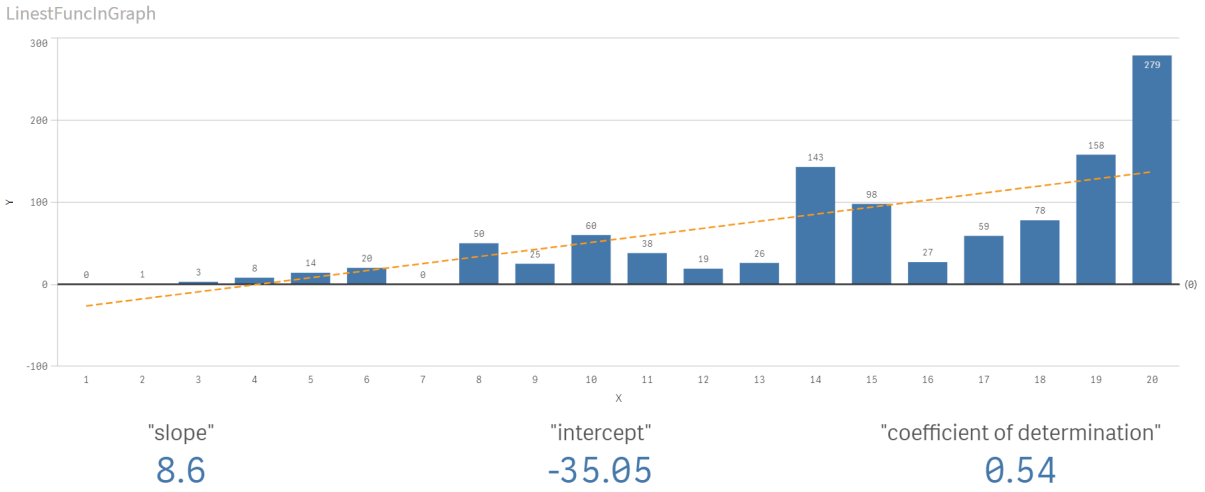
Linest functions	Linest function results
Linest_b	-35.047
Linest_df	18
Linest_f	20.788
Linest_m	8.605
Linest_r2	0.536
Linest_SEB	22.607
Linest_SEM	1.887
Linest_SEY	48.666
Linest_SSREG	49235.014
Linest_SSRESID	42631.186

Пример 3: Выражения диаграммы с использованием linest

Пример: Выражения диаграммы

1. Создайте на листе Qlik Sense визуализацию линейчатой диаграммы, используя измерение **X** и меру **Y**.
2. Добавьте линию линейного тренда в меру Y.
3. Добавьте на лист визуализацию ключевого показателя эффективности.
    1. Добавьте *slope* (наклон) в качестве метки для ключевого показателя эффективности.
    2. Добавьте `sum(Linest_M)` в качестве выражения для ключевого показателя эффективности.
4. Добавьте на лист визуализацию второго ключевого показателя эффективности.
    1. Добавьте *intercept* (отсекаемый отрезок) в качестве метки для ключевого показателя эффективности.
    2. Добавьте `Sum(Linest_B)` в качестве выражения для ключевого показателя эффективности.
5. Добавьте на лист визуализацию третьего ключевого показателя эффективности.
    1. Добавьте *coefficient of determination* (коэффициент детерминации) в качестве метки для ключевого показателя эффективности.
    2. Добавьте `Sum(Linest_R2)` в качестве выражения для ключевого показателя эффективности.

Результат



Объяснение

На линейчатой диаграмме отображается представление данных X и Y. Релевантные функции linest () предоставляют значения для линейного уравнения регрессии, на основе которого строится линия тренда, а именно y = m * x + b. Уравнение использует метод «наименьших квадратов» для вычисления прямой линии (линии тренда), возвращая массив, описывающий линию, которая лучше всего соответствует данным.

Ключевые показатели эффективности отображают результаты функций linest() **sum(Linest_M)** для наклона и **sum(Linest_B)** для отсекаемого отрезка Y, которые представляют собой переменные в линейном уравнении регрессии, а также соответствующее агрегированное значение R2 для коэффициента детерминации.

Статистические тестовые функции

Статистические тестовые функции можно использовать как в скрипте загрузки данных, так и в выражениях диаграмм, но синтаксис имеет различия.

Функции критерия Хи-квадрат

Обычно используются при изучении качественных переменных. Можно сравнить полученные частоты в односторонней таблице частот с ожидаемыми частотами или изучить связь двух переменных в таблице вероятности.

Функции T-критериев

Функции t-критерия используются для статистического исследования двух генеральных средних. T-критерий для двух выборок проверяет, отличаются ли эти выборки. Он обычно используется, когда два обычных распределения имеют неизвестные изменения, и когда в эксперименте используется малый размер выборки.

Функции Z-критериев

Статистическое исследование двух генеральных средних. Z-критерий для двух выборок проверяет, отличаются ли две выборки. Он обычно используется, когда два обычных распределения имеют известные изменения, и когда в эксперименте используется большой размер выборки.

Функции критерия Хи-квадрат

Обычно используются при изучении качественных переменных. Можно сравнить полученные частоты в односторонней таблице частот с ожидаемыми частотами или изучить связь двух переменных в таблице вероятности.Chi-squared test functions are used to determine whether there is a statistically significant difference between the expected frequencies and the observed frequencies in one or more groups. Often a histogram is used, and the different bins are compared to an expected distribution.

Если функция используется в скрипте загрузки данных, значения повторяются в нескольких записях, как определено предложением group by.

Если функция используется в выражении диаграммы, значения повторяются в измерениях диаграммы.

Chi2Test_chi2
Функция **Chi2Test_chi2()** возвращает агрегированное значение критерия Хи-**квадрат** для одной или двух серий значений.

```
Функция Chi2Test_chi2() возвращает агрегированное значение критерия Хи-квадрат для одной или двух серий значений.(col, row, actual_value[, expected_value])
```

Chi2Test_df
Функция **Chi2Test_df()** возвращает агрегированное df-значение критерия Хи-квадрат (степени свободы) для одной или двух серий значений.

```
Функция Chi2Test_df() возвращает агрегированное df-значение критерия Хи-квадрат (степени свободы) для одной или двух серий значений.(col, row, actual_value[, expected_value])
```

Chi2Test_p
Функция **Chi2Test_p()** возвращает агрегированное p-значение критерия Хи-квадрат (важность) для одной или двух серий значений.

```
Chi2Test_p — функция диаграммы(col, row, actual_value[, expected_value])
```

См. также:

- *Функции T-критериев (page 357)*
- *Функции Z-критериев (page 394)*

Chi2Test_chi2

Функция **Chi2Test_chi2()** возвращает агрегированное значение критерия Хи-квадрат для одной или двух серий значений.

Если функция используется в скрипте загрузки данных, значения повторяются в нескольких записях, как определено предложением group by.

Если функция используется в выражении диаграммы, значения повторяются в измерениях диаграммы.

Все Qlik Sense функции chi^2-критерия имеют одинаковые аргументы.

Синтаксис:

```
Chi2Test_chi2(col, row, actual_value[, expected_value])
```

Возвращаемые типы данных: числовое значение

Аргументы:

Аргументы

Аргумент	Описание
col, row	Указанный столбец и строка в матрице значений тестируются.
actual_value	Наблюдаемое значение данных при указанных элементах **col** и **row**.
expected_value	Ожидаемое значение для распределения при указанных элементах **col** и **row**.

Ограничения:

Текстовые значения, значения NULL, а также отсутствующие значения в значении выражения приводят к тому, что функция возвращает значение NULL.

Примеры:

```
Chi2Test_chi2( Grp, Grade, Count )
Chi2Test_chi2( Gender, Description, Observed, Expected )
```

См. также:

- *Примеры использования функций chi2-test в диаграммах (page 410)*
- *Примеры использования функций chi2-test в скрипте загрузки данных (page 413)*

Chi2Test_df

Функция **Chi2Test_df()** возвращает агрегированное df-значение критерия Хи-квадрат (степени свободы) для одной или двух серий значений.

Если функция используется в скрипте загрузки данных, значения повторяются в нескольких записях, как определено предложением group by.

Если функция используется в выражении диаграммы, значения повторяются в измерениях диаграммы.

Все Qlik Sense функции chi^2-критерия имеют одинаковые аргументы.

Синтаксис:

```
Chi2Test_df(col, row, actual_value[, expected_value])
```

Возвращаемые типы данных: числовое значение

Аргументы:

Аргументы

Аргумент	Описание
col, row	Указанный столбец и строка в матрице значений тестируются.
actual_value	Наблюдаемое значение данных при указанных элементах **col** и **row**.
expected_value	Ожидаемое значение для распределения при указанных элементах **col** и **row**.

Ограничения:

Текстовые значения, значения NULL, а также отсутствующие значения в значении выражения приводят к тому, что функция возвращает значение NULL.

Примеры:

```
Chi2Test_df( Grp, Grade, Count )
Chi2Test_df( Gender, Description, Observed, Expected )
```

См. также:

Примеры использования функций chi2-test в диаграммах (page 410)
Примеры использования функций chi2-test в скрипте загрузки данных (page 413)

Chi2Test_p – функция диаграммы

Функция **Chi2Test_p()** возвращает агрегированное p-значение критерия Хи-квадрат (важность) для одной или двух серий значений. Данный тест может выполняться на основе значений в тестировании **actual_value** для отклонений в указанных матрицах **col** и **row** или путем сравнения значений в элементе **actual_value** с соответствующими значениями в элементе **expected_value**, если они указаны.

Если функция используется в скрипте загрузки данных, значения повторяются в нескольких записях, как определено предложением group by.

Если функция используется в выражении диаграммы, значения повторяются в измерениях диаграммы.

Все Qlik Sense функции chi^2-критерия имеют одинаковые аргументы.

Синтаксис:

```
Chi2Test_p(col, row, actual_value[, expected_value])
```

Возвращаемые типы данных: числовое значение

Аргументы:

Аргументы

Аргумент	Описание
col, row	Указанный столбец и строка в матрице значений тестируются.
actual_value	Наблюдаемое значение данных при указанных элементах **col** и **row**.
expected_value	Ожидаемое значение для распределения при указанных элементах **col** и **row**.

Ограничения:

Текстовые значения, значения NULL, а также отсутствующие значения в значении выражения приводят к тому, что функция возвращает значение NULL.

Примеры:

```
Chi2Test_p( Grp, Grade, Count )
Chi2Test_p( Gender, Description, Observed, Expected )
```

См. также:

Примеры использования функций chi2-test в диаграммах (page 410)
Примеры использования функций chi2-test в скрипте загрузки данных (page 413)

Функции T-критериев

Функции t-критерия используются для статистического исследования двух генеральных средних. T-критерий для двух выборок проверяет, отличаются ли эти выборки. Он обычно используется, когда два обычных распределения имеют неизвестные изменения, и когда в эксперименте используется малый размер выборки.

В следующих разделах функции статистического теста t-критерия сгруппированы согласно образцу критерия Стьюдента, применяемого к каждому типу функции.

Создание типичного отчета t-test (page 415)

T-критерии для двух независимых выборок

Следующие функции применяются к t-критериям Стьюдента для двух независимых выборок.

ttest_conf

Функция **TTest_conf** возвращает агрегированное значение доверительного интервала t-критерия для двух независимых выборок.

```
Функция TTest_conf возвращает агрегированное значение доверительного
интервала t-критерия для двух независимых выборок. ( grp, value [, sig[, eq_
var]])
```

ttest_df

Функция **TTest_df()** возвращает агрегированное значение t-критерия Стьюдента (степени свободы) для двух независимых серий значений.

```
Функция TTest_df() возвращает агрегированное значение t-критерия Стьюдента
(степени свободы) для двух независимых серий значений. (grp, value [, eq_var)
```

ttest_dif

Функция **TTest_dif()** – это числовая функция, которая возвращает агрегированное среднее значение разницы t-критерия Стьюдента для двух независимых серий значений.

```
Функция TTest_dif() — это числовая функция, которая возвращает
агрегированное среднее значение разницы t-критерия Стьюдента для двух
независимых серий значений. (grp, value)
```

ttest_lower

Функция **TTest_lower()** возвращает агрегированное значение нижнего предела доверительного интервала для двух независимых серий значений.

```
Функция TTest_lower() возвращает агрегированное значение нижнего предела
доверительного интервала для двух независимых серий значений. (grp, value [,
sig[, eq_var]])
```

ttest_sig

Функция **TTest_sig()** возвращает агрегированное значение двухвостого уровня важности t-критерия Стьюдента для двух независимых серий значений.

```
Функция TTest_sig() возвращает агрегированное значение двухвостого уровня
важности t-критерия Стьюдента для двух независимых серий значений. (grp,
value [, eq_var])
```

ttest_sterr

Функция **TTest_sterr()** возвращает агрегированное среднее значение разницы стандартной ошибки t-критерия Стьюдента для двух независимых серий значений.

```
Функция TTest_sterr() возвращает агрегированное среднее значение разницы стандартной ошибки t-критерия Стьюдента для двух независимых серий значений. (grp, value [, eq_var])
```

ttest_t

Функция **TTest_t()** возвращает агрегированное t-значение для двух независимых серий значений.

```
Функция TTest_t() возвращает агрегированное t-значение для двух независимых серий значений. (grp, value [, eq_var])
```

ttest_upper

Функция **TTest_upper()** возвращает агрегированное значение верхнего предела доверительного интервала для двух независимых серий значений.

```
Функция TTest_upper() возвращает агрегированное значение верхнего предела доверительного интервала для двух независимых серий значений. (grp, value [, sig [, eq_var]])
```

T-критерии для двух независимых взвешенных выборок

Следующие функции применяются к t-критериям Стьюдента двух независимых выборок, где серия вводимых данных дается во взвешенном формате двух столбцов.

ttestw_conf

Функция **TTestw_conf()** возвращает агрегированное t-значение для двух независимых серий значений.

```
Функция TTestw_conf() возвращает агрегированное t-значение для двух независимых серий значений. (weight, grp, value [, sig[, eq_var]])
```

ttestw_df

Функция **TTestw_df()** возвращает агрегированное df-значение t-критерия Стьюдента (степени свободы) для двух независимых серий значений.

```
Функция TTestw_df() возвращает агрегированное df-значение t-критерия Стьюдента (степени свободы) для двух независимых серий значений. (weight, grp, value [, eq_var])
```

ttestw_dif

Функция **TTestw_dif()** возвращает агрегированное среднее значение разницы t-критерия Стьюдента для двух независимых серий значений.

```
Функция TTestw_dif() возвращает агрегированное среднее значение разницы t-критерия Стьюдента для двух независимых серий значений. ( weight, grp, value)
```

ttestw_lower

Функция **TTestw_lower()** возвращает агрегированное значение нижнего предела доверительного интервала для двух независимых серий значений.

```
Функция TTestw_lower() возвращает агрегированное значение нижнего предела доверительного интервала для двух независимых серий значений. (weight, grp, value [, sig[, eq_var]])
```

ttestw_sig

Функция **TTestw_sig()** возвращает агрегированное значение двухвостого уровня важности t-критерия Стьюдента для двух независимых серий значений.

```
Функция TTestw_sig() возвращает агрегированное значение двухвостого уровня важности t-критерия Стьюдента для двух независимых серий значений. ( weight, grp, value [, eq_var])
```

ttestw_sterr

Функция **TTestw_sterr()** возвращает агрегированное среднее значение разницы стандартной ошибки t-критерия Стьюдента для двух независимых серий значений.

```
Функция TTestw_sterr() возвращает агрегированное среднее значение разницы стандартной ошибки t-критерия Стьюдента для двух независимых серий значений. (weight, grp, value [, eq_var])
```

ttestw_t

Функция **TTestw_t()** возвращает агрегированное t-значение для двух независимых серий значений.

```
Функция TTestw_t() возвращает агрегированное t-значение для двух независимых серий значений. (weight, grp, value [, eq_var])
```

ttestw_upper

Функция **TTestw_upper()** возвращает агрегированное значение верхнего предела доверительного интервала для двух независимых серий значений.

```
Функция TTestw_upper() возвращает агрегированное значение верхнего предела доверительного интервала для двух независимых серий значений. (weight, grp, value [, sig [, eq_var]])
```

T-критерии для одной выборки

Следующие функции применяются к t-критериям Стьюдента для одной выборки.

ttest1_conf

Функция **TTest1_conf()** возвращает агрегированное значение доверительного интервала для серии значений.

```
Функция TTest1_conf() возвращает агрегированное значение доверительного интервала для серии значений. (value [, sig])
```

ttest1_df
Функция **TTest1_df()** возвращает агрегированное df-значение t-критерия Стьюдента (степени свободы) для серии значений.

```
Функция TTest1_df() возвращает агрегированное df-значение t-критерия Стьюдента (степени свободы) для серии значений. (value)
```

ttest1_dif
Функция **TTest1_dif()** возвращает агрегированное среднее значение разницы t-критерия Стьюдента для серии значений.

```
Функция TTest1_dif() возвращает агрегированное среднее значение разницы t-критерия Стьюдента для серии значений. (value)
```

ttest1_lower
Функция **TTest1_lower()** возвращает агрегированное значение нижнего предела доверительного интервала для серии значений.

```
Функция TTest1_lower() возвращает агрегированное значение нижнего предела доверительного интервала для серии значений. (value [, sig])
```

ttest1_sig
Функция **TTest1_sig()** возвращает агрегированное значение двухвостого уровня важности t-критерия Стьюдента для серии значений.

```
Функция TTest1_sig() возвращает агрегированное значение двухвостого уровня важности t-критерия Стьюдента для серии значений. (value)
```

ttest1_sterr
Функция **TTest1_sterr()** возвращает агрегированное среднее значение разницы стандартной ошибки t-критерия Стьюдента для серии значений.

```
Функция TTest1_sterr() возвращает агрегированное среднее значение разницы стандартной ошибки t-критерия Стьюдента для серии значений. (value)
```

ttest1_t
Функция **TTest1_t()** возвращает агрегированное t-значение для серии значений.

```
Функция TTest1_t() возвращает агрегированное t-значение для серии значений. (value)
```

ttest1_upper
Функция **TTest1_upper()** возвращает агрегированное значение верхнего предела доверительного интервала для серии значений.

```
Функция TTest1_upper() возвращает агрегированное значение верхнего предела доверительного интервала для серии значений. (value [, sig])
```

T-критерии для одной взвешенной выборки

Следующие функции применяются к t-критериям Стьюдента для одной выборки, где серия вводимых данных дается во взвешенном формате двух столбцов.

ttest1w_conf

Функция **TTest1w_conf()** – это **числовая** функция, которая возвращает агрегированное значение доверительного интервала для серии значений.

```
Функция TTest1w_conf() — это числовая функция, которая возвращает агрегированное значение доверительного интервала для серии значений. (weight, value [, sig])
```

ttest1w_df

Функция **TTest1w_df()** возвращает агрегированное df-значение t-критерия Стьюдента (степени свободы) для серии значений.

```
Функция TTest1w_df() возвращает агрегированное df-значение t-критерия Стьюдента (степени свободы) для серии значений. (weight, value)
```

ttest1w_dif

Функция **TTest1w_dif()** возвращает агрегированное среднее значение разницы t-критерия Стьюдента для серии значений.

```
Функция TTest1w_dif() возвращает агрегированное среднее значение разницы t-критерия Стьюдента для серии значений. (weight, value)
```

ttest1w_lower

Функция **TTest1w_lower()** возвращает агрегированное значение нижнего предела доверительного интервала для серии значений.

```
Функция TTest1w_lower() возвращает агрегированное значение нижнего предела доверительного интервала для серии значений. (weight, value [, sig])
```

ttest1w_sig

Функция **TTest1w_sig()** возвращает агрегированное значение двухвостого уровня важности t-критерия Стьюдента для серии значений.

```
Функция TTest1w_sig() возвращает агрегированное значение двухвостого уровня важности t-критерия Стьюдента для серии значений. (weight, value)
```

ttest1w_sterr

Функция **TTest1w_sterr()** возвращает агрегированное среднее значение разницы стандартной ошибки t-критерия Стьюдента для серии значений.

```
Функция TTest1w_sterr() возвращает агрегированное среднее значение разницы стандартной ошибки t-критерия Стьюдента для серии значений. (weight, value)
```

ttest1w_t

Функция **TTest1w_t()** возвращает агрегированное t-значение для серии значений.

```
Функция TTest1w_t() возвращает агрегированное t-значение для серии значений. ( weight, value)
```

ttest1w_upper

Функция **TTest1w_upper()** возвращает агрегированное значение верхнего предела доверительного интервала для серии значений.

```
Функция TTest1w_upper() возвращает агрегированное значение верхнего предела доверительного интервала для серии значений. (weight, value [, sig])
```

TTest_conf

Функция **TTest_conf** возвращает агрегированное значение доверительного интервала t-критерия для двух независимых выборок.

Эта функция применяется к t-критериям Стьюдента для независимых выборок.

Если функция используется в скрипте загрузки данных, значения повторяются в нескольких записях, как определено предложением group by.

Если функция используется в выражении диаграммы, значения повторяются в измерениях диаграммы.

Синтаксис:

```
TTest_conf ( grp, value [, sig [, eq_var]])
```

Возвращаемые типы данных: числовое значение

Аргументы:

Аргументы

Аргумент	Описание
value	Выборка значений для оценки. Значения выборки должны быть сгруппированы логически, как указано только двумя значениями в элементе **group**. Если имя поля для значений выборки не указано в скрипте загрузки, поле автоматически получит имя **Value**.
grp	Поле, содержащее имена каждой из двух групп с выборками. Если имя поля для группы не указано в скрипте загрузки, поле автоматически получит имя **Type**.
sig	В **sig** можно указать двусторонний уровень важности. При отсутствии значения **sig** устанавливается равным 0,025, что приводит к значению доверительного интервала 95%.
eq_var	Если значение **eq_var** определено как False (0), будут приняты отдельные изменения двух выборок. Если значение **eq_var** определено как True (1), будут приняты равные изменения в выборках.

Ограничения:

Текстовые значения, значения NULL, а также отсутствующие значения в значении выражения приводят к тому, что функция возвращает значение NULL.

Примеры:

```
TTest_conf( Group, Value )
TTest_conf( Group, Value, Sig, false )
```

См. также:

Создание типичного отчета t-test (page 415)

TTest_df

Функция **TTest_df()** возвращает агрегированное значение t-критерия Стьюдента (степени свободы) для двух независимых серий значений.

Эта функция применяется к t-критериям Стьюдента для независимых выборок.

Если функция используется в скрипте загрузки данных, значения повторяются в нескольких записях, как определено предложением group by.

Если функция используется в выражении диаграммы, значения повторяются в измерениях диаграммы.

Синтаксис:

```
TTest_df (grp, value [, eq_var])
```

Возвращаемые типы данных: числовое значение

Аргументы:

Аргументы

Аргумент	Описание
value	Выборка значений для оценки. Значения выборки должны быть сгруппированы логически, как указано только двумя значениями в элементе **group**. Если имя поля для значений выборки не указано в скрипте загрузки, поле автоматически получит имя **Value**.
grp	Поле, содержащее имена каждой из двух групп с выборками. Если имя поля для группы не указано в скрипте загрузки, поле автоматически получит имя **Type**.
eq_var	Если значение **eq_var** определено как False (0), будут приняты отдельные изменения двух выборок. Если значение **eq_var** определено как True (1), будут приняты равные изменения в выборках.

Ограничения:

Текстовые значения, значения NULL, а также отсутствующие значения в значении выражения приводят к тому, что функция возвращает значение NULL.

Примеры:

```
TTest_df( Group, Value )
TTest_df( Group, Value, false )
```

См. также:

Создание типичного отчета t-test (page 415)

TTest_dif

Функция **TTest_dif()** – это числовая функция, которая возвращает агрегированное среднее значение разницы t-критерия Стьюдента для двух независимых серий значений.

Эта функция применяется к t-критериям Стьюдента для независимых выборок.

Если функция используется в скрипте загрузки данных, значения повторяются в нескольких записях, как определено предложением group by.

Если функция используется в выражении диаграммы, значения повторяются в измерениях диаграммы.

Синтаксис:

```
TTest_dif (grp, value [, eq_var] )
```

Возвращаемые типы данных: числовое значение

Аргументы:

Аргументы

Аргумент	Описание
value	Выборка значений для оценки. Значения выборки должны быть сгруппированы логически, как указано только двумя значениями в элементе **group**. Если имя поля для значений выборки не указано в скрипте загрузки, поле автоматически получит имя **Value**.
grp	Поле, содержащее имена каждой из двух групп с выборками. Если имя поля для группы не указано в скрипте загрузки, поле автоматически получит имя **Type**.
eq_var	Если значение **eq_var** определено как False (0), будут приняты отдельные изменения двух выборок. Если значение **eq_var** определено как True (1), будут приняты равные изменения в выборках.

Ограничения:

Текстовые значения, значения NULL, а также отсутствующие значения в значении выражения приводят к тому, что функция возвращает значение NULL.

Примеры:

```
TTest_dif( Group, Value )
TTest_dif( Group, Value, false )
```

См. также:

Создание типичного отчета t-test (page 415)

TTest_lower

Функция **TTest_lower()** возвращает агрегированное значение нижнего предела доверительного интервала для двух независимых серий значений.

Эта функция применяется к t-критериям Стьюдента для независимых выборок.

Если функция используется в скрипте загрузки данных, значения повторяются в нескольких записях, как определено предложением group by.

Если функция используется в выражении диаграммы, значения повторяются в измерениях диаграммы.

Синтаксис:

```
TTest_lower (grp, value [, sig [, eq_var]])
```

Возвращаемые типы данных: числовое значение

Аргументы:

Аргументы

Аргумент	Описание
value	Выборка значений для оценки. Значения выборки должны быть сгруппированы логически, как указано только двумя значениями в элементе **group**. Если имя поля для значений выборки не указано в скрипте загрузки, поле автоматически получит имя **Value**.
grp	Поле, содержащее имена каждой из двух групп с выборками. Если имя поля для группы не указано в скрипте загрузки, поле автоматически получит имя **Type**.
sig	В **sig** можно указать двусторонний уровень важности. При отсутствии значения **sig** устанавливается равным 0,025, что приводит к значению доверительного интервала 95%.
eq_var	Если значение **eq_var** определено как False (0), будут приняты отдельные изменения двух выборок. Если значение **eq_var** определено как True (1), будут приняты равные изменения в выборках.

Ограничения:

Текстовые значения, значения NULL, а также отсутствующие значения в значении выражения приводят к тому, что функция возвращает значение NULL.

Примеры:

```
TTest_lower( Group, Value )
TTest_lower( Group, Value, Sig, false )
```

См. также:

Создание типичного отчета t-test (page 415)

TTest_sig

Функция **TTest_sig()** возвращает агрегированное значение двухвостого уровня важности t-критерия Стьюдента для двух независимых серий значений.

Эта функция применяется к t-критериям Стьюдента для независимых выборок.

Если функция используется в скрипте загрузки данных, значения повторяются в нескольких записях, как определено предложением group by.

Если функция используется в выражении диаграммы, значения повторяются в измерениях диаграммы.

Синтаксис:

```
TTest_sig (grp, value [, eq_var])
```

Возвращаемые типы данных: числовое значение

Аргументы:

Аргументы

Аргумент	Описание
value	Выборка значений для оценки. Значения выборки должны быть сгруппированы логически, как указано только двумя значениями в элементе **group**. Если имя поля для значений выборки не указано в скрипте загрузки, поле автоматически получит имя **Value**.
grp	Поле, содержащее имена каждой из двух групп с выборками. Если имя поля для группы не указано в скрипте загрузки, поле автоматически получит имя **Type**.
eq_var	Если значение **eq_var** определено как False (0), будут приняты отдельные изменения двух выборок. Если значение **eq_var** определено как True (1), будут приняты равные изменения в выборках.

Ограничения:

Текстовые значения, значения NULL, а также отсутствующие значения в значении выражения приводят к тому, что функция возвращает значение NULL.

Примеры:

```
TTest_sig( Group, Value )
TTest_sig( Group, Value, false )
```

См. также:

Создание типичного отчета t-test (page 415)

TTest_sterr

Функция **TTest_sterr()** возвращает агрегированное среднее значение разницы стандартной ошибки t-критерия Стьюдента для двух независимых серий значений.

Эта функция применяется к t-критериям Стьюдента для независимых выборок.

Если функция используется в скрипте загрузки данных, значения повторяются в нескольких записях, как определено предложением group by.

Если функция используется в выражении диаграммы, значения повторяются в измерениях диаграммы.

Синтаксис:

```
TTest_sterr (grp, value [, eq_var])
```

Возвращаемые типы данных: числовое значение

Аргументы:

Аргументы

Аргумент	Описание
value	Выборка значений для оценки. Значения выборки должны быть сгруппированы логически, как указано только двумя значениями в элементе **group**. Если имя поля для значений выборки не указано в скрипте загрузки, поле автоматически получит имя **Value**.
grp	Поле, содержащее имена каждой из двух групп с выборками. Если имя поля для группы не указано в скрипте загрузки, поле автоматически получит имя **Type**.
eq_var	Если значение **eq_var** определено как False (0), будут приняты отдельные изменения двух выборок. Если значение **eq_var** определено как True (1), будут приняты равные изменения в выборках.

Ограничения:

Текстовые значения, значения NULL, а также отсутствующие значения в значении выражения приводят к тому, что функция возвращает значение NULL.

Примеры:

```
TTest_sterr( Group, Value )
TTest_sterr( Group, Value, false )
```

См. также:

Создание типичного отчета t-test (page 415)

TTest_t

Функция **TTest_t()** возвращает агрегированное t-значение для двух независимых серий значений.

Эта функция применяется к t-критериям Стьюдента для независимых выборок.

Если функция используется в скрипте загрузки данных, значения повторяются в нескольких записях, как определено предложением group by.

Если функция используется в выражении диаграммы, значения повторяются в измерениях диаграммы.

Синтаксис:

```
TTest_t(grp, value[, eq_var])
```

Возвращаемые типы данных: числовое значение

Аргументы:

Аргументы

Аргумент	Описание
value	Выборка значений для оценки. Значения выборки должны быть сгруппированы логически, как указано только двумя значениями в элементе **group**. Если имя поля для значений выборки не указано в скрипте загрузки, поле автоматически получит имя **Value**.
grp	Поле, содержащее имена каждой из двух групп с выборками. Если имя поля для группы не указано в скрипте загрузки, поле автоматически получит имя **Type**.
eq_var	Если значение **eq_var** определено как False (0), будут приняты отдельные изменения двух выборок. Если значение **eq_var** определено как True (1), будут приняты равные изменения в выборках.

Ограничения:

Текстовые значения, значения NULL, а также отсутствующие значения в значении выражения приводят к тому, что функция возвращает значение NULL.

Пример:

```
TTest_t( Group, Value, false )
```

См. также:

Создание типичного отчета t-test (page 415)

TTest_upper

Функция **TTest_upper()** возвращает агрегированное значение верхнего предела доверительного интервала для двух независимых серий значений.

Эта функция применяется к t-критериям Стьюдента для независимых выборок.

Если функция используется в скрипте загрузки данных, значения повторяются в нескольких записях, как определено предложением group by.

Если функция используется в выражении диаграммы, значения повторяются в измерениях диаграммы.

Синтаксис:

```
TTest_upper (grp, value [, sig [, eq_var]])
```

Возвращаемые типы данных: числовое значение

Аргументы:

Аргументы

Аргумент	Описание
value	Выборка значений для оценки. Значения выборки должны быть сгруппированы логически, как указано только двумя значениями в элементе **group**. Если имя поля для значений выборки не указано в скрипте загрузки, поле автоматически получит имя **Value**.
grp	Поле, содержащее имена каждой из двух групп с выборками. Если имя поля для группы не указано в скрипте загрузки, поле автоматически получит имя **Type**.
sig	В **sig** можно указать двусторонний уровень важности. При отсутствии значения **sig** устанавливается равным 0,025, что приводит к значению доверительного интервала 95%.
eq_var	Если значение **eq_var** определено как False (0), будут приняты отдельные изменения двух выборок. Если значение **eq_var** определено как True (1), будут приняты равные изменения в выборках.

Ограничения:

Текстовые значения, значения NULL, а также отсутствующие значения в значении выражения приводят к тому, что функция возвращает значение NULL.

Примеры:

```
TTest_upper( Group, Value )
TTest_upper( Group, Value, sig, false )
```

См. также:

Создание типичного отчета t-test (page 415)

TTestw_conf

Функция **TTestw_conf()** возвращает агрегированное t-значение для двух независимых серий значений.

Эта функция применяется к t-критериям Стьюдента для двух независимых выборок, в которых серия входных данных дается во взвешенном формате двух столбцов.

Если функция используется в скрипте загрузки данных, значения повторяются в нескольких записях, как определено предложением group by.

Если функция используется в выражении диаграммы, значения повторяются в измерениях диаграммы.

Синтаксис:

```
TTestw_conf (weight, grp, value [, sig [, eq_var]])
```

Возвращаемые типы данных: числовое значение

Аргументы:

Аргументы

Аргумент	Описание
value	Выборка значений для оценки. Значения выборки должны быть сгруппированы логически, как указано только двумя значениями в элементе **group**. Если имя поля для значений выборки не указано в скрипте загрузки, поле автоматически получит имя **Value**.
weight	Каждое значение в элементе **value** может подсчитываться один или несколько раз согласно соответствующему значению веса в элементе **weight**.
grp	Поле, содержащее имена каждой из двух групп с выборками. Если имя поля для группы не указано в скрипте загрузки, поле автоматически получит имя **Type**.

Аргумент	Описание
sig	В **sig** можно указать двусторонний уровень важности. При отсутствии значения **sig** устанавливается равным 0,025, что приводит к значению доверительного интервала 95%.
eq_var	Если значение **eq_var** определено как False (0), будут приняты отдельные изменения двух выборок. Если значение **eq_var** определено как True (1), будут приняты равные изменения в выборках.

Ограничения:

Текстовые значения, значения NULL, а также отсутствующие значения в значении выражения приводят к тому, что функция возвращает значение NULL.

Примеры:

```
TTestw_conf( Weight, Group, Value )
TTestw_conf( Weight, Group, Value, sig, false )
```

См. также:

Создание типичного отчета t-test (page 415)

TTestw_df

Функция **TTestw_df()** возвращает агрегированное df-значение t-критерия Стьюдента (степени свободы) для двух независимых серий значений.

Эта функция применяется к t-критериям Стьюдента для двух независимых выборок, в которых серия входных данных дается во взвешенном формате двух столбцов.

Если функция используется в скрипте загрузки данных, значения повторяются в нескольких записях, как определено предложением group by.

Если функция используется в выражении диаграммы, значения повторяются в измерениях диаграммы.

Синтаксис:

```
TTestw_df (weight, grp, value [, eq_var])
```

Возвращаемые типы данных: числовое значение

Аргументы:

Аргументы

Аргумент	Описание
weight	Каждое значение в элементе **value** может подсчитываться один или несколько раз согласно соответствующему значению веса в элементе **weight**.
grp	Поле, содержащее имена каждой из двух групп с выборками. Если имя поля для группы не указано в скрипте загрузки, поле автоматически получит имя **Type**.
value	Выборка значений для оценки. Значения выборки должны быть сгруппированы логически, как указано только двумя значениями в элементе **group**. Если имя поля для значений выборки не указано в скрипте загрузки, поле автоматически получит имя **Value**.
eq_var	Если значение **eq_var** определено как False (0), будут приняты отдельные изменения двух выборок. Если значение **eq_var** определено как True (1), будут приняты равные изменения в выборках.

Ограничения:

Текстовые значения, значения NULL, а также отсутствующие значения в значении выражения приводят к тому, что функция возвращает значение NULL.

Примеры:

```
TTestw_df( Weight, Group, Value )
TTestw_df( Weight, Group, Value, false )
```

См. также:

Создание типичного отчета t-test (page 415)

TTestw_dif

Функция **TTestw_dif()** возвращает агрегированное среднее значение разницы t-критерия Стьюдента для двух независимых серий значений.

Эта функция применяется к t-критериям Стьюдента для двух независимых выборок, в которых серия входных данных дается во взвешенном формате двух столбцов.

Если функция используется в скрипте загрузки данных, значения повторяются в нескольких записях, как определено предложением group by.

Если функция используется в выражении диаграммы, значения повторяются в измерениях диаграммы.

Синтаксис:

```
TTestw_dif (weight, grp, value)
```

Возвращаемые типы данных: числовое значение

Аргументы:

Аргументы

Аргумент	Описание
weight	Каждое значение в элементе **value** может подсчитываться один или несколько раз согласно соответствующему значению веса в элементе **weight**.
grp	Поле, содержащее имена каждой из двух групп с выборками. Если имя поля для группы не указано в скрипте загрузки, поле автоматически получит имя **Type**.
value	Выборка значений для оценки. Значения выборки должны быть сгруппированы логически, как указано только двумя значениями в элементе **group**. Если имя поля для значений выборки не указано в скрипте загрузки, поле автоматически получит имя **Value**.

Ограничения:

Текстовые значения, значения NULL, а также отсутствующие значения в значении выражения приводят к тому, что функция возвращает значение NULL.

Примеры:

```
TTestw_dif( Weight, Group, Value )
TTestw_dif( Weight, Group, Value, false )
```

См. также:

Создание типичного отчета t-test (page 415)

TTestw_lower

Функция **TTestw_lower()** возвращает агрегированное значение нижнего предела доверительного интервала для двух независимых серий значений.

Эта функция применяется к t-критериям Стьюдента для двух независимых выборок, в которых серия входных данных дается во взвешенном формате двух столбцов.

Если функция используется в скрипте загрузки данных, значения повторяются в нескольких записях, как определено предложением group by.

Если функция используется в выражении диаграммы, значения повторяются в измерениях диаграммы.

Синтаксис:

```
TTestw_lower (weight, grp, value [, sig [, eq_var]])
```

Возвращаемые типы данных: числовое значение

Аргументы:

Аргументы

Аргумент	Описание
weight	Каждое значение в элементе **value** может подсчитываться один или несколько раз согласно соответствующему значению веса в элементе **weight**.
grp	Поле, содержащее имена каждой из двух групп с выборками. Если имя поля для группы не указано в скрипте загрузки, поле автоматически получит имя **Type**.
value	Выборка значений для оценки. Значения выборки должны быть сгруппированы логически, как указано только двумя значениями в элементе **group**. Если имя поля для значений выборки не указано в скрипте загрузки, поле автоматически получит имя **Value**.
sig	В **sig** можно указать двусторонний уровень важности. При отсутствии значения **sig** устанавливается равным 0,025, что приводит к значению доверительного интервала 95%.
eq_var	Если значение **eq_var** определено как False (0), будут приняты отдельные изменения двух выборок. Если значение **eq_var** определено как True (1), будут приняты равные изменения в выборках.

Ограничения:

Текстовые значения, значения NULL, а также отсутствующие значения в значении выражения приводят к тому, что функция возвращает значение NULL.

Примеры:

```
TTestw_lower( Weight, Group, Value )
TTestw_lower( Weight, Group, Value, sig, false )
```

См. также:

Создание типичного отчета t-test (page 415)

TTestw_sig

Функция **TTestw_sig()** возвращает агрегированное значение двухвостого уровня важности t-критерия Стьюдента для двух независимых серий значений.

Эта функция применяется к t-критериям Стьюдента для двух независимых выборок, в которых серия входных данных дается во взвешенном формате двух столбцов.

Если функция используется в скрипте загрузки данных, значения повторяются в нескольких записях, как определено предложением group by.

Если функция используется в выражении диаграммы, значения повторяются в измерениях диаграммы.

Синтаксис:

```
TTestw_sig ( weight, grp, value [, eq_var])
```

Возвращаемые типы данных: числовое значение

Аргументы:

Аргументы

Аргумент	Описание
weight	Каждое значение в элементе **value** может подсчитываться один или несколько раз согласно соответствующему значению веса в элементе **weight**.
grp	Поле, содержащее имена каждой из двух групп с выборками. Если имя поля для группы не указано в скрипте загрузки, поле автоматически получит имя **Type**.
value	Выборка значений для оценки. Значения выборки должны быть сгруппированы логически, как указано только двумя значениями в элементе **group**. Если имя поля для значений выборки не указано в скрипте загрузки, поле автоматически получит имя **Value**.
eq_var	Если значение **eq_var** определено как False (0), будут приняты отдельные изменения двух выборок. Если значение **eq_var** определено как True (1), будут приняты равные изменения в выборках.

Ограничения:

Текстовые значения, значения NULL, а также отсутствующие значения в значении выражения приводят к тому, что функция возвращает значение NULL.

Примеры:

```
TTestw_sig( Weight, Group, Value )
TTestw_sig( Weight, Group, Value, false )
```

См. также:

Создание типичного отчета t-test (page 415)

TTestw_sterr

Функция **TTestw_sterr()** возвращает агрегированное среднее значение разницы стандартной ошибки t-критерия Стьюдента для двух независимых серий значений.

Эта функция применяется к t-критериям Стьюдента для двух независимых выборок, в которых серия входных данных дается во взвешенном формате двух столбцов.

Если функция используется в скрипте загрузки данных, значения повторяются в нескольких записях, как определено предложением group by.

Если функция используется в выражении диаграммы, значения повторяются в измерениях диаграммы.

Синтаксис:

```
TTestw_sterr (weight, grp, value [, eq_var])
```

Возвращаемые типы данных: числовое значение

Аргументы:

Аргументы

Аргумент	Описание
weight	Каждое значение в элементе **value** может подсчитываться один или несколько раз согласно соответствующему значению веса в элементе **weight**.
grp	Поле, содержащее имена каждой из двух групп с выборками. Если имя поля для группы не указано в скрипте загрузки, поле автоматически получит имя **Type**.
value	Выборка значений для оценки. Значения выборки должны быть сгруппированы логически, как указано только двумя значениями в элементе **group**. Если имя поля для значений выборки не указано в скрипте загрузки, поле автоматически получит имя **Value**.
eq_var	Если значение **eq_var** определено как False (0), будут приняты отдельные изменения двух выборок. Если значение **eq_var** определено как True (1), будут приняты равные изменения в выборках.

Ограничения:

Текстовые значения, значения NULL, а также отсутствующие значения в значении выражения приводят к тому, что функция возвращает значение NULL.

Примеры:

```
TTestw_sterr( Weight, Group, Value )
TTestw_sterr( Weight, Group, Value, false )
```

См. также:

Создание типичного отчета t-test (page 415)

TTestw_t

Функция **TTestw_t()** возвращает агрегированное t-значение для двух независимых серий значений.

Эта функция применяется к t-критериям Стьюдента для двух независимых выборок, в которых серия входных данных дается во взвешенном формате двух столбцов.

Если функция используется в скрипте загрузки данных, значения повторяются в нескольких записях, как определено предложением group by.

Если функция используется в выражении диаграммы, значения повторяются в измерениях диаграммы.

Синтаксис:

```
ttestw_t (weight, grp, value [, eq_var])
```

Возвращаемые типы данных: числовое значение

Аргументы:

Аргументы

Аргумент	Описание
value	Выборка значений для оценки. Значения выборки должны быть сгруппированы логически, как указано только двумя значениями в элементе **group**. Если имя поля для значений выборки не указано в скрипте загрузки, поле автоматически получит имя **Value**.
weight	Каждое значение в элементе **value** может подсчитываться один или несколько раз согласно соответствующему значению веса в элементе **weight**.
grp	Поле, содержащее имена каждой из двух групп с выборками. Если имя поля для группы не указано в скрипте загрузки, поле автоматически получит имя **Type**.
eq_var	Если значение **eq_var** определено как False (0), будут приняты отдельные изменения двух выборок. Если значение **eq_var** определено как True (1), будут приняты равные изменения в выборках.

Ограничения:

Текстовые значения, значения NULL, а также отсутствующие значения в значении выражения приводят к тому, что функция возвращает значение NULL.

Примеры:

```
TTestw_t( Weight, Group, Value )
TTestw_t( Weight, Group, Value, false )
```

См. также:

Создание типичного отчета t-test (page 415)

TTestw_upper

Функция **TTestw_upper()** возвращает агрегированное значение верхнего предела доверительного интервала для двух независимых серий значений.

Эта функция применяется к t-критериям Стьюдента для двух независимых выборок, в которых серия входных данных дается во взвешенном формате двух столбцов.

Если функция используется в скрипте загрузки данных, значения повторяются в нескольких записях, как определено предложением group by.

Если функция используется в выражении диаграммы, значения повторяются в измерениях диаграммы.

Синтаксис:

```
TTestw_upper (weight, grp, value [, sig [, eq_var]])
```

Возвращаемые типы данных: числовое значение

Аргументы:

Аргументы

Аргумент	Описание
weight	Каждое значение в элементе **value** может подсчитываться один или несколько раз согласно соответствующему значению веса в элементе **weight**.
grp	Поле, содержащее имена каждой из двух групп с выборками. Если имя поля для группы не указано в скрипте загрузки, поле автоматически получит имя **Type**.
value	Выборка значений для оценки. Значения выборки должны быть сгруппированы логически, как указано только двумя значениями в элементе **group**. Если имя поля для значений выборки не указано в скрипте загрузки, поле автоматически получит имя **Value**.
sig	В **sig** можно указать двусторонний уровень важности. При отсутствии значения **sig** устанавливается равным 0,025, что приводит к значению доверительного интервала 95%.
eq_var	Если значение **eq_var** определено как False (0), будут приняты отдельные изменения двух выборок. Если значение **eq_var** определено как True (1), будут приняты равные изменения в выборках.

Ограничения:

Текстовые значения, значения NULL, а также отсутствующие значения в значении выражения приводят к тому, что функция возвращает значение NULL.

Примеры:

```
TTestw_upper( Weight, Group, Value )
```

```
TTestw_upper( Weight, Group, Value, sig, false )
```

См. также:

- *Создание типичного отчета t-test (page 415)*

TTest1_conf

Функция **TTest1_conf()** возвращает агрегированное значение доверительного интервала для серии значений.

Эта функция применяется к t-критериям Стьюдента для одной выборки.

Если функция используется в скрипте загрузки данных, значения повторяются в нескольких записях, как определено предложением group by.

Если функция используется в выражении диаграммы, значения повторяются в измерениях диаграммы.

Синтаксис:

```
TTest1_conf (value [, sig ])
```

Возвращаемые типы данных: числовое значение

Аргументы:

Аргументы

Аргумент	Описание
value	Выборки для оценки. Если имя поля для значений выборки не указано в скрипте загрузки, поле автоматически получит имя **Value**.
sig	В **sig** можно указать двусторонний уровень важности. При отсутствии значения **sig** устанавливается равным 0,025, что приводит к значению доверительного интервала 95%.

Ограничения:

Текстовые значения, значения NULL, а также отсутствующие значения в значении выражения приводят к тому, что функция возвращает значение NULL.

Примеры:

```
TTest1_conf( Value )
TTest1_conf( Value, 0.005 )
```

См. также:

- *Создание типичного отчета t-test (page 415)*

TTest1_df

Функция **TTest1_df()** возвращает агрегированное df-значение t-критерия Стьюдента (степени свободы) для серии значений.

Эта функция применяется к t-критериям Стьюдента для одной выборки.

Если функция используется в скрипте загрузки данных, значения повторяются в нескольких записях, как определено предложением group by.

Если функция используется в выражении диаграммы, значения повторяются в измерениях диаграммы.

Синтаксис:

```
TTest1_df (value)
```

Возвращаемые типы данных: числовое значение

Аргументы:

Аргументы

Аргумент	Описание
value	Выборки для оценки. Если имя поля для значений выборки не указано в скрипте загрузки, поле автоматически получит имя **Value**.

Ограничения:

Текстовые значения, значения NULL, а также отсутствующие значения в значении выражения приводят к тому, что функция возвращает значение NULL.

Пример:

```
TTest1_df( Value )
```

См. также:

Создание типичного отчета t-test (page 415)

TTest1_dif

Функция **TTest1_dif()** возвращает агрегированное среднее значение разницы t-критерия Стьюдента для серии значений.

Эта функция применяется к t-критериям Стьюдента для одной выборки.

Если функция используется в скрипте загрузки данных, значения повторяются в нескольких записях, как определено предложением group by.

Если функция используется в выражении диаграммы, значения повторяются в измерениях диаграммы.

Синтаксис:

```
TTest1_dif (value)
```

Возвращаемые типы данных: числовое значение

Аргументы:

Аргументы

Аргумент	Описание
value	Выборки для оценки. Если имя поля для значений выборки не указано в скрипте загрузки, поле автоматически получит имя **Value**.

Ограничения:

Текстовые значения, значения NULL, а также отсутствующие значения в значении выражения приводят к тому, что функция возвращает значение NULL.

Пример:

```
TTest1_dif( Value )
```

См. также:

Создание типичного отчета t-test (page 415)

TTest1_lower

Функция **TTest1_lower()** возвращает агрегированное значение нижнего предела доверительного интервала для серии значений.

Эта функция применяется к t-критериям Стьюдента для одной выборки.

Если функция используется в скрипте загрузки данных, значения повторяются в нескольких записях, как определено предложением group by.

Если функция используется в выражении диаграммы, значения повторяются в измерениях диаграммы.

Синтаксис:

```
TTest1_lower (value [, sig])
```

Возвращаемые типы данных: числовое значение

Аргументы:

Аргументы

Аргумент	Описание
value	Выборки для оценки. Если имя поля для значений выборки не указано в скрипте загрузки, поле автоматически получит имя **Value**.
sig	В **sig** можно указать двусторонний уровень важности. При отсутствии значения **sig** устанавливается равным 0,025, что приводит к значению доверительного интервала 95%.

Ограничения:

Текстовые значения, значения NULL, а также отсутствующие значения в значении выражения приводят к тому, что функция возвращает значение NULL.

Примеры:

```
TTest1_lower( Value )
TTest1_lower( Value, 0.005 )
```

См. также:

Создание типичного отчета t-test (page 415)

TTest1_sig

Функция **TTest1_sig()** возвращает агрегированное значение двухвостого уровня важности t-критерия Стьюдента для серии значений.

Эта функция применяется к t-критериям Стьюдента для одной выборки.

Если функция используется в скрипте загрузки данных, значения повторяются в нескольких записях, как определено предложением group by.

Если функция используется в выражении диаграммы, значения повторяются в измерениях диаграммы.

Синтаксис:

```
TTest1_sig (value)
```

Возвращаемые типы данных: числовое значение

Аргументы:

Аргументы

Аргумент	Описание
value	Выборки для оценки. Если имя поля для значений выборки не указано в скрипте загрузки, поле автоматически получит имя **Value**.

Ограничения:

Текстовые значения, значения NULL, а также отсутствующие значения в значении выражения приводят к тому, что функция возвращает значение NULL.

Пример:

```
TTest1_sig( Value )
```

См. также:

Создание типичного отчета t-test (page 415)

TTest1_sterr

Функция **TTest1_sterr()** возвращает агрегированное среднее значение разницы стандартной ошибки t-критерия Стьюдента для серии значений.

Эта функция применяется к t-критериям Стьюдента для одной выборки.

Если функция используется в скрипте загрузки данных, значения повторяются в нескольких записях, как определено предложением group by.

Если функция используется в выражении диаграммы, значения повторяются в измерениях диаграммы.

Синтаксис:

```
TTest1_sterr (value)
```

Возвращаемые типы данных: числовое значение

Аргументы:

Аргументы

Аргумент	Описание
value	Выборки для оценки. Если имя поля для значений выборки не указано в скрипте загрузки, поле автоматически получит имя **Value**.

Ограничения:

Текстовые значения, значения NULL, а также отсутствующие значения в значении выражения приводят к тому, что функция возвращает значение NULL.

Пример:

```
TTest1_sterr( Value )
```

См. также:

Создание типичного отчета t-test (page 415)

TTest1_t

Функция **TTest1_t()** возвращает агрегированное t-значение для серии значений.

Эта функция применяется к t-критериям Стьюдента для одной выборки.

Если функция используется в скрипте загрузки данных, значения повторяются в нескольких записях, как определено предложением group by.

Если функция используется в выражении диаграммы, значения повторяются в измерениях диаграммы.

Синтаксис:

```
TTest1_t (value)
```

Возвращаемые типы данных: числовое значение

Аргументы:

Аргументы

Аргумент	Описание
value	Выборки для оценки. Если имя поля для значений выборки не указано в скрипте загрузки, поле автоматически получит имя **Value**.

Ограничения:

Текстовые значения, значения NULL, а также отсутствующие значения в значении выражения приводят к тому, что функция возвращает значение NULL.

Пример:

```
TTest1_t( Value )
```

См. также:

- *Создание типичного отчета t-test (page 415)*

TTest1_upper

Функция **TTest1_upper()** возвращает агрегированное значение верхнего предела доверительного интервала для серии значений.

Эта функция применяется к t-критериям Стьюдента для одной выборки.

Если функция используется в скрипте загрузки данных, значения повторяются в нескольких записях, как определено предложением group by.

Если функция используется в выражении диаграммы, значения повторяются в измерениях диаграммы.

Синтаксис:

```
TTest1_upper (value [, sig])
```

Возвращаемые типы данных: числовое значение

Аргументы:

Аргументы

Аргумент	Описание
value	Выборки для оценки. Если имя поля для значений выборки не указано в скрипте загрузки, поле автоматически получит имя **Value**.
sig	В **sig** можно указать двусторонний уровень важности. При отсутствии значения **sig** устанавливается равным 0,025, что приводит к значению доверительного интервала 95%.

Ограничения:

Текстовые значения, значения NULL, а также отсутствующие значения в значении выражения приводят к тому, что функция возвращает значение NULL.

Примеры:

```
TTest1_upper( Value )
TTest1_upper( Value, 0.005 )
```

См. также:

- *Создание типичного отчета t-test (page 415)*

TTest1w_conf

Функция **TTest1w_conf()** – это **числовая** функция, которая возвращает агрегированное значение доверительного интервала для серии значений.

Эта функция применяется к t-критериям Стьюдента для одной выборки, в которой серия входных данных дается во взвешенном формате двух столбцов.

Если функция используется в скрипте загрузки данных, значения повторяются в нескольких записях, как определено предложением group by.

Если функция используется в выражении диаграммы, значения повторяются в измерениях диаграммы.

Синтаксис:

```
TTest1w_conf (weight, value [, sig ])
```

Возвращаемые типы данных: числовое значение

Аргументы:

Аргументы

Аргумент	Описание
value	Выборки для оценки. Если имя поля для значений выборки не указано в скрипте загрузки, поле автоматически получит имя **Value**.
weight	Каждое значение в элементе **value** может подсчитываться один или несколько раз согласно соответствующему значению веса в элементе **weight**.
sig	В **sig** можно указать двусторонний уровень важности. При отсутствии значения **sig** устанавливается равным 0,025, что приводит к значению доверительного интервала 95%.

Ограничения:

Текстовые значения, значения NULL, а также отсутствующие значения в значении выражения приводят к тому, что функция возвращает значение NULL.

Примеры:

```
TTest1w_conf( Weight, Value )
TTest1w_conf( Weight, Value, 0.005 )
```

См. также:

Создание типичного отчета t-test (page 415)

TTest1w_df

Функция **TTest1w_df()** возвращает агрегированное df-значение t-критерия Стьюдента (степени свободы) для серии значений.

Эта функция применяется к t-критериям Стьюдента для одной выборки, в которой серия входных данных дается во взвешенном формате двух столбцов.

Если функция используется в скрипте загрузки данных, значения повторяются в нескольких записях, как определено предложением group by.

Если функция используется в выражении диаграммы, значения повторяются в измерениях диаграммы.

Синтаксис:

```
TTest1w_df (weight, value)
```

Возвращаемые типы данных: числовое значение

Аргументы:

Аргументы

Аргумент	Описание
value	Выборки для оценки. Если имя поля для значений выборки не указано в скрипте загрузки, поле автоматически получит имя **Value**.
weight	Каждое значение в элементе **value** может подсчитываться один или несколько раз согласно соответствующему значению веса в элементе **weight**.

Ограничения:

Текстовые значения, значения NULL, а также отсутствующие значения в значении выражения приводят к тому, что функция возвращает значение NULL.

Пример:

```
TTest1w_df( Weight, Value )
```

См. также:

Создание типичного отчета t-test (page 415)

TTest1w_dif

Функция **TTest1w_dif()** возвращает агрегированное среднее значение разницы t-критерия Стьюдента для серии значений.

Эта функция применяется к t-критериям Стьюдента для одной выборки, в которой серия входных данных дается во взвешенном формате двух столбцов.

Если функция используется в скрипте загрузки данных, значения повторяются в нескольких записях, как определено предложением group by.

Если функция используется в выражении диаграммы, значения повторяются в измерениях диаграммы.

Синтаксис:

```
TTest1w_dif (weight, value)
```

Возвращаемые типы данных: числовое значение

Аргументы:

Аргументы

Аргумент	Описание
value	Выборки для оценки. Если имя поля для значений выборки не указано в скрипте загрузки, поле автоматически получит имя **Value**.
weight	Каждое значение в элементе **value** может подсчитываться один или несколько раз согласно соответствующему значению веса в элементе **weight**.

Ограничения:

Текстовые значения, значения NULL, а также отсутствующие значения в значении выражения приводят к тому, что функция возвращает значение NULL.

Пример:

```
TTest1w_dif( Weight, Value )
```

См. также:

Создание типичного отчета t-test (page 415)

TTest1w_lower

Функция **TTest1w_lower()** возвращает агрегированное значение нижнего предела доверительного интервала для серии значений.

Эта функция применяется к t-критериям Стьюдента для одной выборки, в которой серия входных данных дается во взвешенном формате двух столбцов.

Если функция используется в скрипте загрузки данных, значения повторяются в нескольких записях, как определено предложением group by.

Если функция используется в выражении диаграммы, значения повторяются в измерениях диаграммы.

Синтаксис:

```
TTest1w_lower (weight, value [, sig ])
```

Возвращаемые типы данных: числовое значение

Аргументы:

Аргументы

Аргумент	Описание
value	Выборки для оценки. Если имя поля для значений выборки не указано в скрипте загрузки, поле автоматически получит имя **Value**.
weight	Каждое значение в элементе **value** может подсчитываться один или несколько раз согласно соответствующему значению веса в элементе **weight**.
sig	В **sig** можно указать двусторонний уровень важности. При отсутствии значения **sig** устанавливается равным 0,025, что приводит к значению доверительного интервала 95%.

Ограничения:

Текстовые значения, значения NULL, а также отсутствующие значения в значении выражения приводят к тому, что функция возвращает значение NULL.

Примеры:

```
TTest1w_lower( Weight, Value )
TTest1w_lower( Weight, Value, 0.005 )
```

См. также:

Создание типичного отчета t-test (page 415)

TTest1w_sig

Функция **TTest1w_sig()** возвращает агрегированное значение двухвостого уровня важности t-критерия Стьюдента для серии значений.

Эта функция применяется к t-критериям Стьюдента для одной выборки, в которой серия входных данных дается во взвешенном формате двух столбцов.

Если функция используется в скрипте загрузки данных, значения повторяются в нескольких записях, как определено предложением group by.

Если функция используется в выражении диаграммы, значения повторяются в измерениях диаграммы.

Синтаксис:

```
TTest1w_sig (weight, value)
```

Возвращаемые типы данных: числовое значение

Аргументы:

Аргументы

Аргумент	Описание
value	Выборки для оценки. Если имя поля для значений выборки не указано в скрипте загрузки, поле автоматически получит имя **Value**.
weight	Каждое значение в элементе **value** может подсчитываться один или несколько раз согласно соответствующему значению веса в элементе **weight**.

Ограничения:

Текстовые значения, значения NULL, а также отсутствующие значения в значении выражения приводят к тому, что функция возвращает значение NULL.

Пример:

```
TTest1w_sig( Weight, Value )
```

См. также:

Создание типичного отчета t-test (page 415)

TTest1w_sterr

Функция **TTest1w_sterr()** возвращает агрегированное среднее значение разницы стандартной ошибки t-критерия Стьюдента для серии значений.

Эта функция применяется к t-критериям Стьюдента для одной выборки, в которой серия входных данных дается во взвешенном формате двух столбцов.

Если функция используется в скрипте загрузки данных, значения повторяются в нескольких записях, как определено предложением group by.

Если функция используется в выражении диаграммы, значения повторяются в измерениях диаграммы.

Синтаксис:

```
TTest1w_sterr (weight, value)
```

Возвращаемые типы данных: числовое значение

Аргументы:

Аргументы

Аргумент	Описание
value	Выборки для оценки. Если имя поля для значений выборки не указано в скрипте загрузки, поле автоматически получит имя **Value**.
weight	Каждое значение в элементе **value** может подсчитываться один или несколько раз согласно соответствующему значению веса в элементе **weight**.

Ограничения:

Текстовые значения, значения NULL, а также отсутствующие значения в значении выражения приводят к тому, что функция возвращает значение NULL.

Пример:

```
TTest1w_sterr( Weight, Value )
```

См. также:

Создание типичного отчета t-test (page 415)

TTest1w_t

Функция **TTest1w_t()** возвращает агрегированное t-значение для серии значений.

Эта функция применяется к t-критериям Стьюдента для одной выборки, в которой серия входных данных дается во взвешенном формате двух столбцов.

Если функция используется в скрипте загрузки данных, значения повторяются в нескольких записях, как определено предложением group by.

Если функция используется в выражении диаграммы, значения повторяются в измерениях диаграммы.

Синтаксис:

```
TTest1w_t ( weight, value)
```

Возвращаемые типы данных: числовое значение

Аргументы:

Аргументы

Аргумент	Описание
value	Выборки для оценки. Если имя поля для значений выборки не указано в скрипте загрузки, поле автоматически получит имя **Value**.
weight	Каждое значение в элементе **value** может подсчитываться один или несколько раз согласно соответствующему значению веса в элементе **weight**.

Ограничения:

Текстовые значения, значения NULL, а также отсутствующие значения в значении выражения приводят к тому, что функция возвращает значение NULL.

Пример:

```
TTest1w_t( Weight, Value )
```

См. также:

Создание типичного отчета t-test (page 415)

TTest1w_upper

Функция **TTest1w_upper()** возвращает агрегированное значение верхнего предела доверительного интервала для серии значений.

Эта функция применяется к t-критериям Стьюдента для одной выборки, в которой серия входных данных дается во взвешенном формате двух столбцов.

Если функция используется в скрипте загрузки данных, значения повторяются в нескольких записях, как определено предложением group by.

Если функция используется в выражении диаграммы, значения повторяются в измерениях диаграммы.

Синтаксис:

```
TTest1w_upper (weight, value [, sig])
```

Возвращаемые типы данных: числовое значение

Аргументы:

Аргументы

Аргумент	Описание
value	Выборки для оценки. Если имя поля для значений выборки не указано в скрипте загрузки, поле автоматически получит имя **Value**.
weight	Каждое значение в элементе **value** может подсчитываться один или несколько раз согласно соответствующему значению веса в элементе **weight**.
sig	В **sig** можно указать двусторонний уровень важности. При отсутствии значения **sig** устанавливается равным 0,025, что приводит к значению доверительного интервала 95%.

Ограничения:

Текстовые значения, значения NULL, а также отсутствующие значения в значении выражения приводят к тому, что функция возвращает значение NULL.

Примеры:

```
TTest1w_upper( Weight, Value )
TTest1w_upper( Weight, Value, 0.005 )
```

См. также:

Создание типичного отчета t-test (page 415)

Функции Z-критериев

Статистическое исследование двух генеральных средних. Z-критерий для двух выборок проверяет, отличаются ли две выборки. Он обычно используется, когда два обычных распределения имеют известные изменения, и когда в эксперименте используется большой размер выборки.

Статистические функции тестирования z-критерия сгруппированы согласно типу серии вводимых данных, применяемой к функции.

Если функция используется в скрипте загрузки данных, значения повторяются в нескольких записях, как определено предложением group by.

Если функция используется в выражении диаграммы, значения повторяются в измерениях диаграммы.

Примеры использования функций z-test (page 419)

Функции формата одного столбца

Следующие функции применяются к z-критериям с простыми сериями вводимых данных.

ztest_conf

Функция **ZTest_conf()** возвращает агрегированное z-значение для серии значений.

```
Функция ZTest_conf() возвращает агрегированное z-значение для серии значений. (value [, sigma [, sig ])
```

ztest_dif

Функция **ZTest_dif()** возвращает агрегированное среднее значение разницы z-критерия для серии значений.

```
Функция ZTest_dif() возвращает агрегированное среднее значение разницы z-критерия для серии значений. (value [, sigma])
```

ztest_sig

Функция **ZTest_sig()** возвращает агрегированное значение двухвостого уровня важности z-критерия для серии значений.

```
Функция ZTest_sig() возвращает агрегированное значение двухвостого уровня важности z-критерия для серии значений. (value [, sigma])
```

ztest_sterr

Функция **ZTest_sterr()** возвращает агрегированное среднее значение разницы стандартной ошибки z-критерия для серии значений.

```
Функция ZTest_sterr() возвращает агрегированное среднее значение разницы стандартной ошибки z-критерия для серии значений. (value [, sigma])
```

ztest_z

Функция **ZTest_z()** возвращает агрегированное z-значение для серии значений.

```
Функция ZTest_z() возвращает агрегированное z-значение для серии значений. (value [, sigma])
```

ztest_lower

Функция **ZTest_lower()** возвращает агрегированное значение нижнего предела доверительного интервала для двух независимых серий значений.

```
Функция ZTest_lower() возвращает агрегированное значение нижнего предела доверительного интервала для двух независимых серий значений. (grp, value [, sig [, eq_var]])
```

ztest_upper

Функция **ZTest_upper()** возвращает агрегированное значение верхнего предела доверительного интервала для двух независимых серий значений.

```
Функция ZTest_upper() возвращает агрегированное значение верхнего предела доверительного интервала для двух независимых серий значений. (grp, value [, sig [, eq_var]])
```

Функции взвешенного формата двух столбцов

Следующие функции применяются к z-критериям, в которых серия входных данных дается во взвешенном формате двух столбцов.

ztestw_conf

Функция **ZTestw_conf()** возвращает агрегированное значение доверительного интервала z-критерия для серии значений.

```
Функция ZTestw_conf() возвращает агрегированное значение доверительного интервала z-критерия для серии значений. (weight, value [, sigma [, sig]])
```

ztestw_dif

Функция **ZTestw_dif()** возвращает агрегированное среднее значение разницы z-критерия для серии значений.

```
Функция ZTestw_dif() возвращает агрегированное среднее значение разницы z-критерия для серии значений.  (weight, value [, sigma])
```

ztestw_lower

Функция **ZTestw_lower()** возвращает агрегированное значение нижнего предела доверительного интервала для двух независимых серий значений.

```
Функция ZTestw_lower() возвращает агрегированное значение нижнего предела доверительного интервала для двух независимых серий значений. (weight, value [, sigma])
```

ztestw_sig

Функция **ZTestw_sig()** возвращает агрегированное значение двухвостого уровня важности z-критерия для серии значений.

```
Функция ZTestw_sig() возвращает агрегированное значение двухвостого уровня важности z-критерия для серии значений. (weight, value [, sigma])
```

ztestw_sterr

Функция **ZTestw_sterr()** возвращает агрегированное среднее значение разницы стандартной ошибки z-критерия для серии значений.

```
Функция ZTestw_sterr() возвращает агрегированное среднее значение разницы стандартной ошибки z-критерия для серии значений. (weight, value [, sigma])
```

ztestw_upper

Функция **ZTestw_upper()** возвращает агрегированное значение верхнего предела доверительного интервала для двух независимых серий значений.

Функция ZTestw_upper() возвращает агрегированное значение верхнего предела доверительного интервала для двух независимых серий значений. (weight, value [, sigma])

ztestw_z

Функция **ZTestw_z()** возвращает агрегированное z-значение для серии значений.

Функция ZTestw_z() возвращает агрегированное z-значение для серии значений. (weight, value [, sigma])

ZTest_z

Функция **ZTest_z()** возвращает агрегированное z-значение для серии значений.

Если функция используется в скрипте загрузки данных, значения повторяются в нескольких записях, как определено предложением group by.

Если функция используется в выражении диаграммы, значения повторяются в измерениях диаграммы.

Синтаксис:

ZTest_z(value[, sigma])

Возвращаемые типы данных: числовое значение

Аргументы:

Аргументы

Аргумент	Описание
value	Выборка значений для оценки. Принимается генеральное среднее 0. Чтобы выполнить проверку в отношении другого среднего значения, вычтите это значение из выборки значений.
sigma	Если стандартное отклонение известно, его можно указать в элементе **sigma**. Если элемент **sigma** отсутствует, используется действительное стандартное отклонение выборки.

Ограничения:

Текстовые значения, значения NULL, а также отсутствующие значения в значении выражения приводят к тому, что функция возвращает значение NULL.

Пример:

```
ZTest_z( Value-TestValue )
```

См. также:

Примеры использования функций z-test (page 419)

ZTest_sig

Функция **ZTest_sig()** возвращает агрегированное значение двухвостого уровня важности z-критерия для серии значений.

Если функция используется в скрипте загрузки данных, значения повторяются в нескольких записях, как определено предложением group by.

Если функция используется в выражении диаграммы, значения повторяются в измерениях диаграммы.

Синтаксис:

```
ZTest_sig(value[, sigma])
```

Возвращаемые типы данных: числовое значение

Аргументы:

Аргументы

Аргумент	Описание
value	Выборка значений для оценки. Принимается генеральное среднее 0. Чтобы выполнить проверку в отношении другого среднего значения, вычтите это значение из выборки значений.
sigma	Если стандартное отклонение известно, его можно указать в элементе **sigma**. Если элемент **sigma** отсутствует, используется действительное стандартное отклонение выборки.

Ограничения:

Текстовые значения, значения NULL, а также отсутствующие значения в значении выражения приводят к тому, что функция возвращает значение NULL.

Пример:

```
ZTest_sig(Value-TestValue)
```

См. также:

Примеры использования функций z-test (page 419)

ZTest_dif

Функция **ZTest_dif()** возвращает агрегированное среднее значение разницы z-критерия для серии значений.

Если функция используется в скрипте загрузки данных, значения повторяются в нескольких записях, как определено предложением group by.

Если функция используется в выражении диаграммы, значения повторяются в измерениях диаграммы.

Синтаксис:

```
ZTest_dif(value[, sigma])
```

Возвращаемые типы данных: числовое значение

Аргументы:

Аргументы

Аргумент	Описание
value	Выборка значений для оценки. Принимается генеральное среднее 0. Чтобы выполнить проверку в отношении другого среднего значения, вычтите это значение из выборки значений.
sigma	Если стандартное отклонение известно, его можно указать в элементе **sigma**. Если элемент **sigma** отсутствует, используется действительное стандартное отклонение выборки.

Ограничения:

Текстовые значения, значения NULL, а также отсутствующие значения в значении выражения приводят к тому, что функция возвращает значение NULL.

Пример:

```
ZTest_dif(Value-TestValue)
```

См. также:

Примеры использования функций z-test (page 419)

ZTest_sterr

Функция **ZTest_sterr()** возвращает агрегированное среднее значение разницы стандартной ошибки z-критерия для серии значений.

Если функция используется в скрипте загрузки данных, значения повторяются в нескольких записях, как определено предложением group by.

Если функция используется в выражении диаграммы, значения повторяются в измерениях диаграммы.

Синтаксис:

```
ZTest_sterr(value[, sigma])
```

Возвращаемые типы данных: числовое значение

Аргументы:

Аргументы

Аргумент	Описание
value	Выборка значений для оценки. Принимается генеральное среднее 0. Чтобы выполнить проверку в отношении другого среднего значения, вычтите это значение из выборки значений.
sigma	Если стандартное отклонение известно, его можно указать в элементе **sigma**. Если элемент **sigma** отсутствует, используется действительное стандартное отклонение выборки.

Ограничения:

Текстовые значения, значения NULL, а также отсутствующие значения в значении выражения приводят к тому, что функция возвращает значение NULL.

Пример:

```
ZTest_sterr(Value-TestValue)
```

См. также:

Примеры использования функций z-test (page 419)

ZTest_conf

Функция **ZTest_conf()** возвращает агрегированное z-значение для серии значений.

Если функция используется в скрипте загрузки данных, значения повторяются в нескольких записях, как определено предложением group by.

Если функция используется в выражении диаграммы, значения повторяются в измерениях диаграммы.

Синтаксис:

```
ZTest_conf(value[, sigma[, sig]])
```

Возвращаемые типы данных: числовое значение

Аргументы:

Аргументы

Аргумент	Описание
value	Выборка значений для оценки. Принимается генеральное среднее 0. Чтобы выполнить проверку в отношении другого среднего значения, вычтите это значение из выборки значений.
sigma	Если стандартное отклонение известно, его можно указать в элементе **sigma**. Если элемент **sigma** отсутствует, используется действительное стандартное отклонение выборки.
sig	В **sig** можно указать двусторонний уровень важности. При отсутствии значения **sig** устанавливается равным 0,025, что приводит к значению доверительного интервала 95%.

Ограничения:

Текстовые значения, значения NULL, а также отсутствующие значения в значении выражения приводят к тому, что функция возвращает значение NULL.

Пример:

```
ZTest_conf(Value-TestValue)
```

См. также:

Примеры использования функций z-test (page 419)

ZTest_lower

Функция **ZTest_lower()** возвращает агрегированное значение нижнего предела доверительного интервала для двух независимых серий значений.

Если функция используется в скрипте загрузки данных, значения повторяются в нескольких записях, как определено предложением group by.

Если функция используется в выражении диаграммы, значения повторяются в измерениях диаграммы.

Синтаксис:

```
ZTest_lower (grp, value [, sig [, eq_var]])
```

Возвращаемые типы данных: числовое значение

Аргументы:

Аргументы

Аргумент	Описание
value	Выборка значений для оценки. Значения выборки должны быть сгруппированы логически, как указано только двумя значениями в элементе **group**. Если имя поля для значений выборки не указано в скрипте загрузки, поле автоматически получит имя **Value**.
grp	Поле, содержащее имена каждой из двух групп с выборками. Если имя поля для группы не указано в скрипте загрузки, поле автоматически получит имя **Type**.
sig	В **sig** можно указать двусторонний уровень важности. При отсутствии значения **sig** устанавливается равным 0,025, что приводит к значению доверительного интервала 95%.
eq_var	Если значение **eq_var** определено как False (0), будут приняты отдельные изменения двух выборок. Если значение **eq_var** определено как True (1), будут приняты равные изменения в выборках.

Ограничения:

Текстовые значения, значения NULL, а также отсутствующие значения в значении выражения приводят к тому, что функция возвращает значение NULL.

Примеры:

```
ZTest_lower( Group, Value )
ZTest_lower( Group, Value, sig, false )
```

См. также:

Примеры использования функций z-test (page 419)

ZTest_upper

Функция **ZTest_upper()** возвращает агрегированное значение верхнего предела доверительного интервала для двух независимых серий значений.

Эта функция применяется к t-критериям Стьюдента для независимых выборок.

Если функция используется в скрипте загрузки данных, значения повторяются в нескольких записях, как определено предложением group by.

Если функция используется в выражении диаграммы, значения повторяются в измерениях диаграммы.

Синтаксис:

```
ZTest_upper (grp, value [, sig [, eq_var]])
```

Возвращаемые типы данных: числовое значение

Аргументы:

Аргументы

Аргумент	Описание
value	Выборка значений для оценки. Значения выборки должны быть сгруппированы логически, как указано только двумя значениями в элементе **group**. Если имя поля для значений выборки не указано в скрипте загрузки, поле автоматически получит имя **Value**.
grp	Поле, содержащее имена каждой из двух групп с выборками. Если имя поля для группы не указано в скрипте загрузки, поле автоматически получит имя **Type**.
sig	В **sig** можно указать двусторонний уровень важности. При отсутствии значения **sig** устанавливается равным 0,025, что приводит к значению доверительного интервала 95%.
eq_var	Если значение **eq_var** определено как False (0), будут приняты отдельные изменения двух выборок. Если значение **eq_var** определено как True (1), будут приняты равные изменения в выборках.

Ограничения:

Текстовые значения, значения NULL, а также отсутствующие значения в значении выражения приводят к тому, что функция возвращает значение NULL.

Примеры:

```
ZTest_upper( Group, Value )
ZTest_upper( Group, Value, sig, false )
```

См. также:

Примеры использования функций z-test (page 419)

ZTestw_z

Функция **ZTestw_z()** возвращает агрегированное z-значение для серии значений.

Эта функция применяется к z-критериям, в которых серия входных данных дается во взвешенном формате двух столбцов.

Если функция используется в скрипте загрузки данных, значения повторяются в нескольких записях, как определено предложением group by.

Если функция используется в выражении диаграммы, значения повторяются в измерениях диаграммы.

Синтаксис:

```
ZTestw_z (weight, value [, sigma])
```

Возвращаемые типы данных: числовое значение

Аргументы:

Аргументы

Аргумент	Описание
value	Эти значения возвращаются с помощью **value**. Принимается среднее значение выборки 0. Чтобы выполнить проверку в отношении другого среднего значения, вычтите это значение из выборки значений.
weight	Каждое выборочное значение в элементе **value** может подсчитываться один или несколько раз согласно соответствующему значению веса в элементе **weight**.
sigma	Если стандартное отклонение известно, его можно указать в элементе **sigma**. Если элемент **sigma** отсутствует, используется действительное стандартное отклонение выборки.

Ограничения:

Текстовые значения, значения NULL, а также отсутствующие значения в значении выражения приводят к тому, что функция возвращает значение NULL.

Пример:

```
ZTestw_z( Weight, Value-TestValue)
```

См. также:

Примеры использования функций z-test (page 419)

ZTestw_sig

Функция **ZTestw_sig()** возвращает агрегированное значение двухвостого уровня важности z-критерия для серии значений.

Эта функция применяется к z-критериям, в которых серия входных данных дается во взвешенном формате двух столбцов.

Если функция используется в скрипте загрузки данных, значения повторяются в нескольких записях, как определено предложением group by.

Если функция используется в выражении диаграммы, значения повторяются в измерениях диаграммы.

Синтаксис:

```
ZTestw_sig (weight, value [, sigma])
```

Возвращаемые типы данных: числовое значение

Аргументы:

Аргументы

Аргумент	Описание
value	Эти значения возвращаются с помощью **value**. Принимается среднее значение выборки 0. Чтобы выполнить проверку в отношении другого среднего значения, вычтите это значение из выборки значений.
weight	Каждое выборочное значение в элементе **value** может подсчитываться один или несколько раз согласно соответствующему значению веса в элементе **weight**.
sigma	Если стандартное отклонение известно, его можно указать в элементе **sigma**. Если элемент **sigma** отсутствует, используется действительное стандартное отклонение выборки.

Ограничения:

Текстовые значения, значения NULL, а также отсутствующие значения в значении выражения приводят к тому, что функция возвращает значение NULL.

Пример:

```
ZTestw_sig( Weight, Value-TestValue)
```

См. также:

Примеры использования функций z-test (page 419)

ZTestw_dif

Функция **ZTestw_dif()** возвращает агрегированное среднее значение разницы z-критерия для серии значений.

Эта функция применяется к z-критериям, в которых серия входных данных дается во взвешенном формате двух столбцов.

Если функция используется в скрипте загрузки данных, значения повторяются в нескольких записях, как определено предложением group by.

Если функция используется в выражении диаграммы, значения повторяются в измерениях диаграммы.

Синтаксис:

```
ZTestw_dif ( weight, value [, sigma])
```

Возвращаемые типы данных: числовое значение

Аргументы:

Аргументы

Аргумент	Описание
value	Эти значения возвращаются с помощью **value**. Принимается среднее значение выборки 0. Чтобы выполнить проверку в отношении другого среднего значения, вычтите это значение из выборки значений.
weight	Каждое выборочное значение в элементе **value** может подсчитываться один или несколько раз согласно соответствующему значению веса в элементе **weight**.
sigma	Если стандартное отклонение известно, его можно указать в элементе **sigma**. Если элемент **sigma** отсутствует, используется действительное стандартное отклонение выборки.

Ограничения:

Текстовые значения, значения NULL, а также отсутствующие значения в значении выражения приводят к тому, что функция возвращает значение NULL.

Пример:

```
ZTestw_dif( Weight, Value-TestValue)
```

См. также:

Примеры использования функций z-test (page 419)

ZTestw_sterr

Функция **ZTestw_sterr()** возвращает агрегированное среднее значение разницы стандартной ошибки z-критерия для серии значений.

Эта функция применяется к z-критериям, в которых серия входных данных дается во взвешенном формате двух столбцов.

Если функция используется в скрипте загрузки данных, значения повторяются в нескольких записях, как определено предложением group by.

Если функция используется в выражении диаграммы, значения повторяются в измерениях диаграммы.

Синтаксис:

```
ZTestw_sterr (weight, value [, sigma])
```

Возвращаемые типы данных: числовое значение

Аргументы:

Аргументы

Аргумент	Описание
value	Эти значения возвращаются с помощью **value**. Принимается среднее значение выборки 0. Чтобы выполнить проверку в отношении другого среднего значения, вычтите это значение из выборки значений.
weight	Каждое выборочное значение в элементе **value** может подсчитываться один или несколько раз согласно соответствующему значению веса в элементе **weight**.
sigma	Если стандартное отклонение известно, его можно указать в элементе **sigma**. Если элемент **sigma** отсутствует, используется действительное стандартное отклонение выборки.

Ограничения:

Текстовые значения, значения NULL, а также отсутствующие значения в значении выражения приводят к тому, что функция возвращает значение NULL.

Пример:

```
ZTestw_sterr( Weight, Value-TestValue)
```

См. также:

Примеры использования функций z-test (page 419)

ZTestw_conf

Функция **ZTestw_conf()** возвращает агрегированное значение доверительного интервала z-критерия для серии значений.

Эта функция применяется к z-критериям, в которых серия входных данных дается во взвешенном формате двух столбцов.

Если функция используется в скрипте загрузки данных, значения повторяются в нескольких записях, как определено предложением group by.

Если функция используется в выражении диаграммы, значения повторяются в измерениях диаграммы.

Синтаксис:

```
ZTest_conf(weight, value[, sigma[, sig]])
```

Возвращаемые типы данных: числовое значение

Аргументы:

Аргументы

Аргумент	Описание
value	Выборка значений для оценки. Принимается генеральное среднее 0. Чтобы выполнить проверку в отношении другого среднего значения, вычтите это значение из выборки значений.
weight	Каждое выборочное значение в элементе **value** может подсчитываться один или несколько раз согласно соответствующему значению веса в элементе **weight**.
sigma	Если стандартное отклонение известно, его можно указать в элементе **sigma**. Если элемент **sigma** отсутствует, используется действительное стандартное отклонение выборки.
sig	В **sig** можно указать двусторонний уровень важности. При отсутствии значения **sig** устанавливается равным 0,025, что приводит к значению доверительного интервала 95%.

Ограничения:

Текстовые значения, значения NULL, а также отсутствующие значения в значении выражения приводят к тому, что функция возвращает значение NULL.

Пример:

```
ZTestw_conf( Weight, Value-TestValue)
```

См. также:

Примеры использования функций z-test (page 419)

ZTestw_lower

Функция **ZTestw_lower()** возвращает агрегированное значение нижнего предела доверительного интервала для двух независимых серий значений.

Если функция используется в скрипте загрузки данных, значения повторяются в нескольких записях, как определено предложением group by.

Если функция используется в выражении диаграммы, значения повторяются в измерениях диаграммы.

Синтаксис:

```
ZTestw_lower (grp, value [, sig [, eq_var]])
```

Возвращаемые типы данных: числовое значение

Аргументы:

Аргумент	Описание
value	Выборка значений для оценки. Значения выборки должны быть сгруппированы логически, как указано только двумя значениями в элементе **group**. Если имя поля для значений выборки не указано в скрипте загрузки, поле автоматически получит имя **Value**.
grp	Поле, содержащее имена каждой из двух групп с выборками. Если имя поля для группы не указано в скрипте загрузки, поле автоматически получит имя **Type**.
sig	В **sig** можно указать двусторонний уровень важности. При отсутствии значения **sig** устанавливается равным 0,025, что приводит к значению доверительного интервала 95%.
eq_var	Если значение **eq_var** определено как False (0), будут приняты отдельные изменения двух выборок. Если значение **eq_var** определено как True (1), будут приняты равные изменения в выборках.

Ограничения:

Текстовые значения, значения NULL, а также отсутствующие значения в значении выражения приводят к тому, что функция возвращает значение NULL.

Примеры:

```
ZTestw_lower( Group, Value )
ZTestw_lower( Group, Value, sig, false )
```

См. также:

Примеры использования функций z-test (page 419)

ZTestw_upper

Функция **ZTestw_upper()** возвращает агрегированное значение верхнего предела доверительного интервала для двух независимых серий значений.

Эта функция применяется к t-критериям Стьюдента для независимых выборок.

Если функция используется в скрипте загрузки данных, значения повторяются в нескольких записях, как определено предложением group by.

Если функция используется в выражении диаграммы, значения повторяются в измерениях диаграммы.

Синтаксис:

```
ZTestw_upper (grp, value [, sig [, eq_var]])
```

Возвращаемые типы данных: числовое значение

Аргументы:

Аргументы

Аргумент	Описание
value	Выборка значений для оценки. Значения выборки должны быть сгруппированы логически, как указано только двумя значениями в элементе **group**. Если имя поля для значений выборки не указано в скрипте загрузки, поле автоматически получит имя **Value**.
grp	Поле, содержащее имена каждой из двух групп с выборками. Если имя поля для группы не указано в скрипте загрузки, поле автоматически получит имя **Type**.
sig	В **sig** можно указать двусторонний уровень важности. При отсутствии значения **sig** устанавливается равным 0,025, что приводит к значению доверительного интервала 95%.
eq_var	Если значение **eq_var** определено как False (0), будут приняты отдельные изменения двух выборок. Если значение **eq_var** определено как True (1), будут приняты равные изменения в выборках.

Ограничения:

Текстовые значения, значения NULL, а также отсутствующие значения в значении выражения приводят к тому, что функция возвращает значение NULL.

Примеры:

```
ZTestw_upper( Group, Value )
ZTestw_upper( Group, Value, sig, false )
```

См. также:

Примеры использования функций z-test (page 419)

Примеры статистических тестовых функций

В этом разделе указаны примеры статистических тестовых функций применительно к диаграммам и скрипту загрузки данных.

Примеры использования функций chi2-test в диаграммах

Функции chi2-test используются для обнаружения значений, связанных со статистическим анализом значения Хи-квадрат.

В этом разделе описано, как построить визуализации с помощью данных образца, чтобы найти значения функций теста распределения значения Хи-квадрат, доступных в программе Qlik Sense. Описание синтаксиса и аргументов см. в индивидуальных темах функций диаграммы chi2-test.

Загрузка данных для образцов

Существует три набора данных образца, описывающих три различных статистических образца для загрузки в скрипт.

Выполните следующие действия.

1. Создайте новое приложение.
2. Введите в загрузке данных следующее:

```
// Sample_1 data is pre-aggregated... Note: make sure you set your DecimalSep='.' at the
top of the script.
Sample_1:
LOAD * inline [
Grp,Grade,Count
I,A,15
I,B,7
I,C,9
I,D,20
I,E,26
I,F,19
II,A,10
II,B,11
II,C,7
II,D,15
II,E,21
II,F,16
];
// Sample_2 data is pre-aggregated: If raw data is used, it must be aggregated using
count()...
Sample_2:
LOAD * inline [
Sex,Opinion,OpCount
1,2,58
1,1,11
1,0,10
2,2,35
2,1,25
2,0,23 ] (delimiter is ',');
// Sample_3a data is transformed using the crosstable statement...
Sample_3a:
crosstable(Gender, Actual) LOAD
Description,
[Men (Actual)] as Men,
[Women (Actual)] as Women;
LOAD * inline [
Men (Actual),Women (Actual),Description
58,35,Agree
11,25,Neutral
10,23,Disagree ] (delimiter is ',');
// Sample_3b data is transformed using the crosstable statement...
Sample_3b:
crosstable(Gender, Expected) LOAD
Description,
```

```
[Men (Expected)] as Men,
[Women (Expected)] as Women;
LOAD * inline [
Men (Expected),Women (Expected),Description
45.35,47.65,Agree
17.56,18.44,Neutral
16.09,16.91,Disagree ] (delimiter is ',');
// Sample_3a and Sample_3b will result in a (fairly harmless) Synthetic Key...
```

3. Щелкните для загрузки данных.

Создание визуализаций функции диаграммы chi2-test

Пример: Образец 1

Выполните следующие действия.

1. В редакторе загрузки данных щелкните , чтобы перейти в вид приложения, и нажмите ранее созданный лист.
   Откроется вид листа.
2. Щелкните **Изменить лист**, чтобы изменить лист.
3. Из раздела **Диаграммы** добавьте таблицу, а из раздела **Поля** добавьте элементы Grp, Grade и Count в качестве измерений.
   В этой таблице показаны данные образца.
4. Добавьте другую таблицу со следующими выражениями в качестве измерения:
   `ValueList('p','df','Chi2')`
   В данном случае используется функция синтетического измерения для создания меток для измерений с именами трех функций chi2-test.
5. Добавьте следующее выражение в таблицу в качестве меры:
   ```
   IF(ValueList('p','df','Chi2')='p',Chi2Test_p(Grp,Grade,Count),
   IF(ValueList('p','df','Chi2')='df',Chi2Test_df(Grp,Grade,Count),
   Chi2Test_Chi2(Grp,Grade,Count)))
   ```
   В таком случае результирующее значение каждой функции chi2-test будет помещено в таблицу рядом со связанным с ним синтетическим измерением.
6. Задайте **Формат чисел** меры в положение **Число** и **3 Значащие цифры**.

> *В выражении меры вместо этого можно использовать следующее выражение:* `Pick(Match(ValueList('p','df','Chi2'),'p','df','Chi2'),Chi2Test_p(Grp,Grade,Count),Chi2Test_df(Grp,Grade,Count),Chi2Test_Chi2(Grp,Grade,Count))`

Результат:

Полученная в результате таблица для функций chi2-test для данных образца 1 будет содержать следующие значения.

Результирующая таблица

p	df	Chi2
0.820	5	2.21

Пример: Образец 2

Выполните следующие действия.

1. На листе, который был изменен в примере для образца 1, из раздела **Диаграммы** добавьте таблицу, а из раздела **Поля** добавьте элементы Sex, Opinion и OpCount в качестве измерений.
2. Сделайте копию результатов таблицы из образца 1 с помощью команд **Копировать** и **Вставить**. Измените выражение в мере и замените аргументы во всех трех функциях chi2-test с именами полей, используемыми в данных образца 2, например: `Chi2Test_p (Sex,Opinion,OpCount)`.

Результат:

Полученная в результате таблица для функций chi2-test для данных образца 2 будет содержать следующие значения.

Результирующая таблица

p	df	Chi2
0.000309	2	16.2

Пример: Образец 3

Выполните следующие действия.

1. Создайте еще две таблицы так же, как в примерах для данных образцов 1 и 2. В таблице измерений используйте следующие поля в качестве измерений: Gender, Description, Actual и Expected.
2. В таблице результатов используйте имена полей, используемые в данных образца 3, например: `Chi2Test_p(Gender,Description,Actual,Expected)`.

Результат:

Полученная в результате таблица для функций chi2-test для данных образца 3 будет содержать следующие значения.

Результирующая таблица

p	df	Chi2
0.000308	2	16.2

Примеры использования функций chi2-test в скрипте загрузки данных

Функции chi2-test используются для обнаружения значений, связанных со статистическим анализом значения Хи-квадрат. В этом разделе описано, как

использовать функции теста распределения значения Хи-квадрат, доступные в Qlik Sense в скрипте загрузки данных. Описание синтаксиса и аргументов см. в индивидуальных темах функций скрипта chi2-test.

В этом примере используется таблица, содержащая количество студентов, достигших степени (A-F), для двух групп студентов (I и II).

Data table

Group	A	B	C	D	E	F
I	15	7	9	20	26	19
II	10	11	7	15	21	16

Загрузка данных образца

Выполните следующие действия.

1. Создайте новое приложение.
2. Введите в редакторе загрузки данных следующее:
```
// Sample_1 data is pre-aggregated... Note: make sure you set your DecimalSep='.' at the top of the script.
Sample_1:
LOAD * inline [
Grp,Grade,Count
I,A,15
I,B,7
I,C,9
I,D,20
I,E,26
I,F,19
II,A,10
II,B,11
II,C,7
II,D,15
II,E,21
II,F,16
];
```
3. Щелкните для загрузки данных.

Данные образца загружены.

Загрузка значений функции chi2-test

Теперь мы загрузим значения chi2-test на основе данных образца в новой таблице, сгруппированных по элементу Grp.

Выполните следующие действия.

1. В редакторе загрузки данных добавьте в конце скрипта следующее:
```
// Sample_1 data is pre-aggregated... Note: make sure you set your DecimalSep='.' at the top of the script.
Chi2_table:
LOAD Grp,
```

```
Chi2Test_chi2(Grp, Grade, Count) as chi2,
Chi2Test_df(Grp, Grade, Count) as df,
Chi2Test_p(Grp, Grade, Count) as p
resident Sample_1 group by Grp;
```

2. Щелкните для загрузки данных.

Значения chi2-test загружены в таблицу с именем Chi2_table.

Результаты

Полученные значения chi2-test можно просмотреть в просмотре модели данных в разделе **Предварительный просмотр**. Они должны выглядеть так:

Results

Grp	chi2	df	p
I	16.00	5	0.007
II	9.40	5	0.094

Создание типичного отчета t-test

Типичный отчет Стьюдента t-test может включать таблицы с результатами **Group Statistics** и **Independent Samples Test**.

В следующих разделах мы построим эти таблицы с помощью функций программы Qlik Senset-test, применяемых к двум независимым группам образцов: Observation и Comparison. Соответствующие таблицы для этих образцов будут выглядеть следующим образом:

Group statistics

Type	N	Mean	Standard Deviation	Standard Error Mean
Comparison	20	11.95	14.61245	3.2674431
Observation	20	27.15	12.507997	2.7968933

Independent Sample Test

Independent Sample Test

Type	t	df	Sig. (2-tailed)	Mean Difference	Standard Error Difference	95% Confidence Interval of the Difference (Lower)	95% Confidence Interval of the Difference (Upper)
Equal Variance not Assumed	3.534	37.116717335823	0.001	15.2	4.30101	6.48625	23.9137

Type	t	df	Sig. (2-tailed)	Mean Difference	Standard Error Difference	95% Confidence Interval of the Difference (Lower)	95% Confidence Interval of the Difference (Upper)
Equal Variance Assumed	3.534	38	0.001	15.2	4.30101	6.49306	23.9069

Загрузка данных образца

Выполните следующие действия.

1. Создайте новое приложение с новым листом и откройте этот лист.
2. Введите в редакторе загрузки данных следующее:
```
Table1:
crosstable LOAD recno() as ID, * inline [
Observation|Comparison
35|2
40|27
12|38
15|31
21|1
14|19
46|1
10|34
28|3
48|1
16|2
30|3
32|2
48|1
31|2
22|1
12|3
39|29
19|37
25|2 ] (delimiter is '|');
```
В скрипт загрузки включена функция **recno()**, поскольку для таблицы **crosstable** требуется три аргумента. Поэтому функция **recno()** просто обеспечивает дополнительный аргумент, в данном случае идентификатор для каждой строки. Без этого значения выборки **Comparison** не будут загружены.
3. Щелкните для загрузки данных.

Создание таблицы Group Statistics

Выполните следующие действия.

1. В редакторе загрузки данных щелкните , чтобы перейти в вид приложения, и нажмите ранее созданный лист.
Откроется вид листа.

2. Щелкните **Изменить лист**, чтобы изменить лист.
3. Из раздела **Диаграммы** добавьте таблицу, а из раздела **Поля** добавьте следующие выражения в качестве мер:

Примеры выражений

Метка	Выражение
N	Count(Value)
Mean	Avg(Value)
Standard Deviation	Stdev(Value)
Standard Error Mean	Sterr(Value)

4. Добавьте элемент Type в таблицу в качестве измерения.
5. Нажмите кнопку **Сортировка** и переместите элемент Type в начало списка сортировки.

Результат:

Таблица Group Statistics для этих образцов будет выглядеть следующим образом:

Group statistics

Type	N	Mean	Standard Deviation	Standard Error Mean
Comparison	20	11.95	14.61245	3.2674431
Observation	20	27.15	12.507997	2.7968933

Создание таблицы Two Independent Sample Student's T-test

Выполните следующие действия.

1. Щелкните **Изменить лист**, чтобы изменить лист.
2. Добавьте следующее выражение в таблицу в качестве измерения. `=ValueList (Dual('Equal Variance not Assumed', 0), Dual('Equal Variance Assumed', 1))`

3. Из элемента **Диаграммы** добавьте таблицу со следующими выражениями в качестве мер:

Примеры выражений

Метка	Выражение
conf	if(ValueList (Dual('Equal Variance not Assumed', 0), Dual('Equal Variance Assumed', 1)),TTest_conf(Type, Value),TTest_conf(Type, Value, 0))
t	if(ValueList (Dual('Equal Variance not Assumed', 0), Dual('Equal Variance Assumed', 1)),TTest_t(Type, Value),TTest_t(Type, Value, 0))
df	if(ValueList (Dual('Equal Variance not Assumed', 0), Dual('Equal Variance Assumed', 1)),TTest_df(Type, Value),TTest_df(Type, Value, 0))
Sig. (2-tailed)	if(ValueList (Dual('Equal Variance not Assumed', 0), Dual('Equal Variance Assumed', 1)),TTest_sig(Type, Value),TTest_sig(Type, Value, 0))
Mean Difference	TTest_dif(Type, Value)
Standard Error Difference	if(ValueList (Dual('Equal Variance not Assumed', 0), Dual('Equal Variance Assumed', 1)),TTest_sterr(Type, Value),TTest_sterr(Type, Value, 0))
95% Confidence Interval of the Difference (Lower)	if(ValueList (Dual('Equal Variance not Assumed', 0), Dual('Equal Variance Assumed', 1)),TTest_lower(Type, Value,(1-(95)/100)/2),TTest_lower (Type, Value,(1-(95)/100)/2, 0))
95% Confidence Interval of the Difference (Upper)	if(ValueList (Dual('Equal Variance not Assumed', 0), Dual('Equal Variance Assumed', 1)),TTest_upper(Type, Value,(1-(95)/100)/2),TTest_upper (Type, Value,(1-(95)/100)/2, 0))

Результат:

Independent sample test

Type	t	df	Sig. (2-tailed)	Mean Difference	Standard Error Difference	95% Confidence Interval of the Difference (Lower)	95% Confidence Interval of the Difference (Upper)
Equal Variance not Assumed	3.534	37.116717335823	0.001	15.2	4.30101	6.48625	23.9137
Equal Variance Assumed	3.534	38	0.001	15.2	4.30101	6.49306	23.9069

Примеры использования функций z-test

Функции z-test используются для обнаружения значений, связанных со статистическим анализом z-test для больших выборок данных, обычно больше 30, и где изменения известны.

В этом разделе описано, как построить визуализации с помощью данных образца, чтобы найти значения функций z-test, доступных в программе Qlik Sense. Описание синтаксиса и аргументов см. в индивидуальных темах функций диаграммы z-test.

Загрузка данных образца

Данные образца, используемые здесь, такие же, как данные, используемые в примерах функции t-test. Размер данных образца обычно считается слишком маленьким для анализа z-критериев, но он достаточен для иллюстрации использования различных функций z-test в программе Qlik Sense.

Выполните следующие действия.

1. Создайте новое приложение с новым листом и откройте этот лист.

   Если создано приложение для функций t-test, его можно использовать и создать новый лист для этих функций.

2. Введите в редакторе загрузки данных следующее:

```
Table1:
crosstable LOAD recno() as ID, * inline [
Observation|Comparison
35|2
40|27
12|38
15|31
21|1
14|19
46|1
10|34
28|3
48|1
16|2
30|3
32|2
48|1
31|2
22|1
12|3
39|29
19|37
25|2 ] (delimiter is '|');
```

   В скрипт загрузки включена функция **recno()**, поскольку для таблицы **crosstable** требуется три аргумента. Поэтому функция **recno()** просто обеспечивает дополнительный аргумент, в данном случае идентификатор для каждой строки. Без этого значения выборки **Comparison** не будут загружены.

3. Щелкните  для загрузки данных.

Создание визуализаций функции диаграммы z-test

Выполните следующие действия.

1. В редакторе загрузки данных щелкните , чтобы перейти в вид приложения, и нажмите лист, созданный при загрузке данных.
   Откроется вид листа.
2. Щелкните **Изменить лист**, чтобы изменить лист.
3. Из раздела **Диаграммы** добавьте таблицу, а из раздела **Поля** добавьте элемент Type в качестве измерения.
4. Добавьте следующие выражения в таблицу в качестве мер.

Примеры выражений

Метка	Выражение
ZTest Conf	ZTest_conf(Value)
ZTest Dif	ZTest_dif(Value)
ZTest Sig	ZTest_sig(Value)
ZTest Sterr	ZTest_sterr(Value)
ZTest Z	ZTest_z(Value)

*Может возникнуть необходимость откорректировать формат чисел мер, чтобы увидеть значимые значения. Таблицу будет легче считывать, если для большинства мер установить значение формата чисел **Число>Простой** вместо **Авто**. К примеру, для ZTest Sig используйте формат чисел: **Пользовательский**, затем настройте для образца формата параметр **# ##**.*

Результат:

Полученная в результате таблица для функций z-test для данных образца будет содержать следующие значения.

Результирующая таблица

Type	ZTest Conf	ZTest Dif	ZTest Sig	ZTest Sterr	ZTest Z
Comparison	6.40	11.95	0.000123	3.27	3.66
Value	5.48	27.15	0.001	2.80	9.71

Создание визуализаций функции диаграммы z-testw

Функции z-testw используются, когда серии вводимых данных встречаются в формате двух столбцов. Выражения требуют значение для аргумента weight. Во всех этих примерах используется значение 2, но можно использовать выражение, которое определит значение для элемента weight при каждом просмотре.

Примеры и результаты:

При использовании одинаковых данных образца и формата чисел, как для функций z-test, результирующая таблица для функций z-testw будет содержать следующие значения:

Результирующая таблица

Type	ZTestw Conf	ZTestw Dif	ZTestw Sig	ZTestw Sterr	ZTestw Z
Comparison	3.53	2.95	5.27e-005	1.80	3.88
Value	2.97	34.25	0	4.52	20.49

Строковые функции агрегирования

В этом разделе описаны функции агрегирования, относящиеся к строкам.

Каждая функция подробно описана после обзора. Также можно щелкнуть имя функции в синтаксисе, чтобы получить немедленный доступ к подробной информации об этой конкретной функции.

Строковые функции агрегирования в скрипте загрузки данных

Concat

Функция **Concat()** используется для объединения строковых значений. Эта функция скрипта возвращает агрегированное объединение строк всех значений выражения, повторяемого в нескольких записях, как определено предложением **group by**.

```
Concat ([ distinct ] expression [, delimiter [, sort-weight]])
```

FirstValue

Функция **FirstValue()** возвращает значение, загруженное первым из записей, определенных выражением, отсортированным по предложению **group by**.

Эта функция доступна только как функция скрипта.

```
FirstValue  (expression)
```

LastValue

Функция **LastValue()** возвращает значение, загруженное последним из записей, определенных выражением, отсортированным по предложению **group by**.

Эта функция доступна только как функция скрипта.

```
LastValue  (expression)
```

MaxString

Функция **MaxString()** находит строковые значения в выражении и возвращает последнее по алфавиту текстовое значение в выборке, определенной условием **group by**.

```
MaxString (expression )
```

MinString

Функция **MinString()** находит строковые значения в выражении и возвращает первое по алфавиту текстовое значение в выборке, определенной условием **group by**.

```
MinString (expression )
```

Строковые функции агрегирования в диаграммах

Следующие функции диаграммы доступны для агрегирования строк в диаграммах.

Concat

Функция **Concat()** используется для объединения строковых значений. Функция возвращает агрегированное объединение строк всех значений выражения, оцениваемого по каждому измерению.

```
Concat — функция диаграммы({[SetExpression] [DISTINCT] [TOTAL [<fld{, fld}>]]
string[, delimiter[, sort_weight]])
```

MaxString

Функция **MaxString()** находит строковые значения в выражении или поле и возвращает последнее по алфавиту текстовое значение.

```
MaxString — функция диаграммы({[SetExpression] [TOTAL [<fld{, fld}>]]} expr)
```

MinString

Функция **MinString()** находит строковые значения в выражении или поле и возвращает первое по алфавиту текстовое значение.

```
MinString — функция диаграммы({[SetExpression] [TOTAL [<fld {, fld}>]]} expr)
```

Concat

Функция **Concat()** используется для объединения строковых значений. Эта функция скрипта возвращает агрегированное объединение строк всех значений выражения, повторяемого в нескольких записях, как определено предложением **group by**.

Синтаксис:

```
Concat ([ distinct ] string [, delimiter [, sort-weight]])
```

Возвращаемые типы данных: строка

Аргументы:

Выражение или поле, содержащее строку для обработки.

Аргументы

Аргумент	Описание
string	Выражение или поле, содержащее строку для обработки.
delimiter	Каждое значение может быть разделено строкой, найденной в delimiter.
sort-weight	Порядок объединения может быть определен значением измерения **sort-weight** при его наличии со строкой, соответствующей наименьшему значению, появляющемуся в объединении первым.
distinct	Если слово **distinct** указано перед выражением, все дубликаты будут проигнорированы.

Примеры и результаты:

Добавьте образец скрипта в свое приложение и запустите. Чтобы увидеть результаты, добавьте поля, перечисленные в столбце результатов, в лист приложения.

Примеры и результаты

Пример	Результат	Результаты после добавления на лист

`TeamData:`
`LOAD * inline [`
`SalesGroup|Team|Date|Amount`
`East|Gamma|01/05/2013|20000`
`East|Gamma|02/05/2013|20000`
`West|Zeta|01/06/2013|19000`
`East|Alpha|01/07/2013|25000`
`East|Delta|01/08/2013|14000`
`West|Epsilon|01/09/2013|17000`
`West|Eta|01/10/2013|14000`
`East|Beta|01/11/2013|20000`
`West|Theta|01/12/2013|23000`
`] (delimiter is '|');`
 	
`Concat1:`
`LOAD SalesGroup,Concat(Team) as TeamConcat1`
`Resident TeamData Group By SalesGroup;`
	SalesGroup
	East
	West
		TeamConcat1
		AlphaBetaDeltaGammaGamma
		EpsilonEtaThetaZeta

При условии, что таблица **TeamData** загружается, как в предыдущем примере:
 	
`LOAD SalesGroup,Concat(distinct Team,'-')`
`as TeamConcat2 Resident TeamData Group By`
`SalesGroup;`
	SalesGroup
	East
	West
		TeamConcat2
		Alpha-Beta-Delta-Gamma
		Epsilon-Eta-Theta-Zeta

Пример	Результат	Результаты после добавления на лист
При условии, что таблица **TeamData** загружается, как в предыдущем примере. Поскольку аргумент для элемента **sort-weight** добавлен, порядок результатов определяется значением измерения Amount: `LOAD SalesGroup,Concat(distinct Team,'-',Amount) as TeamConcat2 Resident TeamData Group By SalesGroup;`	SalesGroup East West	TeamConcat2 Delta-Beta-Gamma-Alpha Eta-Epsilon-Zeta-Theta

Concat – функция диаграммы

Функция **Concat()** используется для объединения строковых значений. Функция возвращает агрегированное объединение строк всех значений выражения, оцениваемого по каждому измерению.

Синтаксис:

```
Concat({[SetExpression] [DISTINCT] [TOTAL [<fld{, fld}>]]} string[, delimiter [, sort_weight]])
```

Возвращаемые типы данных: строка

Аргументы:

Аргументы

Аргумент	Описание
string	Выражение или поле, содержащее строку для обработки.
delimiter	Каждое значение может быть разделено строкой, найденной в delimiter.
sort-weight	Порядок объединения может быть определен значением измерения **sort-weight** при его наличии со строкой, соответствующей наименьшему значению, появляющемуся в объединении первым.
SetExpression	По умолчанию функция агрегирования агрегирует множество возможных записей, определенных выборкой. Альтернативный набор записей может быть определен выражением анализа множества.
DISTINCT	Если слово **DISTINCT** указывается до аргументов функции, все дубликаты, возникшие в результате оценки аргументов функции, будут проигнорированы.

Аргумент	Описание
TOTAL	Если слово **TOTAL** стоит перед аргументами функции, вычисление выполняется по всем возможным значениям, указанным в текущих выборках, а не только в тех, которые относятся к значению текущего измерения, т. е. измерения диаграммы игнорируются. При использовании выражения **TOTAL [<fld {.fld}>]**, где префикс **TOTAL** предшествует списку из одного или нескольких имен полей, выступающих в качестве подмножества переменных измерения диаграммы, создается подмножество всех возможных значений.

Примеры и результаты:

Results table

SalesGroup	Amount	Concat(Team)	Concat(TOTAL <SalesGroup> Team)
East	25000	Alpha	AlphaBetaDeltaGammaGamma
East	20000	BetaGammaGamma	AlphaBetaDeltaGammaGamma
East	14000	Delta	AlphaBetaDeltaGammaGamma
West	17000	Epsilon	EpsilonEtaThetaZeta
West	14000	Eta	EpsilonEtaThetaZeta
West	23000	Theta	EpsilonEtaThetaZeta
West	19000	Zeta	EpsilonEtaThetaZeta

Примеры функции

Пример	Результат
`Concat(Team)`	Таблица состоит из измерений SalesGroup и Amount и вариантов меры Concat (Team). Игнорируя результат «Итоги», обратите внимание, что несмотря на то, что существуют данные для восьми значений элемента Team, разбросанные по двум значениям элемента SalesGroup, единственным результатом меры Concat(Team), которая объединяет больше одного значения строки Team в таблице, является строка, содержащая измерение Amount 20 000, результатом которого является BetaGammaGamma. Это обусловлено тем, что во входных данных существует три значения для измерения Amount 20 000. Все прочие результаты остаются не связанными, если мера заполнена по всем измерениям, поскольку существует только одно значение элемента Team для каждой комбинации элементов SalesGroup и Amount.
`Concat (DISTINCT Team,', ')`	Элементы Beta, Gamma, поскольку классификатор DISTINCT означает, что результат дубликата Gamma игнорируется. Также аргумент разделителя определяется как запятая, после которой стоит пробел.

Пример	Результат
`Concat (TOTAL <SalesGroup> Team)`	Все значения строки для всех значений элемента Team объединяются, если используется классификатор TOTAL. Если указана выборка поля <SalesGroup>, результаты делятся на два значения измерения SalesGroup. Для элемента SalesGroupEast результатами являются AlphaBetaDeltaGammaGamma. Для элемента SalesGroupWest результатами являются EpsilonEtaThetaZeta.
`Concat (TOTAL <SalesGroup> Team,';', Amount)`	При добавлении аргумента для элемента **sort-weight**: Amount результаты упорядочиваются значением измерения Amount. Результатом становятся значения DeltaBetaGammaGammaAlpha и EtaEpsilonZEtaTheta.

Данные, используемые в примере:

```
TeamData:
LOAD * inline [
SalesGroup|Team|Date|Amount
East|Gamma|01/05/2013|20000
East|Gamma|02/05/2013|20000
West|Zeta|01/06/2013|19000
East|Alpha|01/07/2013|25000
East|Delta|01/08/2013|14000
West|Epsilon|01/09/2013|17000
West|Eta|01/10/2013|14000
East|Beta|01/11/2013|20000
West|Theta|01/12/2013|23000
] (delimiter is '|');
```

FirstValue

Функция **FirstValue()** возвращает значение, загруженное первым из записей, определенных выражением, отсортированным по предложению **group by**.

Эта функция доступна только как функция скрипта.

Синтаксис:

```
FirstValue ( expr)
```

Возвращаемые типы данных: двойное значение

Аргументы:

Аргументы

Аргумент	Описание
expr	Выражение или поле, содержащее данные для измерения.

Ограничения:

Если текстовые значения не найдены, возвращается значение NULL.

Примеры и результаты:

Добавьте образец скрипта в свое приложение и запустите. Чтобы увидеть результаты, добавьте поля, перечисленные в столбце результатов, в лист приложения.

Результирующие данные

Пример	Результат	Результаты на листе

`TeamData:`
`LOAD * inline [`
`SalesGroup|Team|Date|Amount`
`East|Gamma|01/05/2013|20000`
`East|Gamma|02/05/2013|20000`
`West|Zeta|01/06/2013|19000`
`East|Alpha|01/07/2013|25000`
`East|Delta|01/08/2013|14000`
`West|Epsilon|01/09/2013|17000`
`West|Eta|01/10/2013|14000`
`East|Beta|01/11/2013|20000`
`West|Theta|01/12/2013|23000`
`] (delimiter is '|');`
 	
`FirstValue1:`
`LOAD SalesGroup,FirstValue(Team) as FirstTeamLoaded`
`Resident TeamData Group By SalesGroup;`
	SalesGroup
	East
	West
		FirstTeamLoaded
		Gamma
		Zeta

LastValue

Функция **LastValue()** возвращает значение, загруженное последним из записей, определенных выражением, отсортированным по предложению **group by**.

Эта функция доступна только как функция скрипта.

Синтаксис:

```
LastValue ( expr )
```

Возвращаемые типы данных: двойное значение

Аргументы:

Аргументы

Аргумент	Описание
expr	Выражение или поле, содержащее данные для измерения.

Ограничения:

Если текстовые значения не найдены, возвращается значение NULL.

Примеры и результаты:

Добавьте образец скрипта в свое приложение и запустите. Затем добавьте на лист приложения как минимум поля, указанные в столбце с результатами, чтобы увидеть результаты.

Чтобы столбец с результатами выглядел так же, как столбец с результатами ниже, на панели свойств в разделе «Сортировка» переключите параметр с «Авто» на «Пользовательский», а затем отмените выбор числовой сортировки и сортировки в алфавитном порядке.

Пример	Результат	Результат с пользовательской сортировкой

`TeamData:`
`LOAD * inline [`
`SalesGroup|Team|Date|Amount`
`East|Gamma|01/05/2013|20000`
`East|Gamma|02/05/2013|20000`
`West|Zeta|01/06/2013|19000`
`East|Alpha|01/07/2013|25000`
`East|Delta|01/08/2013|14000`
`West|Epsilon|01/09/2013|17000`
`West|Eta|01/10/2013|14000`
`East|Beta|01/11/2013|20000`
`West|Theta|01/12/2013|23000`
`] (delimiter is '|');`
 	
`LastValue1:`
`LOAD SalesGroup,LastValue(Team) as LastTeamLoaded`
`Resident TeamData Group By SalesGroup;`
	SalesGroup
	East
	West
		LastTeamLoaded
		Beta
		Theta

MaxString

Функция **MaxString()** находит строковые значения в выражении и возвращает последнее по алфавиту текстовое значение в выборке, определенной условием **group by**.

Синтаксис:

```
MaxString ( expr )
```

Возвращаемые типы данных: двойное значение

Аргументы:

Аргумент	Описание
expr	Выражение или поле, содержащее данные для измерения.

Ограничения:

Если текстовые значения не найдены, возвращается значение NULL.

Примеры и результаты:

Добавьте образец скрипта в свое приложение и запустите. Чтобы увидеть результаты, добавьте поля, перечисленные в столбце результатов, в лист приложения.

Пример	Результат	
TeamData: LOAD * inline [SalesGroup|Team|Date|Amount East|Gamma|01/05/2013|20000 East|Gamma|02/05/2013|20000 West|Zeta|01/06/2013|19000 East|Alpha|01/07/2013|25000 East|Delta|01/08/2013|14000 West|Epsilon|01/09/2013|17000 West|Eta|01/10/2013|14000 East|Beta|01/11/2013|20000 West|Theta|01/12/2013|23000] (delimiter is '|'); Concat1: LOAD SalesGroup,MaxString(Team) as MaxString1 Resident TeamData Group By SalesGroup;	SalesGroup East West	MaxString1 Gamma Zeta
При условии, что таблица **TeamData** загружается как в предыдущем примере, а ваш скрипт загрузки данных имеет оператор SET: SET DateFormat='DD/MM/YYYY';': LOAD SalesGroup,MaxString(Date) as MaxString2 Resident TeamData Group By SalesGroup;	SalesGroup East West	MaxString2 01/11/2013 01/12/2013

MaxString – функция диаграммы

Функция **MaxString()** находит строковые значения в выражении или поле и возвращает последнее по алфавиту текстовое значение.

Синтаксис:

```
MaxString({[SetExpression] [TOTAL [<fld{, fld}>]]} expr)
```

Возвращаемые типы данных: двойное значение

Аргументы:

Аргументы

Аргумент	Описание
expr	Выражение или поле, содержащее данные для измерения.
SetExpression	По умолчанию функция агрегирования агрегирует множество возможных записей, определенных выборкой. Альтернативный набор записей может быть определен выражением анализа множества.
TOTAL	Если слово **TOTAL** стоит перед аргументами функции, вычисление выполняется по всем возможным значениям, указанным в текущих выборках, а не только в тех, которые относятся к значению текущего измерения, т. е. измерения диаграммы игнорируются. При использовании выражения **TOTAL [<fld {.fld}>]**, где префикс **TOTAL** предшествует списку из одного или нескольких имен полей, выступающих в качестве подмножества переменных измерения диаграммы, создается подмножество всех возможных значений.

Ограничения:

Если выражение не содержит значений со строковым представлением, возвращается значение NULL.

Примеры и результаты:

Результирующая таблица

SalesGroup	Amount	MaxString(Team)	MaxString(Date)
East	14000	Delta	2013/08/01
East	20000	Gamma	2013/11/01
East	25000	Alpha	2013/07/01
West	14000	Eta	2013/10/01
West	17000	Epsilon	2013/09/01
West	19000	Zeta	2013/06/01
West	23000	Theta	2013/12/01

Примеры функции

Пример	Результат
`MaxString (Team)`	Существует три значения 20 000 для измерения Amount: два измерения элемента Gamma (с различными датами), и одно элемента Beta. Таким образом, результатом меры MaxString (Team) является элемент Gamma, поскольку это наибольшее значение в отсортированных строках.
`MaxString (Date)`	2013/11/01 является самым большим значением Date из трех, ассоциированных с измерением Amount. Так предполагается, что ваш скрипт имеет оператор SET `SET DateFormat='YYYY-MM-DD';'`

Данные, используемые в примере:

```
TeamData:
LOAD * inline [
SalesGroup|Team|Date|Amount
East|Gamma|01/05/2013|20000
East|Gamma|02/05/2013|20000
West|Zeta|01/06/2013|19000
East|Alpha|01/07/2013|25000
East|Delta|01/08/2013|14000
West|Epsilon|01/09/2013|17000
West|Eta|01/10/2013|14000
East|Beta|01/11/2013|20000
West|Theta|01/12/2013|23000
] (delimiter is '|');
```

MinString

Функция **MinString()** находит строковые значения в выражении и возвращает первое по алфавиту текстовое значение в выборке, определенной условием **group by**.

Синтаксис:

```
MinString ( expr )
```

Возвращаемые типы данных: двойное значение

Аргументы:

Аргументы

Аргумент	Описание
expr	Выражение или поле, содержащее данные для измерения.

Ограничения:

Если текстовые значения не найдены, возвращается значение NULL.

Примеры и результаты:

Добавьте образец скрипта в свое приложение и запустите. Чтобы увидеть результаты, добавьте поля, перечисленные в столбце результатов, в лист приложения.

Результирующие данные

Пример	Результат	
TeamData: LOAD * inline [SalesGroup|Team|Date|Amount East|Gamma|01/05/2013|20000 East|Gamma|02/05/2013|20000 West|Zeta|01/06/2013|19000 East|Alpha|01/07/2013|25000 East|Delta|01/08/2013|14000 West|Epsilon|01/09/2013|17000 West|Eta|01/10/2013|14000 East|Beta|01/11/2013|20000 West|Theta|01/12/2013|23000] (delimiter is '|'); Concat1: LOAD SalesGroup,MinString(Team) as MinString1 Resident TeamData Group By SalesGroup;	SalesGroup East West	MinString1 Alpha Epsilon
При условии, что таблица **TeamData** загружается как в предыдущем примере, а ваш скрипт загрузки данных имеет оператор SET: SET DateFormat='DD/MM/YYYY';': LOAD SalesGroup,MinString(Date) as MinString2 Resident TeamData Group By SalesGroup;	SalesGroup East West	MinString2 01/05/2013 01/06/2013

MinString – функция диаграммы

Функция **MinString()** находит строковые значения в выражении или поле и возвращает первое по алфавиту текстовое значение.

Синтаксис:

```
MinString({[SetExpression] [TOTAL [<fld {, fld}>]]} expr)
```

Возвращаемые типы данных: двойное значение

Аргументы:

Аргументы

Аргумент	Описание
expr	Выражение или поле, содержащее данные для измерения.

Аргумент	Описание
SetExpression	По умолчанию функция агрегирования агрегирует множество возможных записей, определенных выборкой. Альтернативный набор записей может быть определен выражением анализа множества.
TOTAL	Если слово **TOTAL** стоит перед аргументами функции, вычисление выполняется по всем возможным значениям, указанным в текущих выборках, а не только в тех, которые относятся к значению текущего измерения, т. е. измерения диаграммы игнорируются. При использовании выражения **TOTAL [<fld {.fld}>]**, где префикс **TOTAL** предшествует списку из одного или нескольких имен полей, выступающих в качестве подмножества переменных измерения диаграммы, создается подмножество всех возможных значений.

Примеры и результаты:

Данные образца

SalesGroup	Amount	MinString(Team)	MinString(Date)
East	14000	Delta	2013/08/01
East	20000	Beta	2013/05/01
East	25000	Alpha	2013/07/01
West	14000	Eta	2013/10/01
West	17000	Epsilon	2013/09/01
West	19000	Zeta	2013/06/01
West	23000	Theta	2013/12/01

Примеры функции

Примеры	Результаты
`MinString (Team)`	Существует три значения 20000 для измерения Amount: два измерения элемента Gamma (с различными датами), и одно элемента Beta. Таким образом, результатом меры MinString (Team) является элемент Beta, поскольку это первое значение в отсортированных строках.
`MinString (Date)`	2013/11/01 является самым ранним значением Date из трех, ассоциированных с измерением Amount. Так предполагается, что ваш скрипт имеет оператор SET `SET DateFormat='YYYY-MM-DD';'`

Данные, используемые в примере:

```
TeamData:
LOAD * inline [
SalesGroup|Team|Date|Amount
East|Gamma|01/05/2013|20000
```

```
East|Gamma|02/05/2013|20000
West|Zeta|01/06/2013|19000
East|Alpha|01/07/2013|25000
East|Delta|01/08/2013|14000
West|Epsilon|01/09/2013|17000
West|Eta|01/10/2013|14000
East|Beta|01/11/2013|20000
West|Theta|01/12/2013|23000
] (delimiter is '|');
```

Функции синтетических измерений

Синтетическое измерение создано в приложении из значений, созданных из функций синтетического измерения, а не напрямую из полей в модели данных. Если значения, созданные функцией синтетического измерения используются в диаграмме как вычисляемые измерения, создается синтетическое измерение. Синтетические измерения позволяют создавать, например, диаграммы с измерениями со значениями, происходящими от ваших данных, т. е. динамические измерения.

Выборки не влияют на синтетические измерения.

Следующие функции синтетических измерений можно использовать в диаграммах.

ValueList

Функция **ValueList()** возвращает набор перечисленных значений, в результате чего при использовании в вычисляемом измерении образуется синтетическое измерение.

```
ValueList – функция диаграммы (v1 {, Expression})
```

ValueLoop

Функция ValueLoop() возвращает набор повторяемых значений, в результате чего при использовании в вычисляемом измерении образуется синтетическое измерение.

```
ValueLoop – функция диаграммы(from [, to [, step ]])
```

ValueList – функция диаграммы

Функция **ValueList()** возвращает набор перечисленных значений, в результате чего при использовании в вычисляемом измерении образуется синтетическое измерение.

*В диаграммах с синтетическим измерением, созданным с помощью функции **ValueList**, можно указать ссылку на значение измерения, соответствующее определенной ячейке выражения. Для этого необходимо повторно запустить функцию **ValueList** с теми же параметрами в выражении диаграммы. Разумеется, функцию можно использовать в любом месте на макете, но, помимо использования для синтетических измерений, эта функция будет иметь смысл только внутри функции агрегирования.*

Выборки не влияют на синтетические измерения.

Синтаксис:

```
ValueList(v1 {,...})
```

Возвращаемые типы данных: двойное значение

Аргументы:

Аргументы

Аргумент	Описание
v1	Статическое значение (обычно выраженное строкой, но возможно и числом).
{,...}	Дополнительный список статических значений.

Примеры и результаты:

Примеры функции

Пример	Результат
`ValueList ('Number of Orders', 'Average Order Size', 'Total Amount')`	При использовании для создания измерения в таблице, например, появляются три значения в строках в виде меток строк в таблице. В выражении на них может быть дана ссылка.
`=IF( ValueList ('Number of Orders', 'Average Order Size', 'Total Amount') = 'Number of Orders', count (SaleID), IF( ValueList ('Number of Orders', 'Average Order Size', 'Total Amount') = 'Average Order Size', avg (Amount), sum (Amount) ))`	Это выражение берет значения из созданного измерения и дает на них ссылку во вложенном операторе IF, как значения, вводимые в три функции агрегирования: **ValueList()** Created dimension | Year | Added expression **522.00** Number of Orders | 2012 | 5.00 Number of Orders | 2013 | 7.00 Average Order Size | 2012 | 13.20 Average Order Size | 2013 | 15.43 Total Amount | 2012 | 66.00 Total Amount | 2013 | 108.00

Данные, используемые в примерах:

```
SalesPeople:
LOAD * INLINE [
SaleID|SalesPerson|Amount|Year
1|1|12|2013
2|1|23|2013
```

```
3|1|17|2013
4|2|9|2013
5|2|14|2013
6|2|29|2013
7|2|4|2013
8|1|15|2012
9|1|16|2012
10|2|11|2012
11|2|17|2012
12|2|7|2012
] (delimiter is '|');
```

ValueLoop – функция диаграммы

Функция ValueLoop() возвращает набор повторяемых значений, в результате чего при использовании в вычисляемом измерении образуется синтетическое измерение. Диапазон генерированных значений ограничивается значениями **from** и **to**, включая промежуточные значения в приращениях шага.

*В диаграммах с синтетическим измерением, созданным с помощью функции **ValueLoop**, можно указать ссылку на значение измерения, соответствующее определенной ячейке выражения. Для этого необходимо повторно запустить функцию **ValueLoop** с теми же параметрами в выражении диаграммы. Разумеется, функцию можно использовать в любом месте на макете, но, помимо использования для синтетических измерений, эта функция будет иметь смысл только внутри функции агрегирования.*

Выборки не влияют на синтетические измерения.

Синтаксис:

```
ValueLoop(from [, to [, step ]])
```

Возвращаемые типы данных: двойное значение

Аргументы:

Аргументы

Аргументы	Описание
from	Необходимо создать начальное значение из ряда значений.
to	Необходимо создать конечное значение из ряда значений.
step	Размер приращения между значениями.

Примеры и результаты:

Примеры функции

Пример	Результат
`ValueLoop (1, 10)`	Создается измерение в таблице, например, такое, которое может быть использовано для обеспечения меток с числами. В этом примере в результате образованы значения от 1 до 10. В выражении на эти значения может быть дана ссылка.
`ValueLoop (2, 10,2)`	В этом примере в результате образованы значения 2, 4, 6, 8, и 10, поскольку аргумент step имеет значение 2.

Вложенные агрегирования

Возможны ситуации, когда необходимо применить агрегирование к результату другого агрегирования. Это называется вложенными агрегированиями.

Большинство выражений диаграммы не могут содержать вложенные агрегирования. Однако можно создавать вложенные агрегирования, если используется квалификатор **TOTAL** во внутренней функции агрегирования.

Допустимо не более 100 уровней вложения.

Вложенные агрегирования с классификатором TOTAL

Пример:

Например, необходимо вычислить сумму поля **Sales**, но должны быть включены только транзакции с элементом **OrderDate**, равным последнему году. Последний год может быть получен через функцию агрегирования **`Max(TOTAL Year(OrderDate))`**.

В результате следующего агрегирования будет получен желаемый результат.

```
Sum(If(Year(OrderDate)=Max(TOTAL Year(OrderDate)), Sales))
```

Qlik Sense требует включения квалификатора **TOTAL** при этом типе вложения. Он требуется для выполнения необходимого сравнения. Этот тип вложенности часто требуется и должен использоваться во всех подходящих случаях.

См. также:

Aggr – функция диаграммы (page 438)

5.3 Aggr – функция диаграммы

Функция **Aggr()** возвращает диапазон значений выражения, вычисленный по указанному измерению или измерениям. Например, максимальное значение продаж по каждому клиенту, по региону.

Функция **Aggr** используется для вложенных агрегирований, в которых ее первый параметр (внутреннее агрегирование) вычисляется один раз для каждого значения измерения. Измерения указываются во втором и последующих параметрах.

Кроме того, функция **Aggr** должна заключаться во внешнюю функцию агрегирования, которая использует массив результатов из функции **Aggr** в качестве ввода в агрегирование, в которое она вложена.

Синтаксис:

```
Aggr({SetExpression}[DISTINCT] [NODISTINCT ] expr, StructuredParameter{,
StructuredParameter})
```

Возвращаемые типы данных: двойное значение

Аргументы:

Аргументы

Аргумент	Описание
expr	Выражение, состоящее из функции агрегирования. По умолчанию функция агрегирования агрегирует множество возможных записей, определенных выборкой.
StructuredParameter	StructuredParameter представляет собой измерение, к которому в некоторых случаях добавляется критерий сортировки следующего формата: `(Dimension(Sort-type, Ordering))` Измерение представляет собой одиночное поле, оно не может быть выражением. Измерение предназначено для определения диапазона значений, для которых вычисляется выражение Aggr. При включении критериев сортировки осуществляется сортировка созданного функцией Aggr диапазона значений, вычисляемого для измерения. Это имеет значение, если порядок сортировки влияет на результат выражения, содержащего функцию Aggr. Сведения о порядке использования критериев сортировки см. в разделе Добавление критериев сортировки к измерению в составе структурированного параметра.

Аргумент	Описание
SetExpression	По умолчанию функция агрегирования агрегирует множество возможных записей, определенных выборкой. Альтернативный набор записей может быть определен выражением анализа множества.
DISTINCT	Если перед аргументом выражения стоит классификатор **distinct** или его вообще нет, то каждая комбинация значений измерений будет создавать только одно возвращаемое значение. Это обычный способ создания агрегирований – каждая комбинация значений измерений будет воспроизводить одну линию в диаграмме.
NODISTINCT	Если перед аргументом выражения стоит классификатор **nodistinct**, то каждая комбинация значений измерений может создавать несколько возвращаемых значений в зависимости от базовой структуры данных. Если измерение только одно, функция **aggr** вернет массив с тем же количеством элементов, что и строк в исходных данных.

Базовые функции агрегирования, такие как **Sum**, **Min** и **Avg**, возвращают отдельное числовое значение, тогда как функцию Aggr() можно сравнить с созданием временного промежуточного набора результатов (виртуальной таблицы), с помощью которого можно провести другое агрегирование. Например, при вычислении среднего значения объема продаж путем сложения сумм продаж по клиентам в операторе **Aggr()** и затем вычисления среднего значения по суммированным результатам: **Avg(TOTAL Aggr(Sum(Sales),Customer))**.

Используйте функцию Aggr() в вычисляемых измерениях, если необходимо создать агрегирование вложенной диаграммы на различных уровнях.

Ограничения:

Каждое измерение функции Aggr() может быть одиночным полем и не может быть выражением (вычисляемое измерение).

Добавление критериев сортировки к измерению в составе структурированного параметра

Базовая форма аргумента StructuredParameter в синтаксисе функции Aggr представляет собой одиночное измерение. Выражение Aggr(Sum(Sales, Month)) служит для вычисления итогового значения продаж за каждый месяц. Однако если выражение входит в состав другой функции агрегирования, в случае отсутствия критериев сортировки результат вычисления может быть неудовлетворительным. Это вызвано тем, что сортировка некоторых изменений может осуществляться в числовом или алфавитном порядке.

В аргументе StructuredParameter функции Aggr можно указать критерии сортировки измерения в составе выражения. Таким образом, к виртуальной таблице, созданной функцией Aggr, применяется определенный порядок сортировки.

Аргумент StructuredParameter имеет следующий синтаксис:

```
(FieldName, (Sort-type, Ordering))
```

Структурированные параметры поддерживают создание вложений:

```
(FieldName, (FieldName2, (Sort-type, Ordering)))
```

Доступны следующие типы сортировки: NUMERIC, TEXT, FREQUENCY или LOAD_ORDER.

С каждым типом сортировки связаны следующие типы упорядочивания:

Допустимые типы упорядочивания

Тип сортировки	Допустимые типы упорядочивания
NUMERIC	ASCENDING, DESCENDING или REVERSE
TEXT	ASCENDING, A2Z, DESCENDING, REVERSE или Z2A
FREQUENCY	DESCENDING, REVERSE или ASCENDING
LOAD_ORDER	ASCENDING, ORIGINAL, DESCENDING или REVERSE

Типы упорядочивания REVERSE и DESCENDING эквивалентны друг другу.

Для типа сортировки TEXT типы упорядочивания ASCENDING и A2Z являются эквивалентными; также эквивалентны типы упорядочивания DESCENDING, REVERSE и Z2A.

Для типа сортировки LOAD_ORDER типы упорядочивания ASCENDING и ORIGINAL являются эквивалентными.

Примеры: Выражения диаграммы с использованием Aggr

Примеры. Выражения диаграммы

Пример выражения диаграммы 1

Скрипт загрузки

Загрузите следующие данные через встроенную загрузку в редакторе загрузки данных, чтобы создать пример с выражениями диаграммы, показанный ниже.

```
ProductData:  LOAD * inline [ Customer|Product|UnitSales|UnitPrice Astrida|AA|4|16
Astrida|AA|10|15 Astrida|BB|9|9 Betacab|BB|5|10 Betacab|CC|2|20 Betacab|DD|25|25
Canutility|AA|8|15 Canutility|CC|0|19 ] (delimiter is '|');
```

Выражение диаграммы

Создайте визуализацию ключевых показателей эффективности на листе Qlik Sense. Добавьте следующее выражение в ключевой показатель эффективности в качестве меры:

```
Avg(Aggr(Sum(UnitSales*UnitPrice), Customer))
```

Результат

376.7

Объяснение

Выражение `Aggr(Sum(UnitSales*UnitPrice), Customer)` вычисляет общее значение продаж для значения **Customer** и возвращает несколько значений: 295, 715 и 120 для трех значений **Customer**.

По сути, мы построили временный список значений, не создавая отдельную таблицу или столбец с этими значениями.

Данные значения выполняют функцию вводимых данных для функции **Avg()**, служащей для вычисления среднего значения продаж, 376.7.

Пример выражения диаграммы 2

Скрипт загрузки

Загрузите следующие данные через встроенную загрузку в редакторе загрузки данных, чтобы создать пример с выражениями диаграммы, показанный ниже.

```
ProductData:  LOAD * inline [ Customer|Product|UnitSales|UnitPrice Astrida|AA|4|16
Astrida|AA|10|15 Astrida|BB|10|15 Astrida|BB|9|9 Betacab|BB|5|10 Betacab|BB|7|12
Betacab|CC|2|22 Betacab|CC|4|20 Betacab|DD|25|25 Canutility|AA|8|15 Canutility|AA|5|11
Canutility|CC|0|19 ] (delimiter is '|');
```

Выражение диаграммы

Создайте на листе Qlik Sense визуализацию таблицы с измерениями **Customer**, **Product**, **UnitPrice** и **UnitSales**. Добавьте в таблицу следующее выражение в качестве меры:

```
Aggr(NODISTINCT Max(UnitPrice), Customer, Product)
```

Результат

Customer	Product	UnitPrice	UnitSales	Aggr(NODISTINCT Max(UnitPrice), Customer, Product)
Astrida	AA	15	10	16
Astrida	AA	16	4	16
Astrida	BB	9	9	15
Astrida	BB	15	10	15
Betacab	BB	10	5	12
Betacab	BB	12	7	12
Betacab	CC	20	4	22
Betacab	CC	22	2	22
Betacab	DD	25	25	25
Canutility	AA	11	5	15

Customer	Product	UnitPrice	UnitSales	Aggr(NODISTINCT Max(UnitPrice), Customer, Product)
Canutility	AA	15	8	15
Canutility	CC	19	0	19

Объяснение

Массив значений: 16, 16, 15, 15, 12, 12, 22, 22, 25, 15, 15 и 19. Классификатор **nodistinct** означает, что диапазон содержит по одному элементу для каждой строки в данных источника: каждый является максимальным значением **UnitPrice** для каждого элемента **Customer** и **Product**.

Пример выражения диаграммы 3

Скрипт загрузки

Загрузите следующие данные через встроенную загрузку в редакторе загрузки данных, чтобы создать пример с выражениями диаграммы, показанный ниже.

```
Set vNumberOfOrders = 1000;  OrderLines: Load   RowNo() as OrderLineID,         OrderID,        OrderDate,
Round((Year(OrderDate)-2005)*1000*Rand()*Rand()*Rand1) as Sales         While Rand()<=0.5 or IterNo
()=1; Load * Where OrderDate<=Today(); Load     Rand() as Rand1,        Date(MakeDate(2013)+Floor
((365*4+1)*Rand())) as OrderDate,       RecNo() as OrderID      Autogenerate vNumberOfOrders;
Calendar: Load distinct         Year(OrderDate) as Year,        Month(OrderDate) as Month,      OrderDate
Resident OrderLines;
```

Выражения диаграммы

Создайте на листе Qlik Sense визуализацию таблицы с измерениями **Year** и **Month**. Добавьте в таблицу следующие выражения в качестве мер:

- `Sum(Sales)`
- `Sum(Aggr( Rangesum(Above(Sum(Sales),0,12)), (Year, (Numeric, Ascending)), (Month, (Numeric, Ascending)) ))` с меткой Structured Aggr() в таблице.

Результат

Year	Month	Sum(Sales)	Structured Aggr()
2013	Jan	53495	53495
2013	Feb	48580	102075
2013	Mar	25651	127726
2013	Apr	36585	164311
2013	May	61211	225522
2013	Jun	23689	249211

Year	Month	Sum(Sales)	Structured Aggr()
2013	Jul	42311	291522
2013	Aug	41913	333435
2013	Sep	28886	362361
2013	Oct	25977	388298
2013	Nov	44455	432753
2013	Dec	64144	496897
2014	Jan	67775	67775

Объяснение

Этот пример демонстрирует агрегированные значения за 12-месячный период каждого года в хронологическом порядке по возрастанию, то есть фрагмент (Numeric, Ascending) структурированных параметров выражения **Aggr()**. В качестве структурных параметров требуются два специфических измерения: **Year** и **Month** с сортировкой (1) в хронологическом порядке по году **Year** (numeric) и (2) в хронологическом порядке по месяцу **Month** (numeric). Эти два измерения должны использоваться в визуализации таблиц или диаграмм. Это необходимо для того, чтобы список измерений функции **Aggr()** соответствовал измерениям объекта, используемого для визуализации.

Различия между этими мерами можно представить в таблице или отдельных линейных графиках:

- `Sum(Aggr( Rangesum(Above(Sum(Sales),0,12)), (Year), (Month) ))`
- `Sum(Aggr( Rangesum(Above(Sum(Sales),0,12)), (Year, (Numeric, Ascending)), (Month, (Numeric, Ascending)) ))`

Должно быть четко видно, что только последнее выражение выполняет требуемую аккумуляцию агрегированных значений.

См. также:

Базовые функции агрегирования (page 225)

5.4 Функции цвета

Эти функции можно использовать в выражениях, связанных с установкой и расчетом свойств цвета объектов диаграммы, а также в скриптах загрузки данных.

*Qlik Sense поддерживает функции цвета **Color()**, **qliktechblue** и **qliktechgray** для обеспечения обратной совместимости, но их использование не рекомендуется.*

ARGB

ARGB() используется в выражениях для установки или оценки свойств цвета объекта диаграммы, где цвет определяется красным **r**, зеленым **g** и синим **b** компонентами с коэффициентом alpha (прозрачность) **alpha**.

```
ARGB (alpha, r, g, b)
```

HSL

HSL() используется в выражениях для установки или оценки свойств цвета объекта диаграммы, где цвет определяется значениями **hue**, **saturation** и **luminosity** в диапазоне от 0 до 1.

```
HSL  (hue, saturation, luminosity)
```

RGB

RGB() возвращает целое число, соответствующее коду цвета, определенного тремя параметрами: красный канал r, зеленый канал g и синий канал b. Значения этих каналов должны быть целыми числами от 0 до 255. Эту функцию можно использовать в выражениях для установки или проверки свойств цвета объекта диаграммы.

```
RGB  (r, g, b)
```

Colormix1

Функция **Colormix1()** используется для возврата представления цвета ARGB от двухцветного градиента на основе значения от 0 до 1.

```
Colormix1 (Value , ColorZero , ColorOne)
```

Value – это действительное число от 0 до 1.

- Если Value = 0, возвращается значение ColorZero .
- Если Value = 1, возвращается значение ColorOne .
- Если 0 < Value< 1, возвращается соответствующий промежуточный оттенок.

ColorZero – это действительное представление цвета RGB для цвета, который будет связан с нижним пределом интервала.

ColorOne – это действительное представление цвета RGB для цвета, который будет связан с верхним пределом интервала.

Пример:

```
Colormix1(0.5, red(), blue())
```

возвращается:

```
ARGB(255,64,0,64) (purple)
```

Colormix2

Функция **Colormix2()** используется в выражениях для возврата представления цвета ARGB от двухцветного градиента на основе значения от -1 до 1 с возможностью указания промежуточного цвета для центральной позиции (0).

```
Colormix2 (Value ,ColorMinusOne , ColorOne[ , ColorZero])
```

Value – это действительное число от -1 до 1.

- Если Value = -1, возвращается первый цвет.
- Если Value = 1, возвращается второй цвет.
- Если -1 < Value< 1, возвращается соответствующий продукт смешивания цветов.

ColorMinusOne – это действительное представление цвета RGB для цвета, который будет связан с нижним пределом интервала.

ColorOne – это действительное представление цвета RGB для цвета, который будет связан с верхним пределом интервала.

ColorZero – это дополнительное действительное представление цвета RGB для цвета, который будет связан с центром интервала.

SysColor
SysColor() возвращает представление цвета ARGB для цвета системы Windows nr, где nr соответствует параметру для функции Windows API **GetSysColor(nr)**.

```
SysColor (nr)
```

ColorMapHue
ColorMapHue() возвращает значение цвета ARGB из карты цветов, которая изменяет компонент оттенка цветовой модели HSV. Цвета в карте цветов начинаются с красного, переходят в желтый, зеленый, голубой, синий, пурпурный и возвращаются к красному. Элемент x должен быть значением от 0 до 1.

```
ColorMapHue (x)
```

ColorMapJet
ColorMapJet() возвращает значение цвета ARGB из карты цветов, в которой цвета начинаются с синего, переходят в голубой, желтый, оранжевый и возвращаются к красному. Элемент x должен быть значением от 0 до 1.

```
ColorMapJet (x)
```

Предопределенные функции цвета

Следующие функции можно использовать в выражениях для предопределенных цветов. Каждая функция возвращает представление цвета RGB.

Дополнительно можно задать параметр для фактора alpha, в этом случае возвращается представление цвета ARGB. Значение фактора alpha 0 соответствует полной прозрачности, а значение фактора alpha 255 соответствует полной непрозрачности. Если значение для фактора alpha не задано, будет использовано значение 255.

Предопределенные функции цвета

Функция цвета	RGB value
black ([alpha])	(0,0,0)

blue([alpha])	(0,0,128)
brown([alpha])	(128,128,0)
cyan([alpha])	(0,128,128)
darkgray([alpha])	(128,128,128)
green([alpha])	(0,128,0)
lightblue([alpha])	(0,0,255)
lightcyan([alpha])	(0,255,255)
lightgray([alpha])	(192,192,192)
lightgreen([alpha])	(0,255,0)
lightmagenta([alpha])	(255,0,255)
lightred([alpha])	(255,0,0)
magenta([alpha])	(128,0,128)
red([alpha])	(128,0,0)
white([alpha])	(255,255,255)
yellow([alpha])	(255,255,0)

Примеры и результаты:

Примеры и результаты

Примеры	**Результаты**
`Blue()`	RGB(0,0,128)
`Blue(128)`	ARGB(128,0,0,128)

ARGB

ARGB() используется в выражениях для установки или оценки свойств цвета объекта диаграммы, где цвет определяется красным **r**, зеленым **g** и синим **b** компонентами с коэффициентом alpha (прозрачность) **alpha**.

Синтаксис:

```
ARGB(alpha, r, g, b)
```

Возвращаемые типы данных: двойное значение

Аргументы:

Аргументы

Аргумент	Описание
alpha	Значение прозрачности в диапазоне 0-255. 0 соответствует полной прозрачности, а 255 соответствует полной непрозрачности.
r, g, b	Значения красного, зеленого и синего компонентов. Цветовой компонент 0 соответствует отсутствию влияния, а компонент 255 соответствует полному влиянию.

Все аргументы должны быть выражениями, которые разрешаются в целые числа в диапазоне от 0 до 255.

При интерпретации и форматировании числового компонента в шестнадцатеричном формате значения цветовых компонентов легче увидеть. Например, номер светло-зеленого цвета 4 278 255 360, что в шестнадцатеричном представлении: FF00FF00. Первые две позиции «FF» (255) обозначают **альфа**-канал. Следующие две позиции «00» обозначают количество **red**, следующие две позиции «FF» обозначают количество **green** и последние две позиции «00» обозначают количество **blue**.

RGB

RGB() возвращает целое число, соответствующее коду цвета, определенного тремя параметрами: красный канал r, зеленый канал g и синий канал b. Значения этих каналов должны быть целыми числами от 0 до 255. Эту функцию можно использовать в выражениях для установки или проверки свойств цвета объекта диаграммы.

Синтаксис:

```
RGB (r, g, b)
```

Возвращаемые типы данных: двойное значение

Аргументы:

Аргументы

Аргумент	Описание
r, g, b	Значения красного, зеленого и синего компонентов. Цветовой компонент 0 соответствует отсутствию влияния, а компонент 255 соответствует полному влиянию.

Все аргументы должны быть выражениями, которые разрешаются в целые числа в диапазоне от 0 до 255.

При интерпретации и форматировании числового компонента в шестнадцатеричном формате значения цветовых компонентов легче увидеть. Например, номер светло-зеленого цвета 4 278 255 360, что в шестнадцатеричном представлении: FF00FF00. Первые две позиции «FF» (255) обозначают **альфа**-канал. В функциях **RGB** и **HSL** это всегда «FF» (непрозрачное). Следующие две позиции «00» обозначают количество **red**, следующие две позиции «FF» обозначают количество **green** и последние две позиции «00» обозначают количество **blue**.

Пример: Выражение диаграммы
Этот пример применяет пользовательский цвет к диаграмме:

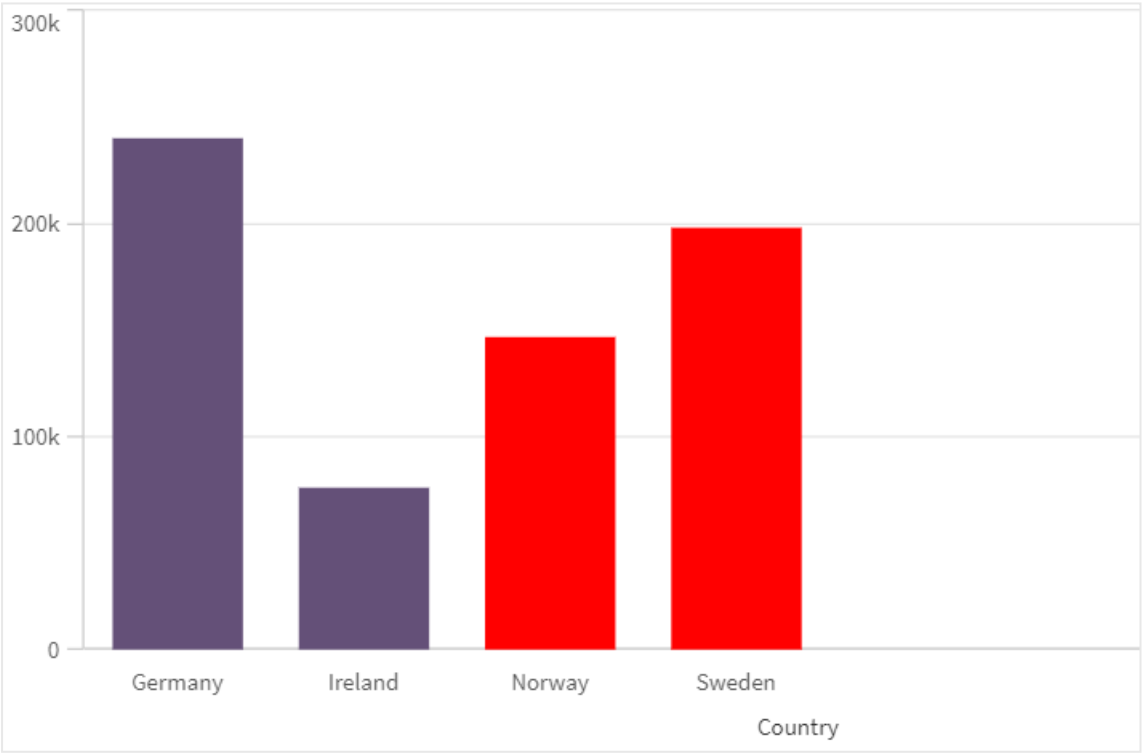
Данные, используемые в этом примере:

```
ProductSales: Load * Inline [Country,Sales,Budget Sweden,100000,50000 Germany, 125000, 175000 Norway, 74850, 68500 Ireland, 45000, 48000 Sweden,98000,50000 Germany, 115000, 175000 Norway, 71850, 68500 Ireland, 31000, 48000 ] (delimiter is ',');
```

Введите следующее выражение на панели свойств **Цвета и легенда**:

```
If (Sum(Sales)>Sum(Budget),RGB(255,0,0),RGB(100,80,120))
```

Результат:



Пример: Скрипт загрузки
Приведенный ниже пример демонстрирует значения RGB, эквивалентные значениям в шестнадцатеричном формате.

```
Load Text(R & G & B) as Text, RGB(R,G,B)      as Color; Load Num#(R,'(HEX)') as R, Num#(G,'(HEX)') as G, Num#(B,'(HEX)') as B Inline [R,G,B 01,02,03 AA,BB,CC];
```

Результат:

Текст	Цвет
010203	RGB(1,2,3)
AABBCC	RGB(170,187,204)

HSL

HSL() используется в выражениях для установки или оценки свойств цвета объекта диаграммы, где цвет определяется значениями **hue**, **saturation** и **luminosity** в диапазоне от 0 до 1.

Синтаксис:

```
HSL (hue, saturation, luminosity)
```

Возвращаемые типы данных: двойное значение

Аргументы:

Аргументы

Аргумент	Описание
hue, saturation, luminosity	Значения компонентов hue, saturation, и luminosity – от 0 до 1.

Все аргументы должны быть выражениями, которые разрешаются в целые числа в диапазоне от 0 до 1.

При интерпретации и форматировании числового компонента в шестнадцатеричном формате значения цветовых компонентов RGB легче увидеть. Например, номер светло-зеленого цвета 4 278 255 360, что в шестнадцатеричном представлении: FF00FF00 и RGB (0,255,0). Это аналогично HSL (80/240, 240/240, 120/240) – значению HSL (0.33, 1, 0.5).

5.5 Условные функции

Все условные функции вычисляют условие и затем возвращают различные ответы в зависимости от значения условия. Функции можно использовать как в скрипте загрузки данных, так и в выражениях диаграмм.

Обзор условных функций

Каждая функция подробно описана после обзора. Также можно щелкнуть имя функции в синтаксисе, чтобы получить немедленный доступ к подробной информации об этой конкретной функции.

alt

Функция **alt** возвращает первый из параметров, имеющий допустимое числовое представление. Если такое совпадение не было найдено, будет возвращен последний параметр. Может использоваться любое количество параметров.

```
alt (expr1[ , expr2 , expr3 , ...] , else)
```

class

Функция **class** назначает первый параметр интервалу классов. Результат – двойное значение с уравнением a<=x<b в качестве текстового значения, где a и b являются верхней и нижней границами диапазона, а нижняя граница является числовым значением.

```
class (expression, interval [ , label [ , offset ]])
```

coalesce

Функция **coalesce** возвращает первый из параметров, имеющий допустимое представление non-NULL. Может использоваться любое количество параметров.

```
coalesce(expr1[ , expr2 , expr3 , ...])
```

if

Функция **if** возвращает значение в зависимости от условия функции: True или False.

```
if (condition , then , else)
```

match

Функция **match** сравнивает первый параметр со всеми последующими и возвращает числовое значение местоположения для совпадающих выражений. При сравнении учитывается регистр.

```
match ( str, expr1 [ , expr2,...exprN ])
```

mixmatch

Функция **mixmatch** сравнивает первый параметр со всеми последующими и возвращает числовое значение местоположения для совпадающих выражений. При сравнении регистр не учитывается.

```
mixmatch ( str, expr1 [ , expr2,...exprN ])
```

pick

Функция отбора возвращает выражение *n* в списке.

```
pick (n, expr1[ , expr2,...exprN])
```

wildmatch

Функция **wildmatch** сравнивает первый параметр со всеми последующими и возвращает число совпадающих выражений. Допускается использование знаков подстановки (***** и **?**) в строках сравнения. ***** соответствует любой последовательности символов. **?** соответствует любому символу. При сравнении регистр не учитывается.

```
wildmatch ( str, expr1 [ , expr2,...exprN ])
```

alt

Функция **alt** возвращает первый из параметров, имеющий допустимое числовое представление. Если такое совпадение не было найдено, будет возвращен последний параметр. Может использоваться любое количество параметров.

Синтаксис:

```
alt(expr1[ , expr2 , expr3 , ...] , else)
```

Аргументы:

Аргументы

Аргумент	Описание
expr1	Первое выражение для проверки допустимого числового представления.
expr2	Второе выражение для проверки допустимого числового представления.
expr3	Третье выражение для проверки допустимого числового представления.
else	Значение, возвращаемое, если ни один из предыдущих параметров не имеет допустимого числового представления.

Функция alt часто используется с функциями интерпретации чисел или дат. Таким образом, программа Qlik Sense может тестировать различные форматы дат в приоритизированном порядке. Эта функция также может использоваться для обработки значений NULL в числовых выражениях.

Примеры:

Примеры

Пример	Результат

`alt( date#( dat , 'YYYY/MM/DD' ),`
 	
`date#( dat , 'MM/DD/YYYY' ),`
 	
`date#( dat , 'MM/DD/YY' ),`
 	
`'No valid date' )`
	Это выражение протестирует наличие даты в поле даты в соответствии с любым из трех указанных форматов. Если дата соответствует формату, будет возвращено двойное значение, содержащее исходную строку и допустимое числовое представление даты. Если совпадение не найдено, будет возвращен текст 'No valid date' (без допустимого числового представления).

`alt(Sales,0) + alt(Margin,0)`	Это выражение добавляет поля Sales и Margin, заменяя отсутствующее значение (NULL) на 0.

class

Функция **class** назначает первый параметр интервалу классов. Результат – двойное значение с уравнением a<=x<b в качестве текстового значения, где a и b являются верхней и нижней границами диапазона, а нижняя граница является числовым значением.

Синтаксис:

```
class(expression, interval [ , label [ , offset ]])
```

Аргументы:

Аргументы

Аргумент	Описание
interval	Число, которое указывает ширину диапазона.
label	Произвольная строка, которая может заменять 'x' в результирующем тексте.
offset	Число, которое может использоваться как смещение от начальной точки по умолчанию для классификации. Начальная точка по умолчанию обычно равна 0.

Примеры:

Примеры

Пример	Результат
`class( var,10 )` с `var = 23`	возвращает'20<=x<30'
`class( var,5,'value' )` с `var = 23`	возвращает'20<= value <25'
`class( var,10,'x',5 )` с `var = 23`	возвращает '15<=x<25'

Пример: скрипт загрузки с использованием class

Пример: скрипт загрузки

Скрипт загрузки

В этом примере мы загружаем таблицу, содержащую имя и возраст людей. Мы хотим добавить поле, которое классифицирует каждого человека по возрастной группе с десятилетним интервалом. Первоначальная исходная таблица выглядит, как показано ниже.

Результаты

Name	Age
John	25

Name	Age
Karen	42
Yoshi	53

Чтобы добавить поле классификации по возрастной группе, можно добавить оператор предшествующей загрузки с помощью функции **class**.

Создайте новую вкладку в редакторе загрузки данных и загрузите следующие данные через встроенную загрузку. Создайте приведенную ниже таблицу ниже в Qlik Sense, чтобы увидеть результаты.

```
 LOAD *, class(Age, 10, 'age') As Agegroup;  LOAD * INLINE [ Age, Name 25, John 42, Karen 53, Yoshi];
```

Результаты

Результаты

Name	Age	Agegroup
John	25	20 <= age < 30
Karen	42	40 <= age < 50
Yoshi	53	50 <= age < 60

coalesce

Функция **coalesce** возвращает первый из параметров, имеющий допустимое представление non-NULL. Может использоваться любое количество параметров.

Синтаксис:

```
coalesce(expr1[ , expr2 , expr3 , ...])
```

Аргументы:

Аргументы

Аргумент	Описание
expr1	Первое выражение для проверки допустимого ненулевого представления.
expr2	Второе выражение для проверки допустимого ненулевого представления.
expr3	Третье выражение для проверки допустимого ненулевого представления.

Примеры:

Примеры

Пример	Результат
	Это выражение изменяет все нулевые значения поля на 'Н/Д'.
`Coalesce(ProductDescription, ProductName, ProductCode, 'no description available')`	Это выражение выберет между тремя различными полями описания продукта, когда в некоторых полях может не быть значений для продукта. Первое из полей в указанном порядке с ненулевым значением будет возвращено. Если ни одно из полей не будет содержать значения, результат будет «нет описания».
`Coalesce(TextBetween(FileName, '"', '"'), FileName)`	Это выражение обрежет возможные кавычки включения в поле *FileName*. Если в данном *FileName* есть кавычки, они будут удалены, а вложенное *FileName* без кавычек будет возвращено. Если функция *TextBetween* не находит разделители, она возвращает нуль, который отклоняется функцией **Coalesce**, возвращающей вместо этого необработанное *FileName*.

if

Функция **if** возвращает значение в зависимости от условия функции: True или False.

Синтаксис:

```
if(condition , then [, else])
```

Функция if имеет три параметра, *condition*, *then* и *else*, все из которых являются выражениями. Два остальных, *then* и *else*, могут относиться к любому типу.

Аргументы

Аргумент	Описание
condition	Выражение, которое интерпретируется логическим образом.
then	Выражение, которое может быть любого типа. Если элемент *condition* равен True, функция if возвращает значение выражения *then*.
else	Выражение, которое может быть любого типа. Если элемент *condition* равен False, функция if возвращает значение выражения *else*. Этот параметр дополнительный. Если *condition* равно False, возвращается NULL, если не указан else.

Пример

Пример	Результат
`if( Amount>= 0, 'OK', 'Alarm' )`	Это выражение проверяет, является ли количество положительным числом (0 или больше) и возвращает значение 'OK', если это так. Если количество меньше 0, будет возвращено значение 'Alarm'.

Пример: скрипт загрузки с использованием if

Пример: Скрипт загрузки

Скрипт загрузки

If можно использовать в скрипте загрузки наряду с другими способами и объектами, в том числе переменными. К примеру, если указана переменная *threshold* и вы хотите включить в модель данных поле, основанное на этом пороговом значении, выполните следующие действия.

Создайте новую вкладку в редакторе загрузки данных и загрузите следующие данные через встроенную загрузку. Создайте приведенную ниже таблицу ниже в Qlik Sense, чтобы увидеть результаты.

```
Transactions: Load * Inline [ transaction_id, transaction_date, transaction_amount,
transaction_quantity, customer_id, size, color_code 3750, 20180830, 23.56, 2, 2038593, L, Red
3751, 20180907, 556.31, 6, 203521, m, orange 3752, 20180916, 5.75, 1, 5646471, S, blue 3753,
20180922, 125.00, 7, 3036491, l, Black 3754, 20180922, 484.21, 13, 049681, xs, Red 3756,
20180922, 59.18, 2, 2038593, M, Blue 3757, 20180923, 177.42, 21, 203521, XL, Black ];   set
threshold = 100;                                        /* Create new table called Transaction_Buckets  Comp
amount field from Transaction table to threshold of 100. Output results into a new field
called Compared to Threshold                                                              */
If(transaction_amount > $(threshold),'Greater than $(threshold)','Less than $(threshold)') as
[Compared to Threshold] Resident Transactions;
```

Результаты

В таблице Qlik Sense представлены результаты использования функции *if* в скрипте загрузки.

transaction_id	По сравнению с пороговым значением
3750	Меньше 100
3751	Больше 100
3752	Меньше 100
3753	Больше 100
3754	Больше 100
3756	Меньше 100
3757	Больше 100

Примеры: выражения диаграммы с использованием if

Примеры: Выражения диаграммы

Выражение диаграммы 1

Скрипт загрузки

Создайте новую вкладку в редакторе загрузки данных и загрузите следующие данные через встроенную загрузку. После загрузки данных создайте приведенные ниже примеры выражения диаграммы ниже в таблице Qlik Sense.

```
MyTable: LOAD * inline [Date, Location, Incidents 1/3/2016, Beijing, 0 1/3/2016, Boston, 12
1/3/2016, Stockholm, 3 1/3/2016, Toronto, 0 1/4/2016, Beijing, 0 1/4/2016, Boston, 8];
```

В таблице Qlik Sense представлены примеры использования функции *if* в выражении диаграммы.

Дата	Местоположение	Инциденты	if(Incidents>=10, 'Critical', 'Ok')	if(Incidents>=10, 'Critical', If(Incidents>=1 and Incidents<10, 'Warning', 'Ok'))
1/3/2016	Пекин	0	OK	OK
1/3/2016	Бостон	12	Критический	Критический
1/3/2016	Стокгольм	3	OK	Предупреждение
1/3/2016	Торонто	0	OK	OK
1/4/2016	Пекин	0	OK	OK
1/4/2016	Бостон	8	OK	Предупреждение

Выражение диаграммы 2

В новом приложении добавьте скрипт на новую вкладку редактора загрузки данных, а затем загрузите данные. Затем можно создать таблицу с приведенными ниже выражениями диаграммы.

```
SET FirstWeekDay=0; Load  Date(MakeDate(2022)+RecNo()-1) as Date Autogenerate 14;
```

В таблице Qlik Sense представлен пример использования функции *if* в выражении диаграммы.

Дата	WeekDay(Date)	If(WeekDay (Date)>=5,'WeekEnd','Normal Day')
1/1/2022	Sat	Выходной день
1/2/2022	Вск	WeekEnd
1/3/2022	Пон	Normal Day

Дата	WeekDay(Date)	If(WeekDay (Date)>=5,'WeekEnd','Normal Day')
1/4/2022	Втр	Normal Day
1/5/2022	Ср	Normal Day
1/6/2022	Thu	Normal Day
1/7/2022	Fri	Normal Day
1/8/2022	Sat	WeekEnd
1/9/2022	Вск	WeekEnd
1/10/2022	Пон	Normal Day
1/11/2022	Втр	Normal Day
1/12/2022	Ср	Normal Day
1/13/2022	Thu	Normal Day
1/14/2022	Fri	Normal Day

match

Функция **match** сравнивает первый параметр со всеми последующими и возвращает числовое значение местоположения для совпадающих выражений. При сравнении учитывается регистр.

Синтаксис:

```
match( str, expr1 [ , expr2,...exprN ])
```

*Если необходимо использовать сравнение, в котором регистр не учитывается, используйте функцию **mixmatch**. Если необходимо использовать сравнение, в котором регистр не учитывается, и знаки подстановки, используйте функцию **wildmatch**.*

Пример: Скрипт загрузки с использованием match

Пример: Скрипт загрузки

Скрипт загрузки

Функцию match можно использовать для загрузки подмножества данных. К примеру, можно вернуть числовое значение выражения в функции. Затем можно ограничить загруженные данные с использованием числового значения. При отсутствии совпадений функция Match возвращает 0. Таким образом, в данном примере все выражения, для которых не были найдены совпадения, вернут 0 и будут исключены из загрузки данных при помощи оператора WHERE.

Создайте новую вкладку в редакторе загрузки данных и загрузите следующие данные через встроенную загрузку. Создайте приведенную ниже таблицу ниже в Qlik Sense, чтобы увидеть результаты.

```
Transactions: Load * Inline [ transaction_id, transaction_date, transaction_amount, transaction_quantity, customer_id, size, color_code 3750, 20180830, 23.56, 2, 2038593, L, Red 3751, 20180907, 556.31, 6, 203521, m, orange 3752, 20180916, 5.75, 1, 5646471, S, blue 3753, 20180922, 125.00, 7, 3036491, l, Black 3754, 20180922, 484.21, 13, 049681, xs, Red 3756, 20180922, 59.18, 2, 2038593, M, Blue 3757, 20180923, 177.42, 21, 203521, XL, Black ];   /* Create new table called Transaction_Buckets Create new fields called Customer, and Color code - Blue and Black Load Transactions table. Match returns 1 for 'Blue', 2 for 'Black'.  Does not return a value for 'blue' because match is case sensitive. Only values that returned numeric value greater than 0  are loaded by WHERE statment into Transactions_Buckets table. */ Transaction_Buckets: Load  customer_id,  customer_id as [Customer],   color_code as [Color Code Blue and Black] Resident Transactions Where match(color_code,'Blue','Black') > 0;
```

Результаты

В таблице Qlik Sense представлены результаты использования функции match в скрипте загрузки

Color Code Blue and Black	Customer
Black	203521
Black	3036491
Blue	2038593

Примеры: выражения диаграммы с использованием match

Примеры: Выражения диаграммы

Выражение диаграммы 1

Скрипт загрузки

Создайте новую вкладку в редакторе загрузки данных и загрузите следующие данные через встроенную загрузку. После загрузки данных создайте приведенные ниже примеры выражения диаграммы ниже в таблице Qlik Sense.

```
MyTable: Load * inline [Cities, Count Toronto, 123 Toronto, 234 Toronto, 231 Boston, 32 Boston, 23 Boston, 1341 Beijing, 234 Beijing, 45 Beijing, 235 Stockholm, 938 Stockholm, 39 Stockholm, 189 zurich, 2342 zurich, 9033 zurich, 0039];
```

Первое выражение в таблице ниже возвращает 0 для значения «Stockholm», так как «Stockholm» не входит в список выражений в функции **match**. Выражение также возвращает 0 для значения «Zurich», так как сравнение **match** учитывает регистр.

В таблице Qlik Sense представлены примеры использования функции *match* в выражении диаграммы

Cities	match(Cities,'Toronto','Boston','Beijing','Zurich')	match(Cities,'Toronto','Boston','Beijing','Stockholm','zurich')
Beijing	3	3
Boston	2	2
Stockholm	0	4
Toronto	1	1
zurich	0	5

Выражение диаграммы 2

Функцию match можно применять для пользовательской сортировки выражений.

По умолчанию сортировка столбцов выполняется в числовом или алфавитном порядке в зависимости от характера данных.

Таблица Qlik Sense с примером порядка сортировки по умолчанию

Cities
Beijing
Boston
Stockholm
Toronto
zurich

Чтобы изменить порядок сортировки, выполните следующие действия:

1. Откройте раздел **Сортировка** диаграммы на панели **Свойства**.
2. Выключите автоматическую сортировку столбца, для которого необходимо применить пользовательскую сортировку.
3. Отмените выбор параметров **Сортировка по численным значениям** и **Сортировка по алфавиту**.
4. Выберите **Сортировка по выражению** и введите выражение следующего вида:
   `=match( Cities, 'Toronto','Boston','Beijing','Stockholm','zurich')`
   Порядок сортировки столбца Cities будет изменен.

Таблица Qlik Sense с примером изменения порядка сортировки при помощи функции *match*

Cities
Toronto

Cities
Boston
Beijing
Stockholm
zurich

Также можно просмотреть возвращенное числовое значение.

Таблица Qlik Sense с примерами числовых значений, возвращенных из функции *match*

Cities	Cities & ' - ' & match (Cities, 'Toronto','Boston', 'Beijing','Stockholm','zurich')
Toronto	Toronto - 1
Boston	Boston - 2
Beijing	Beijing - 3
Stockholm	Stockholm - 4
zurich	zurich - 5

mixmatch

Функция **mixmatch** сравнивает первый параметр со всеми последующими и возвращает числовое значение местоположения для совпадающих выражений. При сравнении регистр не учитывается.

Синтаксис:

```
mixmatch( str, expr1 [ , expr2,...exprN ])
```

Если вместо этого необходимо использовать сравнение, в котором регистр учитывается, используйте функцию **match**. Если необходимо использовать сравнение, в котором регистр не учитывается, и знаки подстановки, используйте функцию **wildmatch**.

Пример: скрипт загрузки с использованием mixmatch

Пример: Скрипт загрузки

Скрипт загрузки

Функцию mixmatch можно использовать для загрузки подмножества данных. К примеру, можно вернуть числовое значение выражения в функции. Затем можно ограничить загруженные данные с использованием числового значения. При отсутствии совпадений функция Mixmatch возвращает 0. Таким образом, в данном примере все выражения, для которых не были найдены совпадения, вернут 0 и будут исключены из загрузки данных при помощи оператора WHERE.

Создайте новую вкладку в редакторе загрузки данных и загрузите следующие данные через встроенную загрузку. Создайте приведенную ниже таблицу ниже в Qlik Sense, чтобы увидеть результаты.

```
Load * Inline [ transaction_id, transaction_date, transaction_amount, transaction_quantity,
customer_id, size, color_code 3750, 20180830, 23.56, 2, 2038593, L, Red 3751, 20180907,
556.31, 6, 203521, m, orange 3752, 20180916, 5.75, 1, 5646471, S, blue 3753, 20180922, 125.00,
7, 3036491, l, Black 3754, 20180922, 484.21, 13, 049681, xs, Red 3756, 20180922, 59.18, 2,
2038593, M, Blue 3757, 20180923, 177.42, 21, 203521, XL, Black ];  /* Create new table called
Transaction_Buckets Create new fields called Customer, and Color code - Black, Blue, blue Load
Transactions table. Mixmatch returns 1 for 'Black', 2 for 'Blue'.  Also returns 3 for 'blue'
because mixmatch is not case sensitive. Only values that returned numeric value greater than 0
 are loaded by WHERE statement into Transactions_Buckets table. */  Transaction_Buckets: Load
  customer_id,     customer_id as [Customer],    color_code as [Color Code - Black, Blue,
blue] Resident Transactions Where mixmatch(color_code,'Black','Blue') > 0;
```

Результаты

В таблице Qlik Sense представлены результаты использования функции mixmatch в скрипте загрузки.

Color Code Black, Blue, blue	Customer
Black	203521
Black	3036491
Blue	2038593
blue	5646471

Примеры: выражения диаграммы с использованием mixmatch

Примеры: Выражения диаграммы

Создайте новую вкладку в редакторе загрузки данных и загрузите следующие данные через встроенную загрузку. После загрузки данных создайте приведенные ниже примеры выражения диаграммы ниже в таблице Qlik Sense.

Выражение диаграммы 1

```
MyTable: Load * inline [Cities, Count Toronto, 123 Toronto, 234 Toronto, 231 Boston, 32
Boston, 23 Boston, 1341 Beijing, 234 Beijing, 45 Beijing, 235 Stockholm, 938 Stockholm, 39
Stockholm, 189 zurich, 2342 zurich, 9033 zurich, 0039];
```

Первое выражение в таблице ниже возвращает 0 для значения «Stockholm», так как «Stockholm» не входит в список выражений в функции **mixmatch**. Выражение возвращает 4 для значения «Zurich», так как сравнение **mixmatch** не учитывает регистр.

В таблице Qlik Sense представлены примеры использования функции *mixmatch* в выражении диаграммы

Cities	mixmatch(Cities,'Toronto','Boston','Beijing','Zurich')	mixmatch(Cities,'Toronto','Boston','Beijing','Stockholm','Zurich')
Beijing	3	3
Boston	2	2
Stockholm	0	4
Toronto	1	1
zurich	4	5

Выражение диаграммы 2

Функцию mixmatch можно применять для пользовательской сортировки выражений.

По умолчанию сортировка столбцов выполняется в алфавитном или числовом порядке в зависимости от характера данных.

Таблица Qlik Sense с примером порядка сортировки по умолчанию

Cities
Beijing
Boston
Stockholm
Toronto
zurich

Чтобы изменить порядок сортировки, выполните следующие действия:

1. Откройте раздел **Сортировка** диаграммы на панели **Свойства**.
2. Выключите автоматическую сортировку столбца, для которого необходимо применить пользовательскую сортировку.
3. Отмените выбор параметров **Сортировка по численным значениям** и **Сортировка по алфавиту**.
4. Выберите **Сортировка по выражению** и введите следующее выражение:
   `=mixmatch( Cities, 'Toronto','Boston','Beijing','Stockholm','Zurich')`
   Порядок сортировки столбца Cities будет изменен.

Таблица Qlik Sense с примером изменения порядка сортировки при помощи функции *mixmatch*.

Cities
Toronto

Cities
Boston
Beijing
Stockholm
zurich

Также можно просмотреть возвращенное числовое значение.

Таблица Qlik Sense с примерами числовых значений, возвращенных из функции *mixmatch*.

Cities	Cities & ' - ' & mixmatch (Cities, 'Toronto','Boston', 'Beijing','Stockholm','Zurich')
Toronto	Toronto - 1
Boston	Boston - 2
Beijing	Beijing - 3
Stockholm	Stockholm - 4
zurich	zurich - 5

pick

Функция отбора возвращает выражение *n* в списке.

Синтаксис:

```
pick(n, expr1[ , expr2,...exprN])
```

Аргументы:

Аргументы

Аргумент	Описание
n	*n* представляет собой целое число от 1 до N.

Пример:

Пример

Пример	Результат
`pick( N, 'A','B',4, 6 )`	возвращает 'B', если N = 2 возвращает 4, если N = 3

wildmatch

Функция **wildmatch** сравнивает первый параметр со всеми последующими и возвращает число совпадающих выражений. Допускается использование знаков подстановки (* и ?) в строках сравнения. * соответствует любой последовательности символов. ? соответствует любому символу. При сравнении регистр не учитывается.

Синтаксис:

```
wildmatch( str, expr1 [ , expr2,...exprN ])
```

Если необходимо использовать сравнение без подстановочных знаков, используйте функции **match** или **mixmatch**.

Пример: Скрипт загрузки с использованием wildmatch

Пример: Скрипт загрузки

Скрипт загрузки

Функцию wildmatch можно использовать для загрузки подмножества данных. К примеру, можно вернуть числовое значение выражения в функции. Затем можно ограничить загруженные данные с использованием числового значения. При отсутствии совпадений функция Wildmatch возвращает 0. Таким образом, в данном примере все выражения, для которых не были найдены совпадения, вернут 0 и будут исключены из загрузки данных при помощи оператора WHERE.

Создайте новую вкладку в редакторе загрузки данных и загрузите следующие данные через встроенную загрузку. Создайте приведенную ниже таблицу ниже в Qlik Sense, чтобы увидеть результаты.

```
Transactions: Load * Inline [ transaction_id, transaction_date, transaction_amount, transaction_quantity, customer_id, size, color_code 3750, 20180830, 23.56, 2, 2038593, L, Red 3751, 20180907, 556.31, 6, 203521, m, orange 3752, 20180916, 5.75, 1, 5646471, S, blue 3753, 20180922, 125.00, 7, 3036491, l, Black 3754, 20180922, 484.21, 13, 049681, xs, Red 3756, 20180922, 59.18, 2, 2038593, M, Blue 3757, 20180923, 177.42, 21, 203521, XL, Black ];   /* Create new table called Transaction_Buckets Create new fields called Customer, and Color code - Black, Blue, blue, red Load Transactions table. Wildmatch returns 1 for 'Black', 'Blue', and 'blue', and 2 for 'Red'.  Only values that returned numeric value greater than 0  are loaded by WHERE statement into Transactions_Buckets table. */   Transaction_Buckets: Load customer_id,    customer_id as [Customer],    color_code as [Color Code Black, Blue, blue, Red] Resident Transactions Where wildmatch(color_code,'Bl*','R??') > 0;
```

Результаты

В таблице Qlik Sense представлены результаты использования функции *wildmatch* в скрипте загрузки

Color Code Black, Blue, blue, Red	Customer
Black	203521

Color Code Black, Blue, blue, Red	Customer
Black	3036491
Blue	2038593
blue	5646471
Red	049681
Red	2038593

Примеры: Выражения диаграммы с использованием wildmatch

Пример: Выражение диаграммы

Выражение диаграммы 1

Создайте новую вкладку в редакторе загрузки данных и загрузите следующие данные через встроенную загрузку. После загрузки данных создайте приведенные ниже примеры выражения диаграммы ниже в таблице Qlik Sense.

```
MyTable: Load * inline [Cities, Count Toronto, 123 Toronto, 234 Toronto, 231 Boston, 32
Boston, 23 Boston, 1341 Beijing, 234 Beijing, 45 Beijing, 235 Stockholm, 938 Stockholm, 39
Stockholm, 189 zurich, 2342 zurich, 9033 zurich, 0039];
```

Первое выражение в таблице ниже возвращает 0 для значения «Stockholm», так как «Stockholm» не входит в список выражений в функции **wildmatch**. Выражение также возвращает 0 для значения «Boston», так как для ? совпадением является одиночный символ.

В таблице Qlik Sense представлены примеры использования функции *wildmatch* в выражении диаграммы

Cities	wildmatch(Cities,'Tor*','?ton','Beijing','*urich')	wildmatch(Cities,'Tor*','???ton','Beijing','Stockholm','*urich')
Beijing	3	3
Boston	0	2
Stockholm	0	4
Toronto	1	1
zurich	4	5

Выражение диаграммы 2

Функцию wildmatch можно применять для пользовательской сортировки выражений.

По умолчанию сортировка столбцов выполняется в числовом или алфавитном порядке в зависимости от характера данных.

Таблица Qlik Sense с примером порядка сортировки по умолчанию

Cities
Beijing
Boston
Stockholm
Toronto
zurich

Чтобы изменить порядок сортировки, выполните следующие действия:

1. Откройте раздел **Сортировка** диаграммы на панели **Свойства**.
2. Выключите автоматическую сортировку столбца, для которого необходимо применить пользовательскую сортировку.
3. Отмените выбор параметров **Сортировка по численным значениям** и **Сортировка по алфавиту**.
4. Выберите **Сортировка по выражению** и введите выражение следующего вида:
   `=wildmatch( Cities, 'Tor*','???ton','Beijing','Stockholm','*urich')`
   Порядок сортировки столбца Cities будет изменен.

Таблица Qlik Sense с примером изменения порядка сортировки при помощи функции *wildmatch*.

Cities
Toronto
Boston
Beijing
Stockholm
zurich

Также можно просмотреть возвращенное числовое значение.

Таблица Qlik Sense с примерами числовых значений, возвращенных из функции *wildmatch*

Cities	Cities & ' - ' & wildmatch (Cities, 'Tor*','???ton','Beijing','Stockholm','*urich')
Toronto	Toronto - 1
Boston	Boston - 2
Beijing	Beijing - 3
Stockholm	Stockholm - 4
zurich	zurich - 5

5.6 Функции счетчика

В этом разделе описаны функции, которые относятся к счетчикам записей во время оценки оператора **LOAD** в скрипте загрузки данных. Единственная функция, которая может использоваться в выражениях диаграммы – это **RowNo()**.

Некоторые функции счетчика не имеют никаких параметров, но завершающие скобки тем не менее требуются.

Обзор функций счетчика

Каждая функция подробно описана после обзора. Также можно щелкнуть имя функции в синтаксисе, чтобы получить немедленный доступ к подробной информации об этой конкретной функции.

autonumber

Эта функция скрипта возвращает уникальное значение целого для каждого определенного оцененного значения *expression*, возникающего в процессе выполнения скрипта. Эта функция может использоваться, например, при создании компактного представления сложного ключа в памяти.

```
autonumber (expression[ , AutoID])
```

autonumberhash128

Эта функция скрипта вычисляет 128-битные случайные данные значений выражений комбинированного ввода и возвращает уникальное значение целого для каждого определенного значения случайных данных, возникающего в процессе выполнения скрипта. Эта функция может использоваться, например, при создании компактного представления сложного ключа в памяти.

```
autonumberhash128 (expression {, expression})
```

autonumberhash256

Эта функция скрипта вычисляет 256-битные случайные данные значений выражений комбинированного ввода и возвращает уникальное значение целого для каждого определенного значения случайных данных, возникающего в процессе выполнения скрипта. Эта функция может использоваться, например, при создании компактного представления сложного ключа в памяти.

```
autonumberhash256 (expression {, expression})
```

IterNo

Эта функция скрипта возвращает целое, указывающее на то, в который раз оценивается одна запись в операторе **LOAD** предложением **while**. Первый шаг цикла – число 1. Функция **IterNo** имеет значение только при условии совместного использования с предложением **while**.

```
IterNo ( )
```

RecNo

Эта функция скрипта возвращает целое число читаемой в текущий момент строки текущей таблицы. Первая запись – число 1.

```
RecNo ( )
```

RowNo - script function

Эта функция возвращает целое значение позиции текущей строки в итоговой внутренней таблице Qlik Sense. Первая строка имеет номер 1.

```
RowNo ( )
```

RowNo - chart function

Функция **RowNo()** возвращает текущие строки в текущий сегмент столбца в таблице. Для растровых диаграмм функция **RowNo()** возвращает текущие строки в эквивалент прямой таблицы диаграммы.

```
RowNo — функция диаграммы([TOTAL])
```

autonumber

Эта функция скрипта возвращает уникальное значение целого для каждого определенного оцененного значения *expression*, возникающего в процессе выполнения скрипта. Эта функция может использоваться, например, при создании компактного представления сложного ключа в памяти.

> *Можно подключить только ключи **autonumber**, созданные в той же загрузке данных, поскольку целое число создается согласно порядку чтения таблицы. При использовании ключей, хранящихся между загрузками данных, независимо от сортировки исходных данных, необходимо использовать функции **hash128**, **hash160** или **hash256**.*

Синтаксис:

```
autonumber(expression[ , AutoID])
```

Аргументы:

Аргумент	Описание
AutoID	Чтобы создать несколько экземпляров счетчиков при использовании функции **autonumber** на различных ключах в скрипте, для названия каждого счетчика может использоваться дополнительный параметр *AutoID*.

Пример: Создание составного ключа

В данном примере мы создаем составной ключ, используя функцию **autonumber** для преобразования памяти. Этот пример представлен в целях демонстрации, поэтому в данном случае информация краткая, но при использовании таблицы, содержащей большое количество строк, информация будет более содержательной.

Данные, используемые в примере

Region	Year	Month	Sales
North	2014	May	245
North	2014	May	347
North	2014	June	127
South	2014	June	645
South	2013	May	367
South	2013	May	221

Исходные данные загружаются с помощью встроенных данных. Затем мы добавляем предшествующую загрузку, которая создает составной ключ из полей Region, Year и Month.

```
RegionSales:
LOAD *,
AutoNumber(Region&Year&Month) as RYMkey;

LOAD * INLINE
[ Region, Year, Month, Sales
North, 2014,   May,    245
North, 2014,   May,    347
North, 2014,   June,   127
South, 2014,   June,   645
South, 2013,   May, 367
South, 2013,   May,    221
];
```

Полученная таблица выглядит следующим образом:

Результирующая таблица

Region	Year	Month	Sales	RYMkey
North	2014	May	245	1
North	2014	May	347	1
North	2014	June	127	2
South	2014	June	645	3
South	2013	May	367	4
South	2013	May	221	4

В этом примере вы можете обратиться к RYMkey, например 1 вместо строки 'North2014May', если необходимо установить связь с другой таблицей.

Теперь мы загружаем исходную таблицу с ценами похожим образом. Поля Region, Year и Month исключены предыдущей загрузкой во избежание создания синтетического ключа, мы уже создаем составной ключ с функцией **autonumber**, связывая таблицы.

```
RegionCosts:
LOAD Costs,
AutoNumber(Region&Year&Month) as RYMkey;

LOAD * INLINE
[ Region, Year, Month, Costs
South, 2013,   May,    167
North, 2014,   May,    56
North, 2014,   June,   199
South, 2014,   June,   64
South, 2013,   May, 172
South, 2013,   May,    126
];
```

Теперь мы можем добавить визуализацию таблицы на лист и добавить поля Region, Year и Month, а также меры Sum для продаж и стоимости. Таблица будет выглядеть так:

Результирующая таблица

Region	Year	Month	Sum([Sales])	Sum([Costs])
Totals	-	-	1952	784
North	2014	June	127	199
North	2014	May	592	56
South	2014	June	645	64
South	2013	May	588	465

autonumberhash128

Эта функция скрипта вычисляет 128-битные случайные данные значений выражений комбинированного ввода и возвращает уникальное значение целого для каждого определенного значения случайных данных, возникающего в процессе выполнения скрипта. Эта функция может использоваться, например, при создании компактного представления сложного ключа в памяти.

*Можно подключить только ключи **autonumberhash128**, созданные в той же загрузке данных, поскольку целое число создается согласно порядку чтения таблицы. При использовании ключей, хранящихся между загрузками данных, независимо от сортировки исходных данных, необходимо использовать функции **hash128**, **hash160** или **hash256**.*

Синтаксис:

```
autonumberhash128(expression {, expression})
```

Пример: Создание составного ключа

В данном примере мы создаем составной ключ, используя функцию **autonumberhash128** для преобразования памяти. Этот пример представлен в целях демонстрации, поэтому в данном случае информация краткая, но при использовании таблицы, содержащей большое количество строк, информация будет более содержательной.

Данные, используемые в примере

Region	Year	Month	Sales
North	2014	May	245
North	2014	May	347
North	2014	June	127
South	2014	June	645
South	2013	May	367
South	2013	May	221

Исходные данные загружаются с помощью встроенных данных. Затем мы добавляем предшествующую загрузку, которая создает составной ключ из полей Region, Year и Month.

```
RegionSales:
LOAD *,
AutoNumberHash128(Region, Year, Month) as RYMkey;

LOAD * INLINE
[ Region, Year, Month, Sales
North, 2014,   May,    245
North, 2014,   May,    347
North, 2014,   June,   127
South, 2014,   June,   645
South, 2013,   May, 367
South, 2013,   May,    221
];
```

Полученная таблица выглядит следующим образом:

Результирующая таблица

Region	Year	Month	Sales	RYMkey
North	2014	May	245	1
North	2014	May	347	1
North	2014	June	127	2
South	2014	June	645	3
South	2013	May	367	4
South	2013	May	221	4

В этом примере вы можете обратиться к RYMkey, например 1 вместо строки 'North2014May', если необходимо установить связь с другой таблицей.

Теперь мы загружаем исходную таблицу с ценами похожим образом. Поля Region, Year и Month исключены предыдущей загрузкой во избежание создания синтетического ключа, мы уже создаем составной ключ с функцией **autonumberhash128**, связывая таблицы.

```
RegionCosts:
LOAD Costs,
AutoNumberHash128(Region, Year, Month) as RYMkey;

LOAD * INLINE
[ Region, Year, Month, Costs
South, 2013,   May,    167
North, 2014,   May,    56
North, 2014,   June,   199
South, 2014,   June,   64
South, 2013,   May, 172
South, 2013,   May,    126
];
```

Теперь мы можем добавить визуализацию таблицы на лист и добавить поля Region, Year и Month, а также меры Sum для продаж и стоимости. Таблица будет выглядеть так:

Результирующая таблица

Region	Year	Month	Sum([Sales])	Sum([Costs])
Totals	-	-	1952	784
North	2014	June	127	199
North	2014	May	592	56
South	2014	June	645	64
South	2013	May	588	465

autonumberhash256

Эта функция скрипта вычисляет 256-битные случайные данные значений выражений комбинированного ввода и возвращает уникальное значение целого для каждого определенного значения случайных данных, возникающего в процессе выполнения скрипта. Эта функция может использоваться, например, при создании компактного представления сложного ключа в памяти.

*Можно подключить только ключи **autonumberhash256**, созданные в той же загрузке данных, поскольку целое число создается согласно порядку чтения таблицы. При использовании ключей, хранящихся между загрузками данных, независимо от сортировки исходных данных, необходимо использовать функции **hash128**, **hash160** или **hash256**.*

Синтаксис:

```
autonumberhash256(expression {, expression})
```

Пример: Создание составного ключа

В данном примере мы создаем составной ключ, используя функцию **autonumberhash256** для преобразования памяти. Этот пример представлен в целях демонстрации, поэтому в данном случае информация краткая, но при использовании таблицы, содержащей большое количество строк, информация будет более содержательной.

Пример таблицы

Region	Year	Month	Sales
North	2014	May	245
North	2014	May	347
North	2014	June	127
South	2014	June	645
South	2013	May	367
South	2013	May	221

Исходные данные загружаются с помощью встроенных данных. Затем мы добавляем предшествующую загрузку, которая создает составной ключ из полей Region, Year и Month.

```
RegionSales:
LOAD *,
AutoNumberHash256(Region, Year, Month) as RYMkey;

LOAD * INLINE
[ Region, Year, Month, Sales
North, 2014,   May,    245
North, 2014,   May,    347
North, 2014,   June,   127
South, 2014,   June,   645
South, 2013,   May, 367
South, 2013,   May,    221
];
```

Полученная таблица выглядит следующим образом:

Результирующая таблица

Region	Year	Month	Sales	RYMkey
North	2014	May	245	1
North	2014	May	347	1

Region	Year	Month	Sales	RYMkey
North	2014	June	127	2
South	2014	June	645	3
South	2013	May	367	4
South	2013	May	221	4

В этом примере вы можете обратиться к RYMkey, например 1 вместо строки 'North2014May', если необходимо установить связь с другой таблицей.

Теперь мы загружаем исходную таблицу с ценами похожим образом. Поля Region, Year и Month исключены предыдущей загрузкой во избежание создания синтетического ключа, мы уже создаем составной ключ с функцией **autonumberhash256**, связывая таблицы.

```
RegionCosts:
LOAD Costs,
AutoNumberHash256(Region, Year, Month) as RYMkey;

LOAD * INLINE
[ Region, Year, Month, Costs
South, 2013,   May,    167
North, 2014,   May,    56
North, 2014,   June,   199
South, 2014,   June,   64
South, 2013,   May, 172
South, 2013,   May,    126
];
```

Теперь мы можем добавить визуализацию таблицы на лист и добавить поля Region, Year и Month, а также меры Sum для продаж и стоимости. Таблица будет выглядеть так:

Результирующая таблица

Region	Year	Month	Sum([Sales])	Sum([Costs])
Totals	-	-	1952	784
North	2014	June	127	199
North	2014	May	592	56
South	2014	June	645	64
South	2013	May	588	465

IterNo

Эта функция скрипта возвращает целое, указывающее на то, в который раз оценивается одна запись в операторе **LOAD** предложением **while**. Первый шаг цикла – число 1. Функция **IterNo** имеет значение только при условии совместного

использования с предложением **while**.

Синтаксис:

```
IterNo( )
```

Примеры и результаты:

Пример:

```
LOAD
   IterNo() as Day,
   Date( StartDate + IterNo() - 1 ) as Date
   While StartDate + IterNo() - 1 <= EndDate;

LOAD * INLINE
[StartDate, EndDate
2014-01-22, 2014-01-26
];
```

Данный оператор **LOAD** генерирует одну запись на дату внутри диапазона, определенного параметрами **StartDate** и **EndDate**.

В этом примере полученная таблица будет выглядеть так:

Результирующая таблица

Day	Date
1	2014-01-22
2	2014-01-23
3	2014-01-24
4	2014-01-25
5	2014-01-26

RecNo

Эта функция скрипта возвращает целое число читаемой в текущий момент строки текущей таблицы. Первая запись – число 1.

Синтаксис:

```
RecNo( )
```

В отличие от функции **RowNo( )**, которая подсчитывает строки в результирующей таблице Qlik Sense, функция **RecNo( )** подсчитывает записи в таблице необработанных данных и сбрасывается при объединении таблицы необработанных данных с другой таблицей.

Пример: Скрипт загрузки данных

Загрузка таблицы с необработанными данными:

```
Tab1:
LOAD * INLINE
[A, B
1, aa
2,cc
3,ee];

Tab2:
LOAD * INLINE
[C, D
5, xx
4,yy
6,zz];
```

Загрузка номеров записей и строк для выбранных строк:

```
QTab:
LOAD *,
RecNo( ),
RowNo( )
resident Tab1 where A<>2;

LOAD
C as A,
D as B,
RecNo( ),
RowNo( )
resident Tab2 where A<>5;

//We don't need the source tables anymore, so we drop them
Drop tables Tab1, Tab2;
```

Результирующая внутренняя таблица Qlik Sense:

Результирующая таблица

A	B	RecNo( )	RowNo( )
1	aa	1	1
3	ee	3	2
4	yy	2	3
6	zz	3	4

RowNo

Эта функция возвращает целое значение позиции текущей строки в итоговой внутренней таблице Qlik Sense. Первая строка имеет номер 1.

Синтаксис:

```
RowNo( [TOTAL])
```

В отличие от **RecNo( )**, которая считает записи в таблице с необработанными данными, функция **RowNo( )** не считает записи, которые исключены предложениями **where**, и не сбрасывается, если таблица с необработанными данными объединена с другой.

*В случае использования предшествующего оператора load, то есть определенного числа операторов **LOAD** , собранных стопкой, считанных из одной таблицы, можно использовать только элемент **RowNo( )** в верхнем операторе **LOAD** . При использовании элемента **RowNo( )** в последовательных операторах **LOAD** , возвращается 0.*

Пример: Скрипт загрузки данных

Загрузка таблицы с необработанными данными:

```
Tab1:
LOAD * INLINE
[A, B
1, aa
2,cc
3,ee];

Tab2:
LOAD * INLINE
[C, D
5, xx
4,yy
6,zz];
```

Загрузка номеров записей и строк для выбранных строк:

```
QTab:
LOAD *,
RecNo( ),
RowNo( )
resident Tab1 where A<>2;

LOAD
C as A,
D as B,
RecNo( ),
RowNo( )
resident Tab2 where A<>5;

//We don't need the source tables anymore, so we drop them
Drop tables Tab1, Tab2;
```

Результирующая внутренняя таблица Qlik Sense:

Результирующая таблица

A	B	RecNo( )	RowNo( )
1	aa	1	1
3	ee	3	2
4	yy	2	3
6	zz	3	4

RowNo – функция диаграммы

Функция **RowNo()** возвращает текущие строки в текущий сегмент столбца в таблице. Для растровых диаграмм функция **RowNo()** возвращает текущие строки в эквивалент прямой таблицы диаграммы.

Если таблица или эквивалент таблицы имеют несколько вертикальных измерений, текущий сегмент столбца будет включать только строки с теми же значениями, что и текущая строка во всех столбцах измерений, кроме столбца с последним измерением в межполевом порядке сортировки.

Сегменты столбцов

> *Сортировка по y-значениям в диаграммах или сортировка по столбцам выражений в таблицах не допускается, если в каком-либо из выражений диаграммы используются функции **RowNo()**. Данные возможности сортировки автоматически отключаются.*

Синтаксис:

```
RowNo([TOTAL])
```

Возвращаемые типы данных: целое

Аргументы:

Аргумент	Описание
TOTAL	Если таблица имеет одно измерение, или если в качестве аргумента используется префикс **TOTAL**, текущий сегмент столбца всегда равен всему столбцу.

Пример: Выражение диаграммы с использованием RowNo

Пример: выражение диаграммы

Скрипт загрузки

Загрузите следующие данные через встроенную загрузку в редакторе загрузки данных, чтобы создать примеры с выражениями диаграммы, показанные ниже.

```
Temp:
 LOAD * inline [ Customer|Product|OrderNumber|UnitSales|UnitPrice Astrida|AA|1|4|16
Astrida|AA|7|10|15 Astrida|BB|4|9|9 Betacab|CC|6|5|10 Betacab|AA|5|2|20 Betacab|BB|1|25| 25
Canutility|AA|3|8|15 Canutility|CC|5|4|19 Divadip|CC|2|4|16 Divadip|DD|3|1|25 ] (delimiter is
'|');
```

Выражение диаграммы

Создайте на листе Qlik Sense визуализацию таблицы с измерениями **Customer** и **UnitSales**. Добавьте `RowNo( )` и `RowNo(TOTAL)` как меры с метками **Row in Segment** (Строка в сегменте) и **Row Number**, соответственно. Добавьте следующее выражение в таблицу в качестве меры:

```
If( RowNo( )=1, 0, UnitSales / Above( UnitSales ))
```

Результат

Customer	UnitSales	Row in Segment	Row Number	If(RowNo( )=1, 0, UnitSales / Above(UnitSales))
Astrida	4	1	1	0
Astrida	9	2	2	2.25
Astrida	10	3	3	1.1111111111111
Betacab	2	1	4	0
Betacab	5	2	5	2.5
Betacab	25	3	6	5
Canutility	4	1	7	0
Canutility	8	2	8	2
Divadip	1	1	9	0
Divadip	4	2	10	4

Объяснение

Столбец **Row in Segment** показывает результаты 1, 2, 3 для сегмента столбца, содержащего значения поля UnitSales для клиента Astrida. Нумерация строк для следующего сегмента столбца, который является Betacab, начинается в таком случае снова с 1.

Столбец **Row Number** игнорирует измерения из-за аргумента `TOTAL` для `RowNo()` и подсчитывает строки в таблице.

Это выражение возвращает значение 0 для первой строки в каждом сегменте столбца. Таким образом, в столбце отображается:

0, 2,25, 1,1111111, 0, 2,5, 5, 0, 2, 0 и 4.

См. также:

Above – функция диаграммы (page 689)

5.7 Функции даты и времени

Функции даты и времени Qlik Sense используются для преобразования значений даты и времени. Все функции можно использовать как в скрипте загрузки данных, так и в выражениях диаграмм.

Функции основываются на серийном номере даты-времени, который равен количеству дней с 30 декабря 1899 г. Целое значение представляет день, а дробное – время дня.

Программа Qlik Sense использует числовое значение параметра, поэтому число может использоваться в качестве параметра также и в тех случаях, когда оно не отформатировано в виде даты или времени. Если параметр не соответствует числовому значению, потому что, например, является строкой, то программа Qlik Sense пытается интерпретировать строку в соответствии с переменными окружения для даты и времени.

Если используемый в параметре формат времени не соответствует установленному в переменных окружения, программа Qlik Sense не сможет правильно выполнить интерпретацию. Для разрешения этой проблемы измените настройки или воспользуйтесь функцией интерпретации.

В примерах для каждой функции допускается время по умолчанию и форматы дат hh:mm:ss и YYYY-MM-DD (ISO 8601).

В ходе обработки метки времени с функцией даты или времени Qlik Sense не учитывает параметры перехода на летнее время, за исключением случаев, когда функция даты или времени включает географическое положение.

Например, `ConvertToLocalTime( filetime('Time.qvd'), 'Paris')` использует параметры перехода на летнее время, тогда как `ConvertToLocalTime(filetime('Time.qvd'), 'GMT-01:00')` не использует параметры перехода на летнее время.

Обзор функций даты и времени

Каждая функция подробно описана после обзора. Также можно щелкнуть имя функции в синтаксисе, чтобы получить немедленный доступ к подробной информации об этой конкретной функции.

Целочисленные выражения времени

second

Эта функция возвращает время в секундах в виде целого числа, а дробное выражение **expression** интерпретируется как время согласно стандартной интерпретации чисел.

```
second (expression)
```

minute

Эта функция возвращает время в минутах в виде целого числа, а дробное выражение **expression** интерпретируется как время согласно стандартной интерпретации чисел.

```
minute (expression)
```

hour

Эта функция возвращает время в часах в виде целого числа, а дробное выражение **expression** интерпретируется как время согласно стандартной интерпретации чисел.

```
hour (expression)
```

day

Эта функция возвращает день в виде целого числа, а дробное выражение **expression** интерпретируется как дата согласно стандартной интерпретации чисел.

```
day (expression)
```

week

Эта функция возвращает номер недели в виде целого числа согласно стандарту ISO 8601. Номер недели высчитывается на основе интерпретации данных выражения согласно стандартной интерпретации чисел.

```
week (expression)
```

month

Эта функция возвращает двойное значение с именем месяца, как определено переменной окружения **MonthNames**, и целое в диапазоне от 1 до 12. Месяц высчитывается на основе интерпретации данных выражения согласно стандартной интерпретации чисел.

```
month (expression)
```

year

Эта функция возвращает год в виде целого числа, а выражение **expression** интерпретируется как дата согласно стандартной интерпретации чисел.

```
year (expression)
```

weekyear

Эта функция возвращает год, которому принадлежит номер недели согласно стандарту ISO 8601. Номер недели в году может быть установлен в пределах от 1 до 52.

```
weekyear (expression)
```

weekday

Эта функция возвращает двойное значение со следующим: Имя дня, как определено переменной окружения **DayNames**. Целое от 0 до 6, соответствующее номинальному дню недели (0-6).

```
weekday (date)
```

Функции меток времени

now

Эта функция возвращает метку текущего времени по системным часам. Значение по умолчанию – 1.

```
now ([ timer_mode])
```

today

Эта функция возвращает текущую дату по системным часам.

```
today ([timer_mode])
```

LocalTime

Эта функция возвращает метку текущего времени по системным часам для указанного часового пояса.

```
localtime ([timezone [, ignoreDST ]])
```

Функции формирования

makedate

Эта функция возвращает дату, рассчитанную в формате год **YYYY**, месяц **MM** и день **DD**.

```
makedate (YYYY [ , MM [ , DD ] ])
```

makeweekdate

Эта функция возвращает дату, рассчитанную в формате год **YYYY**, неделя **WW** и день недели **D**.

```
makeweekdate (YYYY [ , WW [ , D ] ])
```

maketime

Эта функция возвращает время, рассчитанное в формате часы **hh**, минуты **mm** и секунды **ss**.

```
maketime (hh [ , mm [ , ss [ .fff ] ] ])
```

Другие функции даты

AddMonths

Эта функция возвращает дату через **n** месяцев после даты начала **startdate** или, если **n** является отрицательным числом, – дату за **n** месяцев до даты начала **startdate**.

```
addmonths (startdate, n , [ , mode])
```

AddYears

Эта функция возвращает дату через **n** лет после даты начала **startdate** или, если **n** является отрицательным числом, – дату за **n** лет до даты начала **startdate**.

```
addyears (startdate, n)
```

yeartodate

Эта функция определяет, находится ли введенная метка времени в том году, в котором находится дата последней загрузки скрипта, и возвращает значение True, если это так, и False если это не так.

```
yeartodate (date [ , yearoffset [ , firstmonth [ , todaydate] ] ])
```

Функции часовых поясов

timezone

Эта функция возвращает имя текущего часового пояса, соответствующее имени, используемому в Windows.

```
timezone ( )
```

GMT

Эта функция возвращает текущее среднее время Greenwich Mean Time согласно системным часам и настройкам времени в Windows.

```
GMT ( )
```

UTC

Возвращает текущее время Coordinated Universal Time.

```
UTC ( )
```

daylightsaving

Возвращает текущие настройки перехода на летнее время согласно установкам Windows.

```
daylightsaving ( )
```

converttolocaltime

Преобразует формат метки времени UTC или GMT в местное время и выводит в виде двойного значения. Местоположение может задаваться для любого числа городов, мест и часовых поясов Земли.

```
converttolocaltime (timestamp [, place [, ignore_dst=false]])
```

Функции установки времени

setdateyear

Данная функция берет в качестве входных значений **timestamp** и **year** и обновляет значение **timestamp** с учетом указанного входного значения **year** .

```
setdateyear  (timestamp, year)
```

setdateyearmonth

Данная функция берет в качестве входных значений **timestamp**, **month** и **year** и обновляет значение **timestamp** с учетом указанных входных значений **year** и **month** .

```
setdateyearmonth (timestamp, year, month)
```

Функции вхождения

inyear

Эта функция возвращает значение True, если поле **timestamp** находится в пределах года, включающего значение, указанное в поле **base_date**.

```
inyear (date, basedate , shift [, first_month_of_year = 1])
```

inyeartodate

Эта функция возвращает значение True, если значение **timestamp** находится в пределах части года, включающей значение, заданное в поле **base_date** до последней миллисекунды, указанной в поле **base_date**, включительно.

```
inyeartodate (date, basedate , shift [, first_month_of_year = 1])
```

inquarter

Эта функция возвращает значение True, если поле **timestamp** находится в пределах квартала, включающего значение, указанное в поле **base_date**.

```
inquarter (date, basedate , shift [, first_month_of_year = 1])
```

inquartertodate

Эта функция возвращает значение True, если значение **timestamp** находится в пределах части квартала, включающей значение, заданное в поле **base_date** до последней миллисекунды, указанной в поле **base_date**, включительно.

```
inquartertodate (date, basedate , shift [, first_month_of_year = 1])
```

inmonth

Эта функция возвращает значение True, если поле **timestamp** находится в пределах месяца, включающего значение, указанное в поле **base_date**.

```
inmonth (date, basedate , shift)
```

inmonthtodate

Возвращает значение True, если значение **date** находится в пределах части месяца, включающей значение, заданное в поле **basedate** до последней миллисекунды, указанной в поле **basedate**, включительно.

```
inmonthtodate (date, basedate , shift)
```

inmonths

Эта функция определяет, находится ли метка времени в том же месяце, двухмесячном периоде, квартале, триместре или полугодии, что и базовая дата. Также можно проследить, находится ли эта метка времени в предыдущем или последующем временном периоде.

```
inmonths (n, date, basedate , shift [, first_month_of_year = 1])
```

inmonthstodate

Эта функция определяет, находится ли метка времени в части месяца, двухмесячного периода, квартала, триместра или полугодия до последней миллисекунды, указанной в поле **base_date**, включительно. Также можно проследить, находится ли метка времени в предыдущем или в последующем временном периоде.

```
inmonthstodate  (n, date, basedate , shift [, first_month_of_year = 1])
```

inweek

Эта функция возвращает значение True, если поле **timestamp** находится в пределах недели, включающей значение, указанное в поле **base_date**.

```
inweek (date, basedate , shift [, weekstart])
```

inweektodate

Эта функция возвращает значение True, если значение **timestamp** находится в пределах части недели, включающей значение, заданное в поле **base_date** до последней миллисекунды, указанной в поле **base_date**, включительно.

```
inweektodate  (date, basedate , shift [, weekstart])
```

inlunarweek

Эта функция определяет, находится ли значение **timestamp** в пределах лунной недели, включающей значение, указанное в поле **base_date**. Лунные недели в Qlik Sense определяются от 1 января как первого дня недели.

```
inlunarweek  (date, basedate , shift [, weekstart])
```

inlunarweektodate

Эта функция определяет, находится ли значение **timestamp** в пределах части лунной недели до последней миллисекунды, указанной в поле **base_date**, включительно. Лунные недели в Qlik Sense определяются от 1 января как первого дня недели.

```
inlunarweektodate  (date, basedate , shift [, weekstart])
```

inday

Эта функция возвращает значение True, если поле **timestamp** находится в пределах дня, включающего значение, указанное в поле **base_timestamp**.

```
inday (timestamp, basetimestamp , shift [, daystart])
```

indaytotime

Эта функция возвращает значение True, если значение **timestamp** находится в пределах части дня, включающей значение, заданное в поле **base_timestamp** до определенной миллисекунды, указанной в поле **base_timestamp**, включительно.

```
indaytotime (timestamp, basetimestamp , shift [, daystart])
```

Функции начала и конца

yearstart

Эта функция возвращает метку времени, соответствующую началу первого дня года, содержащего значение **date**. По умолчанию для вывода используется формат **DateFormat**, установленный в скрипте.

```
yearstart ( date [, shift = 0 [, first_month_of_year = 1]])
```

yearend

Эта функция возвращает значение, соответствующее метке времени, включающей последнюю миллисекунду последнего дня года, содержащего значение, указанное в поле **date**. По умолчанию для вывода используется формат **DateFormat**, установленный в скрипте.

```
yearend ( date [, shift = 0 [, first_month_of_year = 1]])
```

yearname

Эта функция возвращает 4-значное значение года с базовым числовым значением, соответствующим метке времени с первой миллисекундой первого дня года, содержащего значение, указанное в поле **date**.

```
yearname (date [, shift = 0 [, first_month_of_year = 1]] )
```

quarterstart

Эта функция возвращает значение, соответствующее метке времени, включающей первую миллисекунду квартала, содержащего значение, указанное в поле **date**. По умолчанию для вывода используется формат **DateFormat**, установленный в скрипте.

```
quarterstart (date [, shift = 0 [, first_month_of_year = 1]])
```

quarterend

Эта функция возвращает значение, соответствующее метке времени, включающей последнюю миллисекунду квартала, содержащего значение, указанное в поле **date**. По умолчанию для вывода используется формат **DateFormat**, установленный в скрипте.

```
quarterend (date [, shift = 0 [, first_month_of_year = 1]])
```

quartername

Эта функция возвращает значение, отображающее месяцы квартала (в формате переменной **MonthNames** скрипта) и год с базовым числовым значением, соответствующим метке времени, включающей первую миллисекунду первого дня квартала.

```
quartername (date [, shift = 0 [, first_month_of_year = 1]])
```

monthstart

Эта функция возвращает значение, соответствующее метке времени, включающей первую миллисекунду первого дня месяца, содержащего значение, указанное в поле **date**. По умолчанию для вывода используется формат **DateFormat**, установленный в скрипте.

```
monthstart (date [, shift = 0])
```

monthend

Эта функция возвращает значение, соответствующее метке времени, включающей последнюю миллисекунду последнего дня месяца, содержащего значение, указанное в поле **date**. По умолчанию для вывода используется формат **DateFormat**, установленный в скрипте.

```
monthend (date [, shift = 0])
```

monthname

Эта функция возвращает значение, отображающее месяц (в формате переменной **MonthNames** скрипта) и год с базовым числовым значением, соответствующим метке времени, включающей первую миллисекунду первого дня указанного месяца.

```
monthname (date [, shift = 0])
```

monthsstart

Эта функция возвращает значение, соответствующее метке времени первой миллисекунды месяца, двухмесячного периода, квартала, триместра или полугодия, содержащих базовую дату. Также можно найти метку времени для предыдущего или последующего временного периода.

```
monthsstart (n, date [, shift = 0 [, first_month_of_year = 1]])
```

monthsend

Эта функция возвращает значение, соответствующее метке времени последней миллисекунды месяца, двухмесячного периода, квартала, триместра или полугодия, содержащих базовую дату. Также можно найти метку времени для предыдущего или последующего временного периода.

```
monthsend (n, date [, shift = 0 [, first_month_of_year = 1]])
```

monthsname

Эта функция возвращает значение, представляющее диапазон месяцев периода (форматированного согласно переменным скрипта **MonthNames**), а также года. Базовое числовое значение соответствует метке времени первой миллисекунды месяца, двухмесячного периода, квартала, триместра или полугодия, содержащих базовую дату.

```
monthsname (n, date [, shift = 0 [, first_month_of_year = 1]])
```

weekstart

Эта функция возвращает значение, соответствующее метке времени, включающей первую миллисекунду первого дня (понедельника) календарной недели, содержащего значение, указанное в поле **date**. По умолчанию для вывода используется формат **DateFormat**, установленный в скрипте.

```
weekstart (date [, shift = 0 [,weekoffset = 0]])
```

weekend

Эта функция возвращает значение, соответствующее метке времени, включающей последнюю миллисекунду последней даты (воскресенья) календарной недели, включающей дату, заданную в поле **date**. По умолчанию для вывода используется формат даты **DateFormat**, установленный в скрипте.

```
weekend (date [, shift = 0 [,weekoffset = 0]])
```

weekname

Эта функция возвращает значение года и номер недели с базовым числовым значением, соответствующим метке времени, включающей первую миллисекунду первого дня недели, содержащего значение, указанное в поле **date**.

```
weekname (date [, shift = 0 [,weekoffset = 0]])
```

lunarweekstart

Эта функция возвращает значение, соответствующее метке времени первой миллисекунды лунной недели, содержащей значение, указанное в поле **date**. Лунные недели в Qlik Sense определяются от 1 января как первого дня недели.

```
lunarweekstart (date [, shift = 0 [,weekoffset = 0]])
```

lunarweekend

Эта функция возвращает значение, соответствующее метке времени последней миллисекунды лунной недели, содержащей значение, указанное в поле **date**. Лунные недели в Qlik Sense определяются от 1 января как первого дня недели.

```
lunarweekend (date [, shift = 0 [,weekoffset = 0]])
```

lunarweekname

Эта функция возвращает значение года и номер лунной недели, соответствующие метке времени первой миллисекунды первого дня лунной недели, содержащего значение, указанное в поле **date**. Лунные недели в Qlik Sense определяются от 1 января как первого дня недели.

```
lunarweekname (date [, shift = 0 [,weekoffset = 0]])
```

daystart

Эта функция возвращает значение, соответствующее метке времени, включающей первую миллисекунду дня, содержащуюся в аргументе **time**. По умолчанию для вывода используется формат **TimestampFormat**, установленный в скрипте.

```
daystart (timestamp [, shift = 0 [, dayoffset = 0]])
```

dayend

Эта функция возвращает значение, соответствующее метке времени, включающей последнюю миллисекунду дня, содержащуюся в поле **time**. По умолчанию для вывода используется формат **TimestampFormat**, установленный в скрипте.

```
dayend (timestamp [, shift = 0 [, dayoffset = 0]])
```

dayname

Эта функция возвращает значение даты с базовым числовым значением, соответствующим метке времени, включающей первую миллисекунду дня, содержащего значение, указанное в поле **time**.

```
dayname (timestamp [, shift = 0 [, dayoffset = 0]])
```

Функции нумерации дней

age

Функция **age** возвращает значение возраста в момент времени, заданный в поле **timestamp** (полных лет), человека, дата рождения которого указана в поле **date_of_birth**.

```
age (timestamp, date_of_birth)
```

networkdays

Функция **networkdays** возвращает число рабочих дней (понедельник-пятница) между и включая значения, указанные в поле **start_date** и **end_date**, учитывая выходные, которые можно дополнительно задать в поле **holiday**.

```
networkdays  (start:date, end_date {, holiday})
```

firstworkdate

Функция **firstworkdate** возвращает самую позднюю дату начала, при которой период, заданный в поле **no_of_workdays** (понедельник-пятница), окончится не позднее даты, заданной в поле **end_date**, с учетом возможных выходных. **end_date** и **holiday** должны быть действительными датами или метками времени.

```
firstworkdate (end_date, no_of_workdays {, holiday} )
```

lastworkdate

Функция **lastworkdate** возвращает самую раннюю дату окончания для достижения указанного числа рабочих дней **no_of_workdays** (понедельник-пятница) с начальной датой **start_date** и с учетом выходных, которые можно дополнительно задать в поле **holiday**. Поля **start_date** и **holiday** должны быть действительными датами или метками времени.

```
lastworkdate (start_date, no_of_workdays {, holiday})
```

daynumberofyear

Эта функция вычисляет номер дня года, на который приходится метка времени. Вычисление выполняется с первой миллисекунды первого дня года, но первый месяц может быть смещен.

```
daynumberofyear (date[,firstmonth])
```

daynumberofquarter

Эта функция вычисляет номер дня квартала, на который приходится метка времени.

```
daynumberofquarter (date[,firstmonth])
```

addmonths

Эта функция возвращает дату через **n** месяцев после даты начала **startdate** или, если **n** является отрицательным числом, – дату за **n** месяцев до даты начала **startdate**.

Синтаксис:

```
AddMonths(startdate, n , [ , mode])
```

Возвращаемые типы данных: двойное значение

Функция AddMonths возвращает двойное значение, которое включает строковое и числовое значения. Функция берет числовое значение входного выражения и создает строку, представляющую собой число. Отображается строка, а числовое значение используется для вычислений и сортировки.

Аргументы:

Аргументы

Аргумент	Описание
startdate	Начальная дата в виде метки времени, например '2012-10-12'.
n	Количество месяцев в виде положительного или отрицательного целого числа.
mode	Указывает, добавляется ли месяц относительно начала или конца месяца. Режим по умолчанию при добавлении относительно начала месяца – 0. Для добавлений относительно конца месяца установите режим 1. Если установлен режим 1 и введенная дата – 28-е число или выше, функция оценивает количество дней, оставшихся от начальной даты до конца месяца. То же количество дней до конца месяца устанавливается для возвращенной даты.

Примеры и результаты:

Примеры написания скриптов

Пример	Результат
`addmonths ('2003-01-29',3)`	Возвращает «2003-04-29»
`addmonths ('2003-01-29',3,0)`	Возвращает «2003-04-29»
`addmonths ('2003-01-29',3,1)`	Возвращает «2003-04-28»
`addmonths ('2003-01-29',1,0)`	Возвращает «2003-02-28»
`addmonths ('2003-01-29',1,1)`	Возвращает «2003-02-26»
`addmonths ('2003-02-28',1,0)`	Возвращает «2003-03-28»
`addmonths ('2003-02-28',1,1)`	Возвращает «2003-03-31»
`addmonths ('2003-01-29',-3)`	Возвращает «2002-10-29»

addyears

Эта функция возвращает дату через **n** лет после даты начала **startdate** или, если **n** является отрицательным числом, – дату за **n** лет до даты начала **startdate**.

Синтаксис:

```
AddYears(startdate, n)
```

Возвращаемые типы данных: двойное значение

Аргументы:

Аргументы

Аргумент	Описание
startdate	Начальная дата в виде метки времени, например '2012-10-12'.
n	Количество лет в виде положительного или отрицательного целого числа.

Примеры и результаты:

Примеры написания скриптов

Пример	Результат
`addyears ('2010-01-29',3)`	Возвращает «2013-01-29»
`addyears ('2010-01-29',-1)`	Возвращает «2009-01-29»

age

Функция **age** возвращает значение возраста в момент времени, заданный в поле **timestamp** (полных лет), человека, дата рождения которого указана в поле **date_of_birth**.

Синтаксис:

```
age(timestamp, date_of_birth)
```

Может быть выражением.

Возвращаемые типы данных: числовое значение

Аргументы:

Аргументы

Аргумент	Описание
timestamp	Метка времени или выражение, определяемое по метке времени, до которой необходимо вычислить завершенное количество лет.
date_of_birth	Дата рождения человека, возраст которого вычисляется. Может быть выражением.

Примеры и результаты:
В этих примерах используется формат даты **DD/MM/YYYY**. Формат даты указан в операторе **SET DateFormat** в верхней части скрипта загрузки данных. Измените формат в примерах согласно своим пожеланиям.

Примеры написания скриптов

Пример	Результат
`age('25/01/2014', '29/10/2012')`	Возвращает 1.
`age('29/10/2014', '29/10/2012')`	Возвращает 2.

Пример:

Добавьте образец скрипта в свое приложение и запустите. Чтобы увидеть результаты, добавьте поля, перечисленные в столбце результатов, в лист приложения.

```
Employees:
LOAD * INLINE [
Member|DateOfBirth
John|28/03/1989
Linda|10/12/1990
Steve|5/2/1992
Birg|31/3/1993
Raj|19/5/1994
Prita|15/9/1994
Su|11/12/1994
Goran|2/3/1995
Sunny|14/5/1996
Ajoa|13/6/1996
Daphne|7/7/1998
Biffy|4/8/2000
] (delimiter is |);
AgeTable:
Load *,
age('20/08/2015', DateOfBirth) As Age
Resident Employees;
Drop table Employees;
```

Результирующая таблица показывает возвращенные значения функции age для каждой записи в таблице.

Результирующая таблица

Member	DateOfBirth	Age
John	28/03/1989	26
Linda	10/12/1990	24
Steve	5/2/1992	23
Birg	31/3/1993	22

Member	DateOfBirth	Age
Raj	19/5/1994	21
Prita	15/9/1994	20
Su	11/12/1994	20
Goran	2/3/1995	20
Sunny	14/5/1996	19
Ajoa	13/6/1996	19
Daphne	7/7/1998	17
Biffy	4/8/2000	15

converttolocaltime

Преобразует формат метки времени UTC или GMT в местное время и выводит в виде двойного значения. Местоположение может задаваться для любого числа городов, мест и часовых поясов Земли.

Синтаксис:

```
ConvertToLocalTime(timestamp [, place [, ignore_dst=false]])
```

Возвращаемые типы данных: двойное значение

Аргументы:

Аргументы

Аргумент	Описание
timestamp	Метка времени или выражение, определяемое по метке времени, для преобразования.

Аргумент	Описание
place	Город или часовой пояс из таблицы действительных городов и часовых поясов, указанной ниже. Либо можно использовать GMT или UTC для определения местного времени. Следующие значения и диапазоны смещения времени являются действительными. • GMT • GMT-12:00 - GMT-01:00 • GMT+01:00 - GMT+14:00 • UTC • UTC-12:00 - UTC-01:00 • UTC+01:00 - UTC+14:00 *Можно использовать только стандартные значения смещения времени. Невозможно использовать произвольное смещение времени, например GMT-04:27.*
ignore_dst	Установите значение True, чтобы игнорировать DST (переход на летнее время).

Результирующее время настраивается в соответствии с переходом на летнее время, если для параметра **ignore_dst** не задано значение True.

Действительные города и часовые пояса

A-C	D-K	L-R	S-Z
Abu Dhabi	Darwin	La Paz	Samoa
Adelaide	Dhaka	Lima	Santiago
Alaska	Eastern Time (US & Canada)	Lisbon	Sapporo
Amsterdam	Edinburgh	Ljubljana	Sarajevo
Arizona	Ekaterinburg	London	Saskatchewan
Astana	Fiji	Madrid	Seoul
Athens	Georgetown	Magadan	Singapore
Atlantic Time (Canada)	Greenland	Mazatlan	Skopje
Auckland	Greenwich Mean Time : Dublin	Melbourne	Sofia
Azores	Guadalajara	Mexico City	Solomon Is.
Baghdad	Guam	Mid-Atlantic	Sri Jayawardenepura

A-C	D-K	L-R	S-Z
Baku	Hanoi	Minsk	St. Petersburg
Bangkok	Harare	Monrovia	Stockholm
Beijing	Hawaii	Monterrey	Sydney
Belgrade	Helsinki	Moscow	Taipei
Berlin	Hobart	Mountain Time (US & Canada)	Tallinn
Bern	Hong Kong	Mumbai	Tashkent
Bogota	Indiana (East)	Muscat	Tbilisi
Brasilia	International Date Line West	Nairobi	Tehran
Bratislava	Irkutsk	New Caledonia	Tokyo
Brisbane	Islamabad	New Delhi	Urumqi
Brussels	Istanbul	Newfoundland	Warsaw
Bucharest	Jakarta	Novosibirsk	Wellington
Budapest	Jerusalem	Nuku'alofa	West Central Africa
Buenos Aires	Kabul	Osaka	Vienna
Cairo	Kamchatka	Pacific Time (US & Canada)	Vilnius
Canberra	Karachi	Paris	Vladivostok
Cape Verde Is.	Kathmandu	Perth	Volgograd
Caracas	Kolkata	Port Moresby	Yakutsk
Casablanca	Krasnoyarsk	Prague	Yerevan
Central America	Kuala Lumpur	Pretoria	Zagreb
Central Time (US & Canada)	Kuwait	Quito	-
Chennai	Kyiv	Riga	-
Chihuahua	-	Riyadh	-
Chongqing	-	Rome	-
Copenhagen	-	-	-

Примеры и результаты:

Примеры написания скриптов

Пример	Результат
`ConvertToLocalTime('2007-11-10 23:59:00','Paris')`	Возвращает «2007-11-11 00:59:00» и соответствующее внутреннее представление метки времени.
`ConvertToLocalTime(UTC(), 'GMT-05:00')`	Возвращает время для североамериканского восточного побережья, например Нью-Йорка.
`ConvertToLocalTime(UTC(), 'GMT-05:00', True)`	Возвращает время для североамериканского восточного побережья, например Нью-Йорка, без учета соответствия переходу на летнее время.

day

Эта функция возвращает день в виде целого числа, а дробное выражение **expression** интерпретируется как дата согласно стандартной интерпретации чисел.

Функция возвращает день месяца для определенной даты. Она широко используется с целью получения значения для поля дня в составе измерения календаря.

Синтаксис:

```
day(expression)
```

Возвращаемые типы данных: целое

Примеры функции

Пример	Результат
`day( '1971-10-12' )`	возвращает 12
`day( '35648' )`	возвращает 6, так как 35648 = 1997-08-06

dayend

Эта функция возвращает значение, соответствующее метке времени, включающей последнюю миллисекунду дня, содержащуюся в поле **time**. По умолчанию для вывода используется формат **TimestampFormat**, установленный в скрипте.

Синтаксис:

```
DayEnd(time[, [period_no[, day_start]])
```

Возвращаемые типы данных: двойное значение

Аргументы:

Аргументы

Аргумент	Описание
time	Метка времени для вычисления.
period_no	**period_no** является целым числом или выражением, определяемым по целому числу, где значение 0 означает день, содержащий значение, указанное в поле**time**. Отрицательные значения, заданные в поле **period_no**, означают предшествующие дни, положительные – последующие.
day_start	Чтобы указать дни, которые начинаются не в полночь, укажите смещение в виде десятичного значения в параметре **day_start**. Например 0,125 обозначает 3:00 (3 AM).

Примеры и результаты:
В этих примерах используется формат даты **DD/MM/YYYY**. Формат даты указан в операторе **SET DateFormat** в верхней части скрипта загрузки данных. Измените формат в примерах согласно своим пожеланиям.

Примеры написания скриптов

Пример	Результат
`dayend('25/01/2013 16:45:00')`	Возвращает 25/01/2013 23:59:59.
`dayend('25/01/2013 16:45:00', -1)`	Возвращает 24/01/2013 23:59:59.
`dayend('25/01/2013 16:45:00', 0, 0.5)`	Возвращает 26/01/2013 11:59:59.

Пример:

Добавьте образец скрипта в свое приложение и запустите. Чтобы увидеть результаты, добавьте поля, перечисленные в столбце результатов, в лист приложения.

В этом примере обнаружена метка времени, которая отмечает окончание дня после даты каждого счета в таблице.

```
TempTable:
LOAD RecNo() as InvID, * Inline [
InvDate
28/03/2012
10/12/2012
5/2/2013
31/3/2013
19/5/2013
15/9/2013
11/12/2013
2/3/2014
```

```
14/5/2014
13/6/2014
7/7/2014
4/8/2014
];

InvoiceData:
LOAD *,
DayEnd(InvDate, 1) AS DEnd
Resident TempTable;
Drop table TempTable;
```

Результирующая таблица содержит исходные даты и столбец с возвращенным значением функции dayend(). Чтобы отобразить полную метку времени, укажите форматирование на панели свойств.

Результирующая таблица

InvDate	DEnd
28/03/2012	29/03/2012 23:59:59
10/12/2012	11/12/2012 23:59:59
5/2/2013	07/02/2013 23:59:59
31/3/2013	01/04/2013 23:59:59
19/5/2013	20/05/2013 23:59:59
15/9/2013	16/09/2013 23:59:59
11/12/2013	12/12/2013 23:59:59
2/3/2014	03/03/2014 23:59:59
14/5/2014	15/05/2014 23:59:59
13/6/2014	14/06/2014 23:59:59
7/7/2014	08/07/2014 23:59:59
4/8/2014	05/08/2014 23:59:59

daylightsaving

Возвращает текущие настройки перехода на летнее время согласно установкам Windows.

Синтаксис:

```
DaylightSaving( )
```

Возвращаемые типы данных: dual

Пример:

```
daylightsaving( )
```

dayname

Эта функция возвращает значение даты с базовым числовым значением, соответствующим метке времени, включающей первую миллисекунду дня, содержащего значение, указанное в поле **time**.

Синтаксис:

```
DayName(time[, period_no [, day_start]])
```

Возвращаемые типы данных: двойное значение

Аргументы:

Аргументы

Аргумент	Описание
time	Метка времени для вычисления.
period_no	**period_no** является целым числом или выражением, определяемым по целому числу, где значение 0 означает день, содержащий значение, указанное в поле**time**. Отрицательные значения, заданные в поле **period_no**, означают предшествующие дни, положительные – последующие.
day_start	Чтобы указать дни, которые начинаются не в полночь, укажите смещение в виде десятичного значения в параметре **day_start**. Например 0,125 обозначает 3:00 (3 AM).

Примеры и результаты:

В этих примерах используется формат даты **DD/MM/YYYY**. Формат даты указан в операторе **SET DateFormat** в верхней части скрипта загрузки данных. Измените формат в примерах согласно своим пожеланиям.

Примеры написания скриптов

Пример	Результат
`dayname('25/01/2013 16:45:00')`	Возвращает 25/01/2013.
`dayname('25/01/2013 16:45:00', -1)`	Возвращает 24/01/2013.
`dayname('25/01/2013 16:45:00', 0, 0.5 )`	Возвращает 25/01/2013. Отображение полной метки времени с базовым числовым значением, соответствующим 25/01/2013 12:00:00.000.

Пример:

Добавьте образец скрипта в свое приложение и запустите. Чтобы увидеть результаты, добавьте поля, перечисленные в столбце результатов, в лист приложения.

В этом примере имя дня создано из метки времени, которая отмечает начало дня после даты каждого счета в таблице.

```
TempTable:
LOAD RecNo() as InvID, * Inline [
InvDate
28/03/2012
10/12/2012
5/2/2013
31/3/2013
19/5/2013
15/9/2013
11/12/2013
2/3/2014
14/5/2014
13/6/2014
7/7/2014
4/8/2014
];

InvoiceData:
LOAD *,
DayName(InvDate, 1) AS DName
Resident TempTable;
Drop table TempTable;
```

Результирующая таблица содержит исходные даты и столбец с возвращенным значением функции dayname(). Чтобы отобразить полную метку времени, укажите форматирование на панели свойств.

Результирующая таблица

InvDate	DName
28/03/2012	29/03/2012 00:00:00
10/12/2012	11/12/2012 00:00:00
5/2/2013	07/02/2013 00:00:00
31/3/2013	01/04/2013 00:00:00
19/5/2013	20/05/2013 00:00:00
15/9/2013	16/09/2013 00:00:00
11/12/2013	12/12/2013 00:00:00
2/3/2014	03/03/2014 00:00:00
14/5/2014	15/05/2014 00:00:00

InvDate	DName
13/6/2014	14/06/2014 00:00:00
7/7/2014	08/07/2014 00:00:00
4/8/2014	05/08/2014 00:00:00

daynumberofquarter

Эта функция вычисляет номер дня квартала, на который приходится метка времени.

Синтаксис:

```
DayNumberOfQuarter(timestamp[,start_month])
```

Возвращаемые типы данных: целое

В этой функции год всегда включает 366 дней.

Аргументы:

Аргументы

Аргумент	Описание
timestamp	Дата для вычисления.
start_month	Если в поле **start_month** задать значение от 2 до 12 (1, если значение не указано), то начало года может быть передвинуто вперед на первый день любого месяца. Если, например, необходимо работать в рамках финансового года, начинающегося 1 марта, задайте **start_month** = 3.

Примеры и результаты:

В этих примерах используется формат даты **DD/MM/YYYY**. Формат даты указан в операторе **SET DateFormat** в верхней части скрипта загрузки данных. Измените формат в примерах согласно своим пожеланиям.

Примеры написания скриптов

Пример	Результат
`DayNumberOfQuarter('12/09/2014')`	Возвращает 74, номер дня текущего квартала.
`DayNumberOfQuarter ('12/09/2014',3)`	Возвращает 12, номер дня текущего квартала. В этом случае первый квартал начинается с марта (поскольку элемент start_month указан как 3). Это означает, что текущий квартал является третьим кварталом, который начался первого сентября.

Пример:

Добавьте образец скрипта в свое приложение и запустите. Чтобы увидеть результаты, добавьте поля, перечисленные в столбце результатов, в лист приложения.

```
ProjectTable:
LOAD recno() as InvID, * INLINE [
StartDate
28/03/2014
10/12/2014
5/2/2015
31/3/2015
19/5/2015
15/9/2015
] ;
NrDays:
Load *,
DayNumberOfQuarter(StartDate,4) As DayNrQtr
Resident ProjectTable;
Drop table ProjectTable;
```

Результирующая таблица показывает возвращенные значения функции DayNumberOfQuarter для каждой записи в таблице.

Результирующая таблица

InvID	StartDate	DayNrQtr
1	28/03/2014	88
2	10/12/2014	71
3	5/2/2015	36
4	31/3/2015	91
5	19/5/2015	49
6	15/9/2015	77

daynumberofyear

Эта функция вычисляет номер дня года, на который приходится метка времени. Вычисление выполняется с первой миллисекунды первого дня года, но первый месяц может быть смещен.

Синтаксис:

```
DayNumberOfYear(timestamp[,start_month])
```

Возвращаемые типы данных: целое

В этой функции год всегда включает 366 дней.

Аргументы:

Аргументы

Аргумент	Описание
timestamp	Дата для вычисления.
start_month	Если в поле **start_month** задать значение от 2 до 12 (1, если значение не указано), то начало года может быть передвинуто вперед на первый день любого месяца. Если, например, необходимо работать в рамках финансового года, начинающегося 1 марта, задайте **start_month** = 3.

Примеры и результаты:
В этих примерах используется формат даты **DD/MM/YYYY**. Формат даты указан в операторе **SET DateFormat** в верхней части скрипта загрузки данных. Измените формат в примерах согласно своим пожеланиям.

Примеры написания скриптов

Пример	Результат
`DayNumberOfYear ('12/09/2014')`	Возвращает 256, номер дня, отсчет которого начинается с первого дня года.
`DayNumberOfYear ('12/09/2014',3)`	Возвращает 196, номер дня, отсчет которого начинается с первого марта.

Пример:

Добавьте образец скрипта в свое приложение и запустите. Чтобы увидеть результаты, добавьте поля, перечисленные в столбце результатов, в лист приложения.

```
ProjectTable:
LOAD recno() as InvID, * INLINE [
StartDate
28/03/2014
10/12/2014
5/2/2015
31/3/2015
19/5/2015
15/9/2015
] ;
NrDays:
Load *,
DayNumberOfYear(StartDate,4) As DayNrYear
Resident ProjectTable;
Drop table ProjectTable;
```

Результирующая таблица показывает возвращенные значения функции DayNumberOfYear для каждой записи в таблице.

Результирующая таблица

InvID	StartDate	DayNrYear
1	28/03/2014	363
2	10/12/2014	254
3	5/2/2015	311
4	31/3/2015	366
5	19/5/2015	49
6	15/9/2015	168

daystart

Эта функция возвращает значение, соответствующее метке времени, включающей первую миллисекунду дня, содержащуюся в аргументе **time**. По умолчанию для вывода используется формат **TimestampFormat**, установленный в скрипте.

Синтаксис:

```
DayStart(time[, [period_no[, day_start]])
```

Возвращаемые типы данных: двойное значение

Аргументы:

Аргументы

Аргумент	Описание
time	Метка времени для вычисления.
period_no	**period_no** является целым числом или выражением, определяемым по целому числу, где значение 0 означает день, содержащий значение, указанное в поле**time**. Отрицательные значения, заданные в поле **period_no**, означают предшествующие дни, положительные – последующие.
day_start	Чтобы указать дни, которые начинаются не в полночь, укажите смещение в виде десятичного значения в параметре **day_start**. Например 0,125 обозначает 3:00 (3 AM).

Примеры и результаты:
В этих примерах используется формат даты **DD/MM/YYYY**. Формат даты указан в операторе **SET DateFormat** в верхней части скрипта загрузки данных. Измените формат в примерах согласно своим пожеланиям.

Примеры написания скриптов

Пример	Результат
`daystart('25/01/2013 16:45:00')`	Возвращает 25/01/2013 00:00:00.
`daystart('25/01/2013 16:45:00', -1)`	Возвращает 24/01/2013 00:00:00.
`daystart('25/01/2013 16:45:00', 0, 0.5)`	Возвращает 25/01/2013 12:00:00.

Пример:

Добавьте образец скрипта в свое приложение и запустите. Чтобы увидеть результаты, добавьте поля, перечисленные в столбце результатов, в лист приложения.

В этом примере обнаружена метка времени, которая отмечает начало дня после даты каждого счета в таблице.

```
TempTable:
LOAD RecNo() as InvID, * Inline [
InvDate
28/03/2012
10/12/2012
5/2/2013
31/3/2013
19/5/2013
15/9/2013
11/12/2013
2/3/2014
14/5/2014
13/6/2014
7/7/2014
4/8/2014
];

InvoiceData:
LOAD *,
DayStart(InvDate, 1) AS DStart
Resident TempTable;
Drop table TempTable;
```

Результирующая таблица содержит исходные даты и столбец с возвращенным значением функции daystart(). Чтобы отобразить полную метку времени, укажите форматирование на панели свойств.

Результирующая таблица

InvDate	DStart
28/03/2012	29/03/2012 00:00:00
10/12/2012	11/12/2012 00:00:00
5/2/2013	07/02/2013 00:00:00

InvDate	DStart
31/3/2013	01/04/2013 00:00:00
19/5/2013	20/05/2013 00:00:00
15/9/2013	16/09/2013 00:00:00
11/12/2013	12/12/2013 00:00:00
2/3/2014	03/03/2014 00:00:00
14/5/2014	15/05/2014 00:00:00
13/6/2014	14/06/2014 00:00:00
7/7/2014	08/07/2014 00:00:00
4/8/2014	05/08/2014 00:00:00

firstworkdate

Функция **firstworkdate** возвращает самую позднюю дату начала, при которой период, заданный в поле **no_of_workdays** (понедельник-пятница), окончится не позднее даты, заданной в поле **end_date**, с учетом возможных выходных. **end_date** и **holiday** должны быть действительными датами или метками времени.

Синтаксис:

```
firstworkdate(end_date, no_of_workdays {, holiday} )
```

Возвращаемые типы данных: целое

Аргументы:

Аргументы

Аргумент	Описание
end_date	Метка времени даты окончания для оценки.
no_of_ workdays	Количество рабочих дней, которое должно быть получено.
holiday	Периоды выходных дней для исключения из рабочих дней. Период выходных дней указан как дата начала и дата окончания, разделенные запятыми. **Пример:** `'25/12/2013', '26/12/2013'` Можно указать несколько периодов выходных дней, разделенных запятыми. **Пример:** `'25/12/2013', '26/12/2013', '31/12/2013', '01/01/2014'`

Примеры и результаты:
В этих примерах используется формат даты **DD/MM/YYYY**. Формат даты указан в операторе **SET DateFormat** в верхней части скрипта загрузки данных. Измените формат в примерах согласно своим пожеланиям.

Примеры написания скриптов

Пример	Результат
`firstworkdate ('29/12/2014', 9)`	Возвращает 17/12/2014.
`firstworkdate ('29/12/2014', 9, '25/12/2014', '26/12/2014')`	Возвращает 15/12/2014, поскольку учитывается двухдневный период выходных.

Пример:

Добавьте образец скрипта в свое приложение и запустите. Чтобы увидеть результаты, добавьте поля, перечисленные в столбце результатов, в лист приложения.

```
ProjectTable:
LOAD *, recno() as InvID, INLINE [
EndDate
28/03/2015
10/12/2015
5/2/2016
31/3/2016
19/5/2016
15/9/2016
] ;
NrDays:
Load *,
FirstWorkDate(EndDate,120) As StartDate
Resident ProjectTable;
Drop table ProjectTable;
```

Результирующая таблица показывает возвращенные значения функции FirstWorkDate для каждой записи в таблице.

Результирующая таблица

InvID	EndDate	StartDate
1	28/03/2015	13/10/2014
2	10/12/2015	26/06/2015
3	5/2/2016	24/08/2015
4	31/3/2016	16/10/2015
5	19/5/2016	04/12/2015
6	15/9/2016	01/04/2016

GMT

Эта функция возвращает текущее среднее время Greenwich Mean Time согласно системным часам и настройкам времени в Windows.

Синтаксис:

```
GMT( )
```

Возвращаемые типы данных: dual

Пример:

```
gmt( )
```

hour

Эта функция возвращает время в часах в виде целого числа, а дробное выражение **expression** интерпретируется как время согласно стандартной интерпретации чисел.

Синтаксис:

```
hour(expression)
```

Возвращаемые типы данных: целое

Примеры и результаты:

Примеры написания скриптов

Пример	Результат
`hour( '09:14:36' )`	возвращает 9
`hour( '0.5555' )`	возвращает 13 (так как 0,5555 = 13:19:55)

inday

Эта функция возвращает значение True, если поле **timestamp** находится в пределах дня, включающего значение, указанное в поле **base_timestamp**.

Синтаксис:

```
InDay (timestamp, base_timestamp, period_no[, day_start])
```

Возвращаемые типы данных: Булево значение

Аргументы:

Аргументы

Аргумент	Описание
timestamp	Дата и время, которые требуется сравнить с **base_timestamp**.
base_ timestamp	Дата и время, использующиеся для оценки метки времени.
period_no	День можно сместить, задав значение в поле **period_no**. **period_no** – целое число, где 0 обозначает день, включающий значение, указанное в поле **base_timestamp**. Отрицательные значения, заданные в поле **period_no**, означают предшествующие дни, положительные – последующие.
day_start	Если необходимо работать с днями, которые начинаются не в полночь, задайте смещение в виде десятичного значения в поле **day_start**, например, 0,125, чтобы день начинался в 3 часа (3 am).

Примеры и результаты:

Примеры написания скриптов

Пример	Результат
`inday ('12/01/2006 12:23:00', '12/01/2006 00:00:00', 0)`	Возвращает True
`inday ('12/01/2006 12:23:00', '13/01/2006 00:00', 0)`	Возвращает False
`inday ('12/01/2006 12:23:00', '12/01/2006 00:00:00', -1)`	Возвращает False
`inday ('11/01/2006 12:23:00', '12/01/2006 00:00:00', -1)`	Возвращает True
`inday ('12/01/2006 12:23:00', '12/01/2006 00:00:00', 0, 0.5)`	Возвращает False
`inday ('12/01/2006 11:23:00', '12/01/2006 00:00:00', 0, 0.5)`	Возвращает True

Пример:

Добавьте образец скрипта в свое приложение и запустите. Чтобы увидеть результаты, добавьте поля, перечисленные в столбце результатов, в лист приложения.

В этом примере проверяется, выпадает ли дата счета на период, начиная с base_timestamp.

```
TempTable:
LOAD RecNo() as InvID, * Inline [
InvTime
28/03/2012
10/12/2012
5/2/2013
31/3/2013
19/5/2013
15/9/2013
```

```
11/12/2013
2/3/2014
14/5/2014
13/6/2014
7/7/2014
4/8/2014
];

InvoiceData:
LOAD *,
InDay(InvTime, '28/03/2012 00:00:00', 0) AS InDayEx
Resident TempTable;
Drop table TempTable;
```

Результирующая таблица содержит исходные даты и столбец с возвращенным значением функции inday().

Результирующая таблица

InvTime	InDayEx
28/03/2012	-1 (True)
10/12/2012	0 (False)
5/2/2013	0 (False)
31/3/2013	0 (False)
19/5/2013	0 (False)
15/9/2013	0 (False)
11/12/2013	0 (False)
2/3/2014	0 (False)
14/5/2014	0 (False)
13/6/2014	0 (False)
7/7/2014	0 (False)
4/8/2014	0 (False)

indaytotime

Эта функция возвращает значение True, если значение **timestamp** находится в пределах части дня, включающей значение, заданное в поле **base_timestamp** до определенной миллисекунды, указанной в поле **base_timestamp**, включительно.

Синтаксис:

```
InDayToTime (timestamp, base_timestamp, period_no[, day_start])
```

Возвращаемые типы данных: Булево значение

Аргументы:

Аргументы

Аргумент	Описание
timestamp	Дата и время, которые требуется сравнить с **base_timestamp**.
base_ timestamp	Дата и время, использующиеся для оценки метки времени.
period_no	День можно сместить, задав значение в поле **period_no**. **period_no** – целое число, где 0 обозначает день, включающий значение, указанное в поле **base_timestamp**. Отрицательные значения, заданные в поле **period_no**, означают предшествующие дни, положительные – последующие.
day_start	(дополнительно) Если необходимо работать с днями, которые начинаются не в полночь, задайте смещение в виде десятичного значения в поле **day_start**, например, 0,125, чтобы день начинался в 3 часа (3 am).

Примеры и результаты:

Примеры написания скриптов

Пример	Результат
`indaytotime ('12/01/2006 12:23:00', '12/01/2006 23:59:00', 0)`	Возвращает True
`indaytotime ('12/01/2006 12:23:00', '12/01/2006 00:00:00', 0)`	Возвращает False
`indaytotime ('11/01/2006 12:23:00', '12/01/2006 23:59:00', -1)`	Возвращает True

Пример:

Добавьте образец скрипта в свое приложение и запустите. Чтобы увидеть результаты, добавьте поля, перечисленные в столбце результатов, в лист приложения.

В этом примере проверяется, выпадает ли метка времени счета на период до 17:00:00 дня, начинающегося с base_timestamp.

```
TempTable:
LOAD RecNo() as InvID, * Inline [
InvTime
28/03/2012
10/12/2012
5/2/2013
31/3/2013
19/5/2013
15/9/2013
11/12/2013
2/3/2014
14/5/2014
13/6/2014
```

```
7/7/2014
4/8/2014
];

InvoiceData:
LOAD *,
InDayToTime(InvTime, '28/03/2012 17:00:00', 0) AS InDayExTT
Resident TempTable;
Drop table TempTable;
```

Результирующая таблица содержит исходные даты и столбец с возвращенным значением функции indaytotime().

Результирующая таблица

InvTime	InDayExTT
28/03/2012	-1 (True)
10/12/2012	0 (False)
5/2/2013	0 (False)
31/3/2013	0 (False)
19/5/2013	0 (False)
15/9/2013	0 (False)
11/12/2013	0 (False)
2/3/2014	0 (False)
14/5/2014	0 (False)
13/6/2014	0 (False)
7/7/2014	0 (False)
4/8/2014	0 (False)

inlunarweek

Эта функция определяет, находится ли значение **timestamp** в пределах лунной недели, включающей значение, указанное в поле **base_date**. Лунные недели в Qlik Sense определяются от 1 января как первого дня недели.

Синтаксис:

```
InLunarWeek (timestamp, base_date, period_no[, first_week_day])
```

Возвращаемые типы данных: Булево значение

Аргументы:

Аргументы

Аргумент	Описание
timestamp	Дата, которую необходимо сравнить со значением, указанным в поле **base_date**.
base_date	Дата, использующаяся для оценки лунной недели.
period_no	Лунную неделю можно сместить, задав значение в поле **period_no**. period_no – целое число, где 0 обозначает лунную неделю, включающую значение, указанное в поле **base_date**. Отрицательные значения, заданные в поле **period_no**, означают предшествующие лунные недели, положительные – последующие.
first_week_day	Смещение, которое может быть больше или меньше нуля. Оно изменяет начало года указанным количеством дней и/или десятичных значений.

Примеры и результаты:

Примеры написания скриптов

Пример	Результат
`inlunarweek ('12/01/2013', '14/01/2013', 0)`	Возвращает True. Поскольку значение timestamp, 12/01/2013 выпадает на период с 08/01/2013 по 14/01/2013.
`inlunarweek ('12/01/2013', '07/01/2013', 0)`	Возвращает False. Поскольку base_date 07/01/2013 находится в пределах лунной недели, определенной как период с 01/01/2013 по 07/01/2013.
`inlunarweek ('12/01/2013', '14/01/2013', -1)`	Возвращает False. Поскольку при указании значения period_no как -1 происходит сдвиг недели на предыдущую неделю с 01/01/2013 по 07/01/2013.
`inlunarweek ('07/01/2013', '14/01/2013', -1)`	Возвращает True. По сравнению с предыдущим примером метка времени находится в периоде с учетом сдвига назад.
`inlunarweek ('11/01/2006', '08/01/2006', 0, 3)`	Возвращает False. Поскольку при указании значения для first_week_day как 3 начало года вычисляется от даты 04/01/2013, и, таким образом, значение base_date выпадает на первую неделю, а значение timestamp выпадает на период с 11/01/2013 по 17/01/2013.

Пример:

Добавьте образец скрипта в свое приложение и запустите. Чтобы увидеть результаты, добавьте поля, перечисленные в столбце результатов, в лист приложения.

В этом примере проверяется, выпадает ли дата счета на неделю, сдвинутую от значения base_date на четыре недели.

```
TempTable:
LOAD RecNo() as InvID, * Inline [
InvDate
28/03/2012
10/12/2012
5/2/2013
31/3/2013
19/5/2013
15/9/2013
11/12/2013
2/3/2014
14/5/2014
13/6/2014
7/7/2014
4/8/2014
];

InvoiceData:
LOAD *,
InLunarWeek(InvDate, '11/01/2013', 4) AS InLWeekPlus4
Resident TempTable;
Drop table TempTable;
```

Результирующая таблица содержит исходные даты и столбец с возвращенным значением функции inlunarweek().

Функция возвращает True для значения InvDate5/2/2013, поскольку значение base_date, 11/01/2013, сдвигается на четыре недели, и, таким образом, выпадает на период с 5/02/2013 по 11/02/2013.

Результирующая таблица

InvDate	InLWeekPlus4
28/03/2012	0 (False)
10/12/2012	0 (False)
5/2/2013	-1 (True)
31/3/2013	0 (False)
19/5/2013	0 (False)
15/9/2013	0 (False)
11/12/2013	0 (False)
2/3/2014	0 (False)
14/5/2014	0 (False)
13/6/2014	0 (False)
7/7/2014	0 (False)
4/8/2014	0 (False)

inlunarweektodate

Эта функция определяет, находится ли значение **timestamp** в пределах части лунной недели до последней миллисекунды, указанной в поле **base_date**, включительно. Лунные недели в Qlik Sense определяются от 1 января как первого дня недели.

Синтаксис:

```
InLunarWeekToDate (timestamp, base_date, period_no [, first_week_day])
```

Возвращаемые типы данных: Булево значение

Аргументы:

Аргументы

Аргумент	Описание
timestamp	Дата, которую необходимо сравнить со значением, указанным в поле **base_date**.
base_date	Дата, использующаяся для оценки лунной недели.
period_no	Лунную неделю можно сместить, задав значение в поле **period_no**. period_no – целое число, где 0 обозначает лунную неделю, включающую значение, указанное в поле **base_date**. Отрицательные значения, заданные в поле **period_no**, означают предшествующие лунные недели, положительные – последующие.
first_week_day	Смещение, которое может быть больше или меньше нуля. Оно изменяет начало года указанным количеством дней и/или десятичных значений.

Примеры и результаты:

Примеры написания скриптов

Пример	Результат
`inlunarweektodate ('12/01/2013', '13/01/2013', 0)`	Возвращает True. Поскольку значение timestamp, 12/01/2013, выпадает на период с 08/01/2013 по 13/01/2013.
`inlunarweektodate ('12/01/2013', '11/01/2013', 0)`	Возвращает False. Поскольку значение timestamp имеет более позднюю дату, чем значение base_date, несмотря на то что обе даты приходятся на одну и ту же лунную неделю до 12/01/2012.
`inlunarweektodate ('12/01/2006', '05/01/2006', 1)`	Возвращает True. При указании значения 1 для period_no происходит сдвиг base_date на одну неделю вперед, таким образом, значение timestamp выпадает на часть лунной недели.

Пример:

Добавьте образец скрипта в свое приложение и запустите. Чтобы увидеть результаты, добавьте поля, перечисленные в столбце результатов, в лист приложения.

В этом примере проверяется, выпадает ли дата счета на часть недели, сдвинутую от значения base_date на четыре недели.

```
TempTable:
LOAD RecNo() as InvID, * Inline [
InvDate
28/03/2012
10/12/2012
5/2/2013
31/3/2013
19/5/2013
15/9/2013
11/12/2013
2/3/2014
14/5/2014
13/6/2014
7/7/2014
4/8/2014
];

InvoiceData:
LOAD *,
InLunarWeekToDate(InvDate, '07/01/2013', 4) AS InLWeek2DPlus4
Resident TempTable;
Drop table TempTable;
```

Результирующая таблица содержит исходные даты и столбец с возвращенным значением функции inlunarweek().

Функция возвращает True для значения InvDate5/2/2013, поскольку значение base_date, 11/01/2013, сдвигается на четыре недели, и, таким образом, выпадает на часть недели с 5/02/2013 по 07/02/2013.

Результирующая таблица

InvDate	InLWeek2DPlus4
28/03/2012	0 (False)
10/12/2012	0 (False)
5/2/2013	-1 (True)
31/3/2013	0 (False)
19/5/2013	0 (False)
15/9/2013	0 (False)
11/12/2013	0 (False)
2/3/2014	0 (False)
14/5/2014	0 (False)
13/6/2014	0 (False)
7/7/2014	0 (False)
4/8/2014	0 (False)

inmonth

Эта функция возвращает значение True, если поле **timestamp** находится в пределах месяца, включающего значение, указанное в поле **base_date**.

Синтаксис:

```
InMonth (timestamp, base_date, period_no[, first_month_of_year])
```

Возвращаемые типы данных: Булево значение

Аргументы:

Аргументы

Аргумент	Описание
timestamp	Дата, которую необходимо сравнить со значением, указанным в поле **base_date**.
base_date	Дата, использующаяся для оценки месяца.
period_no	Месяц можно сместить, задав значение в поле **period_no**. **period_no** – целое число, где 0 обозначает месяц, включающий значение, указанное в поле **base_date**. Отрицательные значения, заданные в поле **period_no**, означают предшествующие месяцы, положительные – последующие.
first_month_of_year	Параметр **first_month_of_year** отключен и зарезервирован для использования в будущем.

Примеры и результаты:

Примеры написания скриптов

Пример	Результат
`inmonth ('25/01/2013', '01/01/2013', 0 )`	Возвращает True
`inmonth('25/01/2013', '01/04/2013', 0)`	Возвращает False
`inmonth ('25/01/2013', '01/01/2013', -1)`	Возвращает False
`inmonth ('25/12/2012', '01/01/2013', -1)`	Возвращает True

Пример:

Добавьте образец скрипта в свое приложение и запустите. Чтобы увидеть результаты, добавьте поля, перечисленные в столбце результатов, в лист приложения.

В этом примере проверяется, выпадает ли дата счета на какой-нибудь день четвертого месяца после месяца в base_date при указании period_no как 4.

```
TempTable:
LOAD RecNo() as InvID, * Inline [
InvDate
28/03/2012
```

```
10/12/2012
5/2/2013
31/3/2013
19/5/2013
15/9/2013
11/12/2013
2/3/2014
14/5/2014
13/6/2014
7/7/2014
4/8/2014
];

InvoiceData:
LOAD *,
InMonth(InvDate, '31/01/2013', 4) AS InMthPlus4
Resident TempTable;
Drop table TempTable;
```

Результирующая таблица содержит исходные даты и столбец с возвращенным значением функции inmonth().

Результирующая таблица

InvDate	InMthPlus4
28/03/2012	0 (False)
10/12/2012	0 (False)
5/2/2013	0 (False)
31/3/2013	0 (False)
19/5/2013	-1 (True)
15/9/2013	0 (False)
11/12/2013	0 (False)
2/3/2014	0 (False)
14/5/2014	0 (False)
13/6/2014	0 (False)
7/7/2014	0 (False)
4/8/2014	0 (False)

inmonths

Эта функция определяет, находится ли метка времени в том же месяце, двухмесячном периоде, квартале, триместре или полугодии, что и базовая дата. Также можно проследить, находится ли эта метка времени в предыдущем или последующем временном периоде.

Синтаксис:

```
InMonths(n_months, timestamp, base_date, period_no [, first_month_of_year])
```

Возвращаемые типы данных: Булево значение

Аргументы:

Аргументы

Аргумент	Описание
n_months	Число месяцев, обозначающее период. Целое число или выражение, определяемое по целому числу, которое должно быть одним из следующих значений: 1 (эквивалентно функции inmonth()), 2 (двухмесячный период), 3 (эквивалентно функции inquarter()), 4 (триместр) или 6 (полугодие).
timestamp	Дата, которую необходимо сравнить со значением, указанным в поле **base_date**.
base_date	Дата, использующаяся для оценки периода.
period_no	Период можно сместить, задав значение в поле **period_no**, целом числе или выражении, определяемом по целому числу, где 0 обозначает период, включающий значение, указанное в поле **base_date**. Отрицательные значения, заданные в поле **period_no**, означают предшествующие периоды, положительные – последующие.
first_month_of_year	Если необходимо работать с годами (финансовыми), которые начинаются не в январе, задайте значение от 2 до 12 в поле **first_month_of_year**.

Примеры и результаты:
В этих примерах используется формат даты **DD/MM/YYYY**. Формат даты указан в операторе **SET DateFormat** в верхней части скрипта загрузки данных. Измените формат в примерах согласно своим пожеланиям.

Примеры написания скриптов

Пример	Результат
`inmonths(4, '25/01/2013', '25/04/2013', 0)`	Возвращает True. Поскольку значение timestamp, 25/01/2013, находится в четырехмесячном периоде с 01/01/2013 по 30/04/2013, в котором находится значение base_date 25/04/2013.
`inmonths(4, '25/05/2013', '25/04/2013', 0)`	Возвращает False. Поскольку 25/05/2013 находится за пределами периода, указанного в предыдущем примере.

Пример	Результат
`inmonths(4, '25/11/2012', '01/02/2013', -1 )`	Возвращает True. Поскольку значение period_no, -1, сдвигает период поиска на один период из четырех месяцев назад (значение n-months), вследствие чего период поиска будет составлять промежуток с 01/09/2012 по 31/12/2012.
`inmonths( 4, '25/05/2006', '01/03/2006', 0, 3)`	Возвращает True. Поскольку для значения first_month_of_year задано значение 3, вследствие чего период поиска будет составлять промежуток с 01/03/2006 по 30/07/2006 вместо промежутка с 01/01/2006 по 30/04/2006.

Пример:

Добавьте образец скрипта в свое приложение и запустите. Чтобы увидеть результаты, добавьте поля, перечисленные в столбце результатов, в лист приложения.

В этом примере проверяется, выпадает ли дата счета в таблице на двухмесячный период, который включает элемент base_date, смещенный вперед на один двухмесячный период (путем указания period_no как 1).

```
TempTable:
LOAD RecNo() as InvID, * Inline [
InvDate
28/03/2012
10/12/2012
5/2/2013
31/3/2013
19/5/2013
15/9/2013
11/12/2013
2/3/2014
14/5/2014
13/6/2014
7/7/2014
4/8/2014
];

InvoiceData:
LOAD *,
InMonths(2, InvDate, '11/02/2013', 1) AS InMthsPlus1
Resident TempTable;
Drop table TempTable;
```

Результирующая таблица содержит исходные даты и столбец с возвращенным значением функции InMonths().

Период поиска составляет промежуток с 01/03/2013 по 30/04/2013, поскольку значение base_date смещено на два месяца вперед от значения в функции (11/02/2013).

Результирующая таблица

InvDate	InMthsPlus1
28/03/2012	0 (False)
10/12/2012	0 (False)
5/2/2013	0 (False)
31/3/2013	-1 (True)
19/5/2013	0 (False)
15/9/2013	0 (False)
11/12/2013	0 (False)
2/3/2014	0 (False)
14/5/2014	0 (False)
13/6/2014	0 (False)
7/7/2014	0 (False)
4/8/2014	0 (False)

inmonthstodate

Эта функция определяет, находится ли метка времени в части месяца, двухмесячного периода, квартала, триместра или полугодия до последней миллисекунды, указанной в поле **base_date**, включительно. Также можно проследить, находится ли метка времени в предыдущем или в последующем временном периоде.

Синтаксис:

```
InMonths (n_months, timestamp, base_date, period_no[, first_month_of_year ])
```

Возвращаемые типы данных: Булево значение

Аргументы:

Аргументы

Аргумент	Описание
n_months	Число месяцев, обозначающее период. Целое число или выражение, определяемое по целому числу, которое должно быть одним из следующих значений: 1 (эквивалентно функции inmonth()), 2 (двухмесячный период), 3 (эквивалентно функции inquarter()), 4 (триместр) или 6 (полугодие).
timestamp	Дата, которую необходимо сравнить со значением, указанным в поле **base_date**.
base_date	Дата, использующаяся для оценки периода.
period_no	Период можно сместить, задав значение в поле **period_no**, целом числе или выражении, определяемом по целому числу, где 0 обозначает период, включающий значение, указанное в поле **base_date**. Отрицательные значения, заданные в поле **period_no**, означают предшествующие периоды, положительные – последующие.
first_month_of_year	Если необходимо работать с годами (финансовыми), которые начинаются не в январе, задайте значение от 2 до 12 в поле **first_month_of_year**.

Примеры и результаты:
В этих примерах используется формат даты **DD/MM/YYYY**. Формат даты указан в операторе **SET DateFormat** в верхней части скрипта загрузки данных. Измените формат в примерах согласно своим пожеланиям.

Примеры написания скриптов

Пример	Результат
`inmonthstodate (4, '25/01/2013', '25/04/2013', 0)`	Возвращает True. Поскольку значение timestamp, 25/01/2013, находится в четырехмесячном периоде с 01/01/2013 по конец 25/04/2013, в котором находится значение base_date 25/04/2013.
`inmonthstodate (4, '26/04/2013', '25/04/2006', 0)`	Возвращает False. Поскольку 26/04/2013 находится за пределами периода, указанного в предыдущем примере.
`inmonthstodate (4, '25/09/2005', '01/02/2006', -1)`	Возвращает True. Поскольку значение period_no, -1, сдвигает период поиска на один период из четырех месяцев назад (значение n-months), вследствие чего период поиска будет составлять промежуток с 01/09/2005 по 01/02/2006.

Пример	Результат
`inmonthstodate (4, '25/04/2006', '01/06/2006', 0, 3)`	Возвращает True. Поскольку для значения first_month_of_year задано значение 3, вследствие чего период поиска будет составлять промежуток с 01/03/2006 по 01/06/2006 вместо промежутка с 01/05/2006 по 01/06/2006.

Пример:

Добавьте образец скрипта в свое приложение и запустите. Чтобы увидеть результаты, добавьте поля, перечисленные в столбце результатов, в лист приложения.

В этом примере проверяется, выпадает ли дата счета в таблице на часть двухмесячного периода, который включает элемент base_date, смещенный вперед на четыре двухмесячных периода (путем указания period_no как 4).

```
TempTable:
LOAD RecNo() as InvID, * Inline [
InvDate
28/03/2012
10/12/2012
5/2/2013
31/3/2013
19/5/2013
15/9/2013
11/12/2013
2/3/2014
14/5/2014
13/6/2014
7/7/2014
4/8/2014
];

InvoiceData:
LOAD *,
InMonthsToDate(2, InvDate, '15/02/2013', 4) AS InMths2DPlus4
Resident TempTable;
Drop table TempTable;
```

Результирующая таблица содержит исходные даты и столбец с возвращенным значением функции InMonths().

Период поиска составляет промежуток с 01/09/2013 по 15/10/2013, поскольку значение base_date смещено на восемь месяцев вперед от значения в функции (15/02/2013).

Результирующая таблица

InvDate	InMths2DPlus4
28/03/2012	0 (False)
10/12/2012	0 (False)

InvDate	InMths2DPlus4
5/2/2013	0 (False)
31/3/2013	0 (False)
19/5/2013	0 (False)
15/9/2013	-1 (True)
11/12/2013	0 (False)
2/3/2014	0 (False)
14/5/2014	0 (False)
13/6/2014	0 (False)
7/7/2014	0 (False)
4/8/2014	0 (False)

inmonthtodate

Возвращает значение True, если значение **date** находится в пределах части месяца, включающей значение, заданное в поле **basedate** до последней миллисекунды, указанной в поле **basedate**, включительно.

Синтаксис:

```
InMonthToDate (timestamp, base_date, period_no)
```

Возвращаемые типы данных: Булево значение

Аргументы:

Аргументы

Аргумент	Описание
timestamp	Дата, которую необходимо сравнить со значением, указанным в поле **base_date**.
base_date	Дата, использующаяся для оценки месяца.
period_no	Месяц можно сместить, задав значение в поле **period_no**. **period_no** – целое число, где 0 обозначает месяц, включающий значение, указанное в поле **base_date**. Отрицательные значения, заданные в поле **period_no**, означают предшествующие месяцы, положительные – последующие.

Примеры и результаты:

Примеры написания скриптов

Пример	Результат
`inmonthtodate ('25/01/2013', '25/01/2013', 0)`	Возвращает True
`inmonthtodate ('25/01/2013', '24/01/2013', 0)`	Возвращает False
`inmonthtodate ('25/01/2013', '28/02/2013', -1)`	Возвращает True

Пример:

Добавьте образец скрипта в свое приложение и запустите. Чтобы увидеть результаты, добавьте поля, перечисленные в столбце результатов, в лист приложения.

При указании значения 4 для period_no в этом примере проверяется, выпадает ли дата счета на четвертый месяц после месяца в base_date, но до конца дня, указанного в base_date.

```
TempTable:
LOAD RecNo() as InvID, * Inline [
InvDate
28/03/2012
10/12/2012
5/2/2013
31/3/2013
19/5/2013
15/9/2013
11/12/2013
2/3/2014
14/5/2014
13/6/2014
7/7/2014
4/8/2014
];

InvoiceData:
LOAD *,
InMonthToDate(InvDate, '31/01/2013', 4) AS InMthPlus42D
Resident TempTable;
Drop table TempTable;
```

Результирующая таблица содержит исходные даты и столбец с возвращенным значением функции inmonthtodate().

Результирующая таблица

InvDate	InMthPlus42D
28/03/2012	0 (False)
10/12/2012	0 (False)
5/2/2013	0 (False)
31/3/2013	0 (False)

InvDate	InMthPlus42D
19/5/2013	-1 (True)
15/9/2013	0 (False)
11/12/2013	0 (False)
2/3/2014	0 (False)
14/5/2014	0 (False)
13/6/2014	0 (False)
7/7/2014	0 (False)
4/8/2014	0 (False)

inquarter

Эта функция возвращает значение True, если поле **timestamp** находится в пределах квартала, включающего значение, указанное в поле **base_date**.

Синтаксис:

```
InQuarter (timestamp, base_date, period_no[, first_month_of_year])
```

Возвращаемые типы данных: Булево значение

Аргументы:

Аргументы

Аргумент	Описание
timestamp	Дата, которую необходимо сравнить со значением, указанным в поле **base_date**.
base_date	Дата, использующаяся для оценки квартала.
period_no	Квартал можно сместить, задав значение в поле **period_no**. **period_no** – целое число, где 0 обозначает квартал, включающий значение, указанное в поле **base_date**. Отрицательные значения, заданные в поле **period_no**, означают предшествующие кварталы, положительные – последующие.
first_month_of_year	Если необходимо работать с годами (финансовыми), которые начинаются не в январе, задайте значение от 2 до 12 в поле **first_month_of_year**.

Примеры и результаты:

Примеры написания скриптов

Пример	Результат
`inquarter ('25/01/2013', '01/01/2013', 0)`	Возвращает True
`inquarter ('25/01/2013', '01/04/2013', 0)`	Возвращает False

Пример	Результат
`inquarter ('25/01/2013', '01/01/2013', -1)`	Возвращает False
`inquarter ('25/12/2012', '01/01/2013', -1)`	Возвращает True
`inquarter ('25/01/2013', '01/03/2013', 0, 3)`	Возвращает False
`inquarter ('25/03/2013', '01/03/2013', 0, 3)`	Возвращает True

Пример:

Добавьте образец скрипта в свое приложение и запустите. Чтобы увидеть результаты, добавьте поля, перечисленные в столбце результатов, в лист приложения.

В этом примере проверяется, выпадает ли дата счета на четвертый квартал финансового года, указанного путем указания значения 4 для элемента first_month_of_year, при этом значение для base_date равно 31/01/2013.

```
TempTable:
LOAD RecNo() as InvID, * Inline [
InvDate
28/03/2012
10/12/2012
5/2/2013
31/3/2013
19/5/2013
15/9/2013
11/12/2013
2/3/2014
14/5/2014
13/6/2014
7/7/2014
4/8/2014
];

InvoiceData:
LOAD *,
InQuarter(InvDate, '31/01/2013', 0, 4) AS Qtr4FinYr1213
Resident TempTable;
Drop table TempTable;
```

Результирующая таблица содержит исходные даты и столбец с возвращенным значением функции inquarter().

Результирующая таблица

InvDate	Qtr4Fin1213
28/03/2012	0 (False)
10/12/2012	0 (False)
5/2/2013	-1 (True)

InvDate	Qtr4Fin1213
31/3/2013	-1 (True)
19/5/2013	0 (False)
15/9/2013	0 (False)
11/12/2013	0 (False)
2/3/2014	0 (False)
14/5/2014	0 (False)
13/6/2014	0 (False)
7/7/2014	0 (False)
4/8/2014	0 (False)

inquartertodate

Эта функция возвращает значение True, если значение **timestamp** находится в пределах части квартала, включающей значение, заданное в поле **base_date** до последней миллисекунды, указанной в поле **base_date**, включительно.

Синтаксис:

```
InQuarterToDate (timestamp, base_date, period_no [, first_month_of_year])
```

Возвращаемые типы данных: Булево значение

Аргументы:

Аргументы

Аргумент	Описание
timestamp	Дата, которую необходимо сравнить со значением, указанным в поле **base_date**.
base_date	Дата, использующаяся для оценки квартала.
period_no	Квартал можно сместить, задав значение в поле **period_no**. **period_no** – целое число, где 0 обозначает квартал, включающий значение, указанное в поле **base_date**. Отрицательные значения, заданные в поле **period_no**, означают предшествующие кварталы, положительные – последующие.
first_month_of_year	Если необходимо работать с годами (финансовыми), которые начинаются не в январе, задайте значение от 2 до 12 в поле **first_month_of_year**.

Примеры и результаты:

Примеры написания скриптов

Пример	Результат
`inquartertodate ('25/01/2013', '25/01/2013', 0)`	Возвращает True
`inquartertodate ( 25/01/2013', '24/01/2013', 0)`	Возвращает False
`inquartertodate ('25/01/2012', '01/02/2013', -1)`	Возвращает True

Пример:

Добавьте образец скрипта в свое приложение и запустите. Чтобы увидеть результаты, добавьте поля, перечисленные в столбце результатов, в лист приложения.

В этом примере проверяется, выпадает ли дата счета на финансовый год, указанный путем указания значения 4 для элемента first_month_of_year, и на четвертую четверть до окончания 28/02/2013.

```
TempTable:
LOAD RecNo() as InvID, * Inline [
InvDate
28/03/2012
10/12/2012
5/2/2013
31/3/2013
19/5/2013
15/9/2013
11/12/2013
2/3/2014
14/5/2014
13/6/2014
7/7/2014
4/8/2014
];

InvoiceData:
LOAD *,
InQuarterToDate(InvDate, '28/02/2013', 0, 4) AS Qtr42Date
Resident TempTable;
Drop table TempTable;
```

Результирующая таблица содержит исходные даты и столбец с возвращенным значением функции inquartertodate().

Результирующая таблица

InvDate	Qtr42Date
28/03/2012	0 (False)
10/12/2012	0 (False)
5/2/2013	-1 (True)

InvDate	Qtr42Date
31/3/2013	0 (False)
19/5/2013	0 (False)
15/9/2013	0 (False)
11/12/2013	0 (False)
2/3/2014	0 (False)
14/5/2014	0 (False)
13/6/2014	0 (False)
7/7/2014	0 (False)
4/8/2014	0 (False)

inweek

Эта функция возвращает значение True, если поле **timestamp** находится в пределах недели, включающей значение, указанное в поле **base_date**.

Синтаксис:

```
InWeek (timestamp, base_date, period_no[, first_week_day])
```

Возвращаемые типы данных: Булево значение

Аргументы:

Аргументы

Аргумент	Описание
timestamp	Дата, которую необходимо сравнить со значением, указанным в поле **base_date**.
base_date	Дата, использующаяся для оценки недели.
period_no	Неделю можно сместить, задав значение в поле **period_no**. **period_no** – целое число, где 0 обозначает неделю, включающую значение, указанное в поле **base_date**. Отрицательные значения, заданные в поле **period_no**, означают предшествующие недели, положительные – последующие.
first_week_day	По умолчанию первым днем недели является понедельник, началом которого является полночь в ночь с воскресенья на понедельник. Чтобы задать другой день в качестве начала недели, укажите смещение в поле **first_week_day**. Это может быть целое число дней и/или десятичное значение.

Примеры и результаты:

Примеры написания скриптов

Пример	Результат
`inweek ('12/01/2006', '14/01/2006', 0)`	Возвращает True
`inweek ('12/01/2006', '20/01/2006', 0 )`	Возвращает False
`inweek ('12/01/2006', '14/01/2006', -1 )`	Возвращает False
`inweek ('07/01/2006', '14/01/2006', -1)`	Возвращает True
`inweek ('12/01/2006', '09/01/2006', 0, 3)`	Возвращает False Для элемента first_week_day указано значение 3 (четверг), в результате чего элемент 12/01/2006 становится первым днем недели после недели с элементом 09/01/2006.

Пример:

Добавьте образец скрипта в свое приложение и запустите. Чтобы увидеть результаты, добавьте поля, перечисленные в столбце результатов, в лист приложения.

В этом примере проверяется, выпадает ли дата счета на какой-нибудь день четвертой недели после недели в base_date при указании значения 4 для элемента period_no.

```
TempTable:
LOAD RecNo() as InvID, * Inline [
InvDate
28/03/2012
10/12/2012
5/2/2013
31/3/2013
19/5/2013
15/9/2013
11/12/2013
2/3/2014
14/5/2014
13/6/2014
7/7/2014
4/8/2014
];

InvoiceData:
LOAD *,
InWeek(InvDate, '11/01/2013', 4) AS InWeekPlus4
Resident TempTable;
Drop table TempTable;
```

Результирующая таблица содержит исходные даты и столбец с возвращенным значением функции inweek().

InvDate5/2/2013 выпадает на неделю через четыре недели после base_date: 11/1/2013.

Результирующая таблица

InvDate	InWeekPlus4
28/03/2012	0 (False)
10/12/2012	0 (False)
5/2/2013	-1 (True)
31/3/2013	0 (False)
19/5/2013	0 (False)
15/9/2013	0 (False)
11/12/2013	0 (False)
2/3/2014	0 (False)
14/5/2014	0 (False)
13/6/2014	0 (False)
7/7/2014	0 (False)
4/8/2014	0 (False)

inweektodate

Эта функция возвращает значение True, если значение **timestamp** находится в пределах части недели, включающей значение, заданное в поле **base_date** до последней миллисекунды, указанной в поле **base_date**, включительно.

Синтаксис:

```
InWeekToDate (timestamp, base_date, period_no [, first_week_day])
```

Возвращаемые типы данных: Булево значение

Аргументы:

Аргументы

Аргумент	Описание
timestamp	Дата, которую необходимо сравнить со значением, указанным в поле **base_date**.
base_date	Дата, использующаяся для оценки недели.

Аргумент	Описание
period_no	Неделю можно сместить, задав значение в поле **period_no**. **period_no** – целое число, где 0 обозначает неделю, включающую значение, указанное в поле **base_date**. Отрицательные значения, заданные в поле **period_no**, означают предшествующие недели, положительные – последующие.
first_week_day	По умолчанию первым днем недели является понедельник, началом которого является полночь в ночь с воскресенья на понедельник. Чтобы задать другой день в качестве начала недели, укажите смещение в поле **first_week_day**. Это может быть целое число дней и/или десятичное значение.

Примеры и результаты:

Примеры написания скриптов

Пример	Результат
`inweektodate ('12/01/2006', '12/01/2006', 0)`	Возвращает True
`inweektodate ('12/01/2006', '11/01/2006', 0)`	Возвращает False
`inweektodate ('12/01/2006', '18/01/2006', -1)`	Возвращает False Поскольку для элемента period_no указано значение -1, дата вступления в силу timestamp измеряется на основе 11/01/2006.
`inweektodate ( '11/01/2006', '12/01/2006', 0, 3 )`	Возвращает False Для элемента first_week_day указано значение 3 (четверг), в результате чего элемент 12/01/2006 становится первым днем недели после недели с элементом 12/01/2006.

Пример:

Добавьте образец скрипта в свое приложение и запустите. Чтобы увидеть результаты, добавьте поля, перечисленные в столбце результатов, в лист приложения.

В этом примере проверяется, выпадает ли дата счета на четвертую неделю после недели в base_date путем указания значения 4 для элемента period_no, но до значения base_date.

```
TempTable:
LOAD RecNo() as InvID, * Inline [
InvDate
28/03/2012
10/12/2012
5/2/2013
31/3/2013
19/5/2013
15/9/2013
11/12/2013
2/3/2014
14/5/2014
```

```
13/6/2014
7/7/2014
4/8/2014
];

InvoiceData:
LOAD *,
InWeekToDate(InvDate, '11/01/2013', 4) AS InWeek2DPlus4
Resident TempTable;
Drop table TempTable;
```

Результирующая таблица содержит исходные даты и столбец с возвращенным значением функции inweek().

Результирующая таблица

InvDate	InWeek2DPlus4
28/03/2012	0 (False)
10/12/2012	0 (False)
5/2/2013	-1 (True)
31/3/2013	0 (False)
19/5/2013	0 (False)
15/9/2013	0 (False)
11/12/2013	0 (False)
2/3/2014	0 (False)
14/5/2014	0 (False)
13/6/2014	0 (False)
7/7/2014	0 (False)
4/8/2014	0 (False)

inyear

Эта функция возвращает значение True, если поле **timestamp** находится в пределах года, включающего значение, указанное в поле **base_date**.

Синтаксис:

```
InYear (timestamp, base_date, period_no [, first_month_of_year])
```

Возвращаемые типы данных: Булево значение

Аргументы:

Аргументы

Аргумент	Описание
timestamp	Дата, которую необходимо сравнить со значением, указанным в поле **base_date**.
base_date	Дата, использующаяся для оценки года.
period_no	Год можно сместить, задав значение в поле **period_no**. **period_no** – целое число, где 0 обозначает год, включающий значение, указанное в поле **base_date**. Отрицательные значения, заданные в поле **period_no**, означают предшествующие годы, положительные – последующие.
first_month_of_year	Если необходимо работать с годами (финансовыми), которые начинаются не в январе, задайте значение от 2 до 12 в поле **first_month_of_year**.

Примеры и результаты:
В этих примерах используется формат даты **DD/MM/YYYY**. Формат даты указан в операторе **SET DateFormat** в верхней части скрипта загрузки данных. Измените формат в примерах согласно своим пожеланиям.

Примеры написания скриптов

Пример	Результат
`inyear ('25/01/2013', '01/01/2013', 0 )`	Возвращает True
`inyear ('25/01/2012', '01/01/2013', 0)`	Возвращает False
`inyear ('25/01/2013', '01/01/2013', -1)`	Возвращает False
`inyear ('25/01/2012', '01/01/2013', -1 )`	Возвращает True
`inyear ('25/01/2013', '01/01/2013', 0, 3)`	Возвращает True Значение base_date и first_month_of_year указывает, что значение timestamp должно выпадать на период с 01/03/2012 по 28/02/2013
`inyear ('25/03/2013', '01/07/2013', 0, 3 )`	Возвращает True

Пример:

Добавьте образец скрипта в свое приложение и запустите. Чтобы увидеть результаты, добавьте поля, перечисленные в столбце результатов, в лист приложения.

В этом примере проверяется, выпадает ли дата счета на финансовый год, указанный путем указания значения 4 для элемента first_month_of_year, при этом значение для base_date составляет промежуток с 1/4/2012 по 31/03/2013.

```
TempTable:
LOAD RecNo() as InvID, * Inline [
InvDate
28/03/2012
10/12/2012
5/2/2013
31/3/2013
19/5/2013
15/9/2013
11/12/2013
2/3/2014
14/5/2014
13/6/2014
7/7/2014
4/8/2014
];
```

Проверьте, выпадает ли значение InvDate на финансовый год в промежутке с 1/04/2012 по 31/03/2013:

```
InvoiceData:
LOAD *,
InYear(InvDate, '31/01/2013', 0, 4) AS FinYr1213
Resident TempTable;
Drop table TempTable;
```

Результирующая таблица содержит исходные даты и столбец с возвращенным значением функции inyear().

Результирующая таблица

InvDate	FinYr1213
28/03/2012	0 (False)
10/12/2012	-1 (True)
5/2/2013	-1 (True)
31/3/2013	-1 (True)
19/5/2013	0 (False)
15/9/2013	0 (False)
11/12/2013	0 (False)
2/3/2014	0 (False)
14/5/2014	0 (False)
13/6/2014	0 (False)
7/7/2014	0 (False)
4/8/2014	0 (False)

inyeartodate

Эта функция возвращает значение True, если значение **timestamp** находится в пределах части года, включающей значение, заданное в поле **base_date** до последней миллисекунды, указанной в поле **base_date**, включительно.

Синтаксис:

```
InYearToDate (timestamp, base_date, period_no[, first_month_of_year])
```

Возвращаемые типы данных: Булево значение

Аргументы:

Аргументы

Аргумент	Описание
timestamp	Дата, которую необходимо сравнить со значением, указанным в поле **base_date**.
base_date	Дата, использующаяся для оценки года.
period_no	Год можно сместить, задав значение в поле **period_no**. **period_no** – целое число, где 0 обозначает год, включающий значение, указанное в поле **base_date**. Отрицательные значения, заданные в поле **period_no**, означают предшествующие годы, положительные – последующие.
first_month_of_year	Если необходимо работать с годами (финансовыми), которые начинаются не в январе, задайте значение от 2 до 12 в поле **first_month_of_year**.

Примеры и результаты:

Примеры написания скриптов

Пример	Результат
`inyeartodate ('2013/01/25', '2013/02/01', 0)`	Возвращает True
`inyeartodate ('2012/01/25', '2013/01/01', 0)`	Возвращает False
`inyeartodate ('2012/01/25', '2013/02/01', -1)`	Возвращает True
`inyeartodate ('2012/11/25', '2013/01/31', 0, 4)`	Возвращает True Значение timestamp выпадает на финансовый год, начинающийся в четвертом месяце, и до значения base_date.

Пример	Результат
`inyeartodate ( '2013/3/31', '2013/01/31', 0, 4 )`	Возвращает False По сравнению с предыдущим примером, значение timestamp все еще находится в финансовом году, но после значения base_date, таким образом, оно находится за пределами части года.

Пример:

Добавьте образец скрипта в свое приложение и запустите. Чтобы увидеть результаты, добавьте поля, перечисленные в столбце результатов, в лист приложения.

В этом примере проверяется, выпадает ли дата счета на финансовый год, указанный путем указания значения 4 для элемента first_month_of_year, и на часть года до окончания 31/01/2013.

```
TempTable:
LOAD RecNo() as InvID, * Inline [
InvDate
28/03/2012
10/12/2012
5/2/2013
31/3/2013
19/5/2013
15/9/2013
11/12/2013
2/3/2014
14/5/2014
13/6/2014
7/7/2014
4/8/2014
];

InvoiceData:
LOAD *,
InYearToDate(InvDate, '31/01/2013', 0, 4) AS FinYr2Date
Resident TempTable;
Drop table TempTable;
```

Результирующая таблица содержит исходные даты и столбец с возвращенным значением функции inyeartodate().

Результирующая таблица

InvDate	FinYr2Date
28/03/2012	0 (False)
10/12/2012	-1 (True)
5/2/2013	0 (False)
31/3/2013	0 (False)
19/5/2013	0 (False)

InvDate	FinYr2Date
15/9/2013	0 (False)
11/12/2013	0 (False)
2/3/2014	0 (False)
14/5/2014	0 (False)
13/6/2014	0 (False)
7/7/2014	0 (False)
4/8/2014	0 (False)

lastworkdate

Функция **lastworkdate** возвращает самую раннюю дату окончания для достижения указанного числа рабочих дней **no_of_workdays** (понедельник-пятница) с начальной датой **start_date** и с учетом выходных, которые можно дополнительно задать в поле **holiday**. Поля **start_date** и **holiday** должны быть действительными датами или метками времени.

Синтаксис:

```
lastworkdate(start_date, no_of_workdays {, holiday})
```

Возвращаемые типы данных: двойное значение

Аргументы:

Аргументы

Аргумент	Описание
start_date	Начальная дата для вычисления.
no_of_ workdays	Количество рабочих дней, которое должно быть получено.
holiday	Периоды выходных дней для исключения из рабочих дней. Период выходных дней указан как дата начала и дата окончания, разделенные запятыми. **Пример:** `'25/12/2013', '26/12/2013'` Можно указать несколько периодов выходных дней, разделенных запятыми. **Пример:** `'25/12/2013', '26/12/2013', '31/12/2013', '01/01/2014'`

Примеры и результаты:
В этих примерах используется формат даты **DD/MM/YYYY**. Формат даты указан в операторе **SET DateFormat** в верхней части скрипта загрузки данных. Измените формат в примерах согласно своим пожеланиям.

Примеры написания скриптов

Пример	Результат
`lastworkdate ('19/12/2014', 9)`	Возвращает «31/12/2014»
`lastworkdate ('19/12/2014', 9, '2014-12-25', '2014-12-26')`	Возвращает «02/01/2015», поскольку учитывается двухдневный период выходных.

Пример:

Добавьте образец скрипта в свое приложение и запустите. Чтобы увидеть результаты, добавьте поля, перечисленные в столбце результатов, в лист приложения.

```
ProjectTable:
LOAD *, recno() as InvID, INLINE [
StartDate
28/03/2014
10/12/2014
5/2/2015
31/3/2015
19/5/2015
15/9/2015
] ;
NrDays:
Load *,
LastWorkDate(StartDate,120) As EndDate
Resident ProjectTable;
Drop table ProjectTable;
```

Результирующая таблица показывает возвращенные значения функции LastWorkDate для каждой записи в таблице.

Результирующая таблица

InvID	StartDate	EndDate
1	28/03/2014	11/09/2014
2	10/12/2014	26/05/2015
3	5/2/2015	27/07/2015
4	31/3/2015	14/09/2015
5	19/5/2015	02/11/2015
6	15/9/2015	29/02/2016

localtime

Эта функция возвращает метку текущего времени по системным часам для указанного часового пояса.

Синтаксис:

```
LocalTime([timezone [, ignoreDST ]])
```

Возвращаемые типы данных: двойное значение

Аргументы:

Аргументы

Аргумент	Описание
timezone	Параметр **timezone** задается как строка, содержащая любое географическое название, указанное в разделе **Часовой пояс** в расположении **Панель управления Windows** для поля **Date and Time** или в виде строки в формате «GMT+чч:мм». Если часовой пояс не задан, будет возвращено местное время.
ignoreDST	Если элемент **ignoreDST** равен -1 (True), то переход на летнее время будет игнорироваться.

Примеры и результаты:

Примеры ниже основаны на функции, вызываемой 2014-10-22 в 12:54:47 по местному времени, часовой пояс местного времени GMT+01:00.

Примеры написания скриптов

Пример	Результат
`localtime ()`	Возвращает местное время в виде 2014-10-22 12:54:47.
`localtime ('London')`	Возвращает местное время в Лондоне виде 2014-10-22 11:54:47.
`localtime ('GMT+02:00')`	Возвращает местное время в часовом поясе GMT+02:00, 2014-10-22 13:54:47.
`localtime ('Paris','-1')`	Возвращает местное время в Париже без учета перехода на летнее время 2014-10-22 11:54:47.

lunarweekend

Эта функция возвращает значение, соответствующее метке времени последней миллисекунды лунной недели, содержащей значение, указанное в поле **date**. Лунные недели в Qlik Sense определяются от 1 января как первого дня недели.

Синтаксис:

```
LunarweekEnd(date[, period_no[, first_week_day]])
```

Возвращаемые типы данных: двойное значение

Аргументы:

Аргументы

Аргумент	Описание
date	Дата для вычисления.
period_no	period_no является целым числом или выражением, определяемым по целому числу, где значение 0 означает лунную неделю, содержащую значение, указанное в поле date. Отрицательные значения, заданные в поле period_no, означают предшествующие лунные недели, положительные – последующие.
first_week_day	Смещение, которое может быть больше или меньше нуля. Оно изменяет начало года указанным количеством дней и/или десятичных значений.

Примеры и результаты:
В этих примерах используется формат даты DD/MM/YYYY. Формат даты указан в операторе SET DateFormat в верхней части скрипта загрузки данных. Измените формат в примерах согласно своим пожеланиям.

Примеры написания скриптов

Пример	Результат
`lunarweekend('12/01/2013')`	Возвращает 14/01/2013 23:59:59.
`lunarweekend('12/01/2013', -1)`	Возвращает 7/01/2013 23:59:59.
`lunarweekend('12/01/2013', 0, 1)`	Возвращает 15/01/2013 23:59:59.

Пример:

Добавьте образец скрипта в свое приложение и запустите. Чтобы увидеть результаты, добавьте поля, перечисленные в столбце результатов, в лист приложения.

В этом примере выполняется обнаружение последнего дня лунной недели каждой даты счета в таблице, где значение date смещается на одну неделю путем указания для элемента period_no значения 1.

```
TempTable:
LOAD RecNo() as InvID, * Inline [
InvDate
28/03/2012
10/12/2012
5/2/2013
31/3/2013
19/5/2013
```

```
15/9/2013
11/12/2013
2/3/2014
14/5/2014
13/6/2014
7/7/2014
4/8/2014
];

InvoiceData:
LOAD *,
LunarWeekEnd(InvDate, 1) AS LWkEnd
Resident TempTable;
Drop table TempTable;
```

Результирующая таблица содержит исходные даты и столбец с возвращенным значением функции lunarweekend(). Чтобы отобразить полную метку времени, укажите форматирование на панели свойств.

Результирующая таблица

InvDate	LWkEnd
28/03/2012	07/04/2012
10/12/2012	22/12/2012
5/2/2013	18/02/2013
31/3/2013	08/04/2013
19/5/2013	27/05/2013
15/9/2013	23/09/2013
11/12/2013	23/12/2013
2/3/2014	11/03/2014
14/5/2014	27/05/2014
13/6/2014	24/06/2014
7/7/2014	15/07/2014
4/8/2014	12/08/2014

lunarweekname

Эта функция возвращает значение года и номер лунной недели, соответствующие метке времени первой миллисекунды первого дня лунной недели, содержащего значение, указанное в поле **date**. Лунные недели в Qlik Sense определяются от 1 января как первого дня недели.

Синтаксис:

```
LunarWeekName(date [, period_no[, first_week_day]])
```

Возвращаемые типы данных: двойное значение

Аргументы:

Аргументы

Аргумент	Описание
date	Дата для вычисления.
period_no	**period_no** является целым числом или выражением, определяемым по целому числу, где значение 0 означает лунную неделю, содержащую значение, указанное в поле **date**. Отрицательные значения, заданные в поле **period_no**, означают предшествующие лунные недели, положительные – последующие.
first_week_day	Смещение, которое может быть больше или меньше нуля. Оно изменяет начало года указанным количеством дней и/или десятичных значений.

Примеры и результаты:

Примеры написания скриптов

Пример	Результат
`lunarweekname('12/01/2013')`	Возвращает 2006/02.
`lunarweekname('12/01/2013', -1)`	Возвращает 2006/01.
`lunarweekname('12/01/2013', 0, 1)`	Возвращает 2006/02.

Пример:

Добавьте образец скрипта в свое приложение и запустите. Чтобы увидеть результаты, добавьте поля, перечисленные в столбце результатов, в лист приложения.

В этом примере для каждой даты счета в таблице имя лунной недели создается из года, в котором находится эта неделя, и связанного с ней номера недели, смещенного на одну неделю путем указания для элемента period_no значения 1.

```
TempTable:
LOAD RecNo() as InvID, * Inline [
InvDate
28/03/2012
10/12/2012
5/2/2013
31/3/2013
19/5/2013
15/9/2013
11/12/2013
2/3/2014
14/5/2014
13/6/2014
7/7/2014
4/8/2014
];
```

```
InvoiceData:
LOAD *,
LunarWeekName(InvDate, 1) AS LWkName
Resident TempTable;
Drop table TempTable;
```

Результирующая таблица содержит исходные даты и столбец с возвращенным значением функции lunarweekname(). Чтобы отобразить полную метку времени, укажите форматирование на панели свойств.

Результирующая таблица

InvDate	LWkName
28/03/2012	2012/14
10/12/2012	2012/51
5/2/2013	2013/07
31/3/2013	2013/14
19/5/2013	2013/21
15/9/2013	2013/38
11/12/2013	2013/51
2/3/2014	2014/10
14/5/2014	2014/21
13/6/2014	2014/25
7/7/2014	2014/28
4/8/2014	2014/32

lunarweekstart

Эта функция возвращает значение, соответствующее метке времени первой миллисекунды лунной недели, содержащей значение, указанное в поле **date**. Лунные недели в Qlik Sense определяются от 1 января как первого дня недели.

Синтаксис:

```
LunarweekStart(date[, period_no[, first_week_day]])
```

Возвращаемые типы данных: двойное значение

Аргументы:

Аргументы

Аргумент	Описание
date	Дата для вычисления.
period_no	**period_no** является целым числом или выражением, определяемым по целому числу, где значение 0 означает лунную неделю, содержащую значение, указанное в поле **date**. Отрицательные значения, заданные в поле **period_no**, означают предшествующие лунные недели, положительные – последующие.
first_week_day	Смещение, которое может быть больше или меньше нуля. Оно изменяет начало года указанным количеством дней и/или десятичных значений.

Примеры и результаты:
В этих примерах используется формат даты **DD/MM/YYYY**. Формат даты указан в операторе **SET DateFormat** в верхней части скрипта загрузки данных. Измените формат в примерах согласно своим пожеланиям.

Примеры написания скриптов

Пример	Результат
`lunarweekstart ('12/01/2013')`	Возвращает 08/01/2013.
`lunarweekstart ('12/01/2013', -1)`	Возвращает 01/01/2013.
`lunarweekstart ('12/01/2013', 0, 1 )`	Возвращает 09/01/2013. Поскольку смещение, определенное значением 1 для first_week_day, означает, что начало года изменяется на 02/01/2013.

Пример:

Добавьте образец скрипта в свое приложение и запустите. Чтобы увидеть результаты, добавьте поля, перечисленные в столбце результатов, в лист приложения.

В этом примере выполняется обнаружение первого дня лунной недели каждой даты счета в таблице, где значение date смещается на одну неделю путем указания для элемента period_no значения 1.

```
TempTable:
LOAD RecNo() as InvID, * Inline [
InvDate
28/03/2012
10/12/2012
5/2/2013
31/3/2013
```

```
19/5/2013
15/9/2013
11/12/2013
2/3/2014
14/5/2014
13/6/2014
7/7/2014
4/8/2014
];

InvoiceData:
LOAD *,
LunarWeekStart(InvDate, 1) AS LWkStart
Resident TempTable;
Drop table TempTable;
```

Результирующая таблица содержит исходные даты и столбец с возвращенным значением функции lunarweekstart(). Чтобы отобразить полную метку времени, укажите форматирование на панели свойств.

Результирующая таблица

InvDate	LWkStart
28/03/2012	01/04/2012
10/12/2012	16/12/2012
5/2/2013	12/02/2013
31/3/2013	02/04/2013
19/5/2013	21/05/2013
15/9/2013	17/09/2013
11/12/2013	17/12/2013
2/3/2014	05/03/2014
14/5/2014	21/05/2014
13/6/2014	18/06/2014
7/7/2014	09/07/2014
4/8/2014	06/08/2014

makedate

Эта функция возвращает дату, рассчитанную в формате год **YYYY**, месяц **MM** и день **DD**.

Синтаксис:

```
MakeDate(YYYY [ , MM [ , DD ] ])
```

Возвращаемые типы данных: двойное значение

Аргументы:

Аргументы

Аргумент	Описание
YYYY	Год – целое число.
MM	Месяц – целое число. Если месяц не задан, используется 1 (январь).
DD	День – целое число. Если день не задан, используется 1 (1-е число).

Пример: Выражение диаграммы

Примеры выражений диаграммы

Пример	Результат
`makedate(2012)`	возвращает 2012-01-01
`makedate(12)`	возвращает 0012-01-01
`makedate(2012,12)`	возвращает 2012-12-01
`makedate(2012,2,14)`	возвращает 2012-02-14

Пример: Скрипт загрузки
Функцию *makedate* можно использовать в скрипте загрузки для объединения данных из разных полей в новое поле данных. В приведенном ниже примере данные года, месяца и дня из полей transaction_year, transaction_month и transaction_day объединены в новое поле с именем Transaction Date.

В **Редакторе загрузки данных** создайте новый раздел, добавьте образец скрипта и запустите его. Затем добавьте на лист приложения как минимум поля, указанные в столбце с результатами, чтобы увидеть результаты.

Скрипт загрузки

```
SET DateFormat='DD/MM/YYYY';
SET TimestampFormat='DD/MM/YYYY h:mm:ss[.fff] TT';
SET FirstWeekDay=0;
SET BrokenWeeks=1;
SET ReferenceDay=0;
SET DayNames='Mon;Tue;Wed;Thu;Fri;Sat;Sun';
SET LongDayNames='Monday;Tuesday;Wednesday;Thursday;Friday;Saturday;Sunday';

Transactions:
Load
*,
MakeDate(transaction_year, transaction_month, transaction_day) as "Transaction Date",
;
```

```
Load * Inline [
transaction_id, transaction_year, transaction_month, transaction_day, transaction_amount,
transaction_quantity, discount, customer_id, size, color_code
3750, 2018, 08, 30, 12423.56, 23, 0,2038593, L, Red
3751, 2018, 09, 07, 5356.31, 6, 0.1, 203521, m, orange
3752, 2018, 09, 16, 15.75, 1, 0.22, 5646471, S, blue
3753, 2018, 09, 22, 1251, 7, 0, 3036491, l, Black
3754, 2018, 09, 22, 21484.21, 1356, 75, 049681, xs, Red
3756, 2018, 09, 22, -59.18, 2, 0.333333333333333, 2038593, M, Blue
3757, 2018, 09, 23, 3177.4, 21, .14, 203521, XL, Black
];
```

Результаты

Qlik Sense table showing results of the makedate function being used in the load script.

transaction_id	Transaction Date
3750	30/08/2018
3751	07/09/2018
3752	16/09/2018
3753	22/09/2018
3754	22/09/2018
3756	22/09/2018
3757	23/09/2018

maketime

Эта функция возвращает время, рассчитанное в формате часы **hh**, минуты **mm** и секунды **ss**.

Синтаксис:

```
MakeTime(hh [ , mm [ , ss ] ])
```

Возвращаемые типы данных: двойное значение

Аргументы:

Аргументы

Аргумент	Описание
hh	Час – целое число.

Аргумент	Описание
mm	Минута – целое число. Если минута не задана, используется 00.
ss	Секунда – целое число. Если секунда не задана, используется 00.

Примеры и результаты:

Примеры написания скриптов

Пример	Результат
`maketime( 22 )`	возвращает 22:00:00
`maketime( 22, 17 )`	возвращает 22:17:00
`maketime( 22, 17, 52 )`	возвращает22:17:52

makeweekdate

Эта функция возвращает дату, рассчитанную в формате год YYYY, неделя WW и день недели D.

Синтаксис:

```
MakeWeekDate(YYYY [ , WW [ , D ] ])
```

Возвращаемые типы данных: двойное значение

Аргументы:

Аргументы

Аргумент	Описание
YYYY	Год – целое число.
WW	Неделя – целое число.
D	День недели – целое число. Если день недели не задан, используется 0 (понедельник).

Примеры и результаты:

Примеры написания скриптов

Пример	Результат
`makeweekdate(2014,6,6)`	возвращает 2014-02-09
`makeweekdate(2014,6,1)`	возвращает2014-02-04
`makeweekdate(2014,6)`	возвращает 2014-02-03 (для недели допускается значение 0)

minute

Эта функция возвращает время в минутах в виде целого числа, а дробное выражение **expression** интерпретируется как время согласно стандартной интерпретации чисел.

Синтаксис:

```
minute(expression)
```

Возвращаемые типы данных: целое

Примеры и результаты:

Примеры написания скриптов

Пример	Результат
`minute ( '09:14:36' )`	возвращает 14
`minute ( '0.5555' )`	возвращает 19 (так как 0,5555 = 13:19:55)

month

Эта функция возвращает двойное значение с именем месяца, как определено переменной окружения **MonthNames**, и целое в диапазоне от 1 до 12. Месяц высчитывается на основе интерпретации данных выражения согласно стандартной интерпретации чисел.

Функция возвращает название месяца в формате системной переменной `MonthName` для определенной даты. Она широко используется с целью создания поля даты в качестве измерения в основном календаре.

Синтаксис:

```
month(expression)
```

Возвращаемые типы данных: целое

Примеры функции

Пример	Результат
`month( '2012-10-12' )`	возвращает Oct (октябрь)
`month( '35648' )`	возвращает Aug (август), так как 35648 = 1997-08-06

monthend

Эта функция возвращает значение, соответствующее метке времени, включающей последнюю миллисекунду последнего дня месяца, содержащего значение, указанное в поле **date**. По умолчанию для вывода используется формат **DateFormat**,

установленный в скрипте.

Синтаксис:

```
MonthEnd(date[, period_no])
```

Возвращаемые типы данных: двойное значение

Аргументы:

Аргументы

Аргумент	Описание
date	Дата для вычисления.
period_no	**period_no** является целым числом, значение 0 которого или отсутствие значения означает месяц, содержащий значение, указанное в поле **date**. Отрицательные значения, заданные в поле **period_no**, означают предшествующие месяцы, положительные – последующие.

Примеры и результаты:
В этих примерах используется формат даты **DD/MM/YYYY**. Формат даты указан в операторе **SET DateFormat** в верхней части скрипта загрузки данных. Измените формат в примерах согласно своим пожеланиям.

Примеры написания скриптов

Пример	Результат
`monthend('19/02/2012')`	Возвращает 29/02/2012 23:59:59.
`monthend('19/02/2001', -1)`	Возвращает 31/01/2001 23:59:59.

Пример:

Добавьте образец скрипта в свое приложение и запустите. Чтобы увидеть результаты, добавьте поля, перечисленные в столбце результатов, в лист приложения.

В этом примере выполняется обнаружение последнего дня в месяце каждой даты счета в таблице, где базовая дата смещается на четыре месяца путем указания для элемента *period_no* значения 4.

```
TempTable:
LOAD RecNo() as InvID, * Inline [
InvDate
28/03/2012
10/12/2012
5/2/2013
31/3/2013
19/5/2013
15/9/2013
11/12/2013
2/3/2014
14/5/2014
13/6/2014
```

```
7/7/2014
4/8/2014
];

InvoiceData:
LOAD *,
MonthEnd(InvDate, 4) AS MthEnd
Resident TempTable;
Drop table TempTable;
```

Результирующая таблица содержит исходные даты и столбец с возвращенным значением функции monthend(). Чтобы отобразить полную метку времени, укажите форматирование на панели свойств.

Результирующая таблица

InvDate	MthEnd
28/03/2012	31/07/2012
10/12/2012	30/04/2013
5/2/2013	30/06/2013
31/3/2013	31/07/2013
19/5/2013	30/09/2013
15/9/2013	31/01//2014
11/12/2013	30/04//2014
2/3/2014	31/07//2014
14/5/2014	30/09/2014
13/6/2014	31/10/2014
7/7/2014	30/11/2014
4/8/2014	31/12/2014

monthname

Эта функция возвращает значение, отображающее месяц (в формате переменной **MonthNames** скрипта) и год с базовым числовым значением, соответствующим метке времени, включающей первую миллисекунду первого дня указанного месяца.

Синтаксис:

```
MonthName(date[, period_no])
```

Возвращаемые типы данных: двойное значение

Аргументы:

Аргументы

Аргумент	Описание
date	Дата для вычисления.
period_no	**period_no** является целым числом, значение 0 которого или отсутствие значения означает месяц, содержащий значение, указанное в поле **date**. Отрицательные значения, заданные в поле **period_no**, означают предшествующие месяцы, положительные – последующие.

Пример: Выражение диаграммы
В данном примере используется формат даты **DD/MM/YYYY**, указанный в операторе **SET DateFormat** в верхней части скрипта загрузки данных. Измените формат в примерах согласно своим пожеланиям. Для оператора **SET Monthnames** установлено значение Jan;Feb;Mar и т. д.

Примеры выражений диаграммы

Пример	Результат
`monthname('19/10/2013')`	Возвращает Oct 2013
`monthname('19/10/2013', -1)`	Возвращает Sep 2013

Пример: Скрипт загрузки
В этом примере для каждой даты счета в таблице имя месяца создается на основе имени месяца, смещенного на четыре месяца от поля base_date и на основе года.

В **Редакторе загрузки данных** создайте новый раздел, добавьте образец скрипта и запустите его. Затем добавьте на лист приложения как минимум поля, указанные в столбце с результатами, чтобы увидеть результаты.

Скрипт загрузки

```
TempTable:
LOAD RecNo() as InvID, * Inline [
InvDate
28/03/2012
10/12/2012
5/2/2013
31/3/2013
19/5/2013
15/9/2013
11/12/2013
2/3/2014
14/5/2014
13/6/2014
7/7/2014
4/8/2014
```

```
];

InvoiceData:
LOAD *,
MonthName(InvDate, 4) AS MthName
Resident TempTable;
Drop table TempTable;
```

Результаты

Результирующая таблица содержит исходные даты и столбец с возвращенным значением функции monthname().

InvDate	MthName
28/03/2012	Jul 2012
10/12/2012	Apr 2013
5/2/2013	Jun 2013
31/3/2013	Jul 2013
19/5/2013	Sep 2013
15/9/2013	Jan 2014
11/12/2013	Apr 2014
2/3/2014	Jul 2014
14/5/2014	Sep 2014
13/6/2014	Oct 2014
7/7/2014	Nov 2014
4/8/2014	Dec 2014

Пример: Скрипт загрузки
В данном примере для каждого поля transaction_date в таблице создается значение Returnable_Until. Значение Returnable_Until вычисляется путем смещения месяца transaction_date на один месяц вперед.

В **Редакторе загрузки данных** создайте новый раздел, добавьте образец скрипта и запустите его. Затем добавьте на лист приложения как минимум поля, указанные в столбце с результатами, чтобы увидеть результаты.

Скрипт загрузки

```
SET DateFormat='YYYYMMDD';
SET TimestampFormat='YYYYMMDD h:mm:ss[.fff] TT';
SET FirstMonthOfYear=1;
```

```
SET MonthNames='Jan;Feb;Mar;Apr;May;Jun;Jul;Aug;Sep;Oct;Nov;Dec';
SET
LongMonthNames='January;February;March;April;May;June;July;August;September;October;November;D
ecember';

Transactions:
Load
*,
MonthName(Date#(transaction_date,'YYYYMMDD'), 1) as Returnable_Until,
;

Load * Inline [
transaction_id, transaction_date, transaction_amount, transaction_quantity, discount,
customer_id, size, color_code
3750, 20180830, 12423.56, 23, 0,2038593, L, Red
3751, 20180907, 5356.31, 6, 0.1, 203521, m, orange
3752, 20180916, 15.75, 1, 0.22, 5646471, S, blue
3753, 20180922, 1251, 7, 0, 3036491, l, Black
3754, 20180922, 21484.21, 1356, 75, 049681, xs, Red
3756, 20180922, -59.18, 2, 0.333333333333333, 2038593, M, Blue
3757, 20180923, 3177.4, 21, .14, 203521, XL, Black
];
```

Результаты

Qlik Sense table showing results of the monthname function being used in the load script.

transaction_id	transaction_date	Returnable_Until
3750	20180830	Sep 2018
3751	20180907	Oct 2018
3752	20180916	Oct 2018
3753	20180922	Oct 2018
3754	20180922	Oct 2018
3756	20180922	Oct 2018
3757	20180923	Oct 2018

monthsend

Эта функция возвращает значение, соответствующее метке времени последней миллисекунды месяца, двухмесячного периода, квартала, триместра или полугодия, содержащих базовую дату. Также можно найти метку времени для предыдущего или последующего временного периода.

Синтаксис:

```
MonthsEnd(n_months, date[, period_no [, first_month_of_year]])
```

Возвращаемые типы данных: двойное значение

Аргументы:

Аргументы

Аргумент	Описание
n_months	Число месяцев, обозначающее период. Целое число или выражение, определяемое по целому числу, которое должно быть одним из следующих значений: 1 (эквивалентно функции inmonth()), 2 (двухмесячный период), 3 (эквивалентно функции inquarter()), 4 (триместр) или 6 (полугодие).
date	Дата для вычисления.
period_no	Период можно сместить, задав значение в поле **period_no**, целом числе или выражении, определяемом по целому числу, где 0 обозначает период, включающий значение, указанное в поле **base_date**. Отрицательные значения, заданные в поле **period_no**, означают предшествующие периоды, положительные – последующие.
first_month_of_year	Если необходимо работать с годами (финансовыми), которые начинаются не в январе, задайте значение от 2 до 12 в поле **first_month_of_year**.

Примеры и результаты:
В этих примерах используется формат даты **DD/MM/YYYY**. Формат даты указан в операторе **SET DateFormat** в верхней части скрипта загрузки данных. Измените формат в примерах согласно своим пожеланиям.

Примеры написания скриптов

Пример	Результат
`monthsend(4, '19/07/2013')`	Возвращает 31/08/2013.
`monthsend(4, '19/10/2013', -1)`	Возвращает 31/08/2013.
`monthsend(4, '19/10/2013', 0, 2)`	Возвращает 31/01/2014. Поскольку началом года становится месяц 2.

Пример:

Добавьте образец скрипта в свое приложение и запустите. Чтобы увидеть результаты, добавьте поля, перечисленные в столбце результатов, в лист приложения.

В этом примере выполняется обнаружение окончания последнего дня двухмесячного периода для каждой даты счета, смещенного вперед на один двухмесячный период.

```
TempTable:
LOAD RecNo() as InvID, * Inline [
InvDate
```

```
28/03/2012
10/12/2012
5/2/2013
31/3/2013
19/5/2013
15/9/2013
11/12/2013
2/3/2014
14/5/2014
13/6/2014
7/7/2014
4/8/2014
];

InvoiceData:
LOAD *,
MonthsEnd(2, InvDate, 1) AS BiMthsEnd
Resident TempTable;
Drop table TempTable;
```

Результирующая таблица содержит исходные даты и столбец с возвращенным значением функции MonthsEnd().

Результирующая таблица

InvDate	BiMthsEnd
28/03/2012	30/06/2012
10/12/2012	28/02/2013
5/2/2013	30/04/2013
31/3/2013	30/04/2013
19/5/2013	31/08/2013
15/9/2013	31/12/2013
11/12/2013	28/02/2014
2/3/2014	30/06/2014
14/5/2014	31/08/2014
13/6/2014	31/08/2014
7/7/2014	31/10/2014
4/8/2014	31/10/2014

monthsname

Эта функция возвращает значение, представляющее диапазон месяцев периода (форматированного согласно переменным скрипта **MonthNames**), а также года. Базовое числовое значение соответствует метке времени первой миллисекунды месяца, двухмесячного периода, квартала, триместра или полугодия, содержащих базовую дату.

Синтаксис:

```
MonthsName(n_months, date[, period_no[, first_month_of_year]])
```

Возвращаемые типы данных: двойное значение

Аргументы:

Аргументы

Аргумент	Описание
n_months	Число месяцев, обозначающее период. Целое число или выражение, определяемое по целому числу, которое должно быть одним из следующих значений: 1 (эквивалентно функции inmonth()), 2 (двухмесячный период), 3 (эквивалентно функции inquarter()), 4 (триместр) или 6 (полугодие).
date	Дата для вычисления.
period_no	Период можно сместить, задав значение в поле **period_no**, целом числе или выражении, определяемом по целому числу, где 0 обозначает период, включающий значение, указанное в поле **base_date**. Отрицательные значения, заданные в поле **period_no**, означают предшествующие периоды, положительные – последующие.
first_ month_of_ year	Если необходимо работать с годами (финансовыми), которые начинаются не в январе, задайте значение от 2 до 12 в поле **first_month_of_year**.

Примеры и результаты:

В этих примерах используется формат даты **DD/MM/YYYY**. Формат даты указан в операторе **SET DateFormat** в верхней части скрипта загрузки данных. Измените формат в примерах согласно своим пожеланиям.

Примеры написания скриптов

Пример	Результат
`monthsname(4, '19/10/2013')`	Возвращает Sep-Dec 2013. Поскольку в этом и других примерах оператор **SET Monthnames** задан как Jan;Feb;Mar и т. д.
`monthsname(4, '19/10/2013', -1)`	Возвращает May-Aug 2013.
`monthsname(4, '19/10/2013', 0, 2)`	Возвращает Oct-Jan 2014. Поскольку год указан начинающимся в месяце 2, поэтому четырехмесячный период заканчивается в первом месяце следующего года.

Пример:

Добавьте образец скрипта в свое приложение и запустите. Чтобы увидеть результаты, добавьте поля, перечисленные в столбце результатов, в лист приложения.

В этом примере для каждой даты счета в таблице имя месяцев создается на основе диапазона месяцев в двухмесячном периоде и на основе года. Диапазон смещен месяцами 4x2 путем указания для элемента period_no значения 4.

```
TempTable:
LOAD RecNo() as InvID, * Inline [
InvDate
28/03/2012
10/12/2012
5/2/2013
31/3/2013
19/5/2013
15/9/2013
11/12/2013
2/3/2014
14/5/2014
13/6/2014
7/7/2014
4/8/2014
];

InvoiceData:
LOAD *,
MonthsName(2, InvDate, 4) AS MthsName
Resident TempTable;
Drop table TempTable;
```

Результирующая таблица содержит исходные даты и столбец с возвращенным значением функции monthsname().

Результирующая таблица

InvDate	MthsName
28/03/2012	Nov-Dec 2012
10/12/2012	Jul-Aug 2013
5/2/2013	Sep-Oct 2013
31/3/2013	Nov-Dec2013
19/5/2013	Jan-Feb 2014
15/9/2013	May-Jun 2014
11/12/2013	Jul-Aug 2014
2/3/2014	Nov-Dec 2014
14/5/2014	Jan-Feb 2015

InvDate	MthsName
13/6/2014	Jan-Feb 2015
7/7/2014	Mar-Apr 2015
4/8/2014	Mar-Apr 2015

monthsstart

Эта функция возвращает значение, соответствующее метке времени первой миллисекунды месяца, двухмесячного периода, квартала, триместра или полугодия, содержащих базовую дату. Также можно найти метку времени для предыдущего или последующего временного периода.

Синтаксис:

```
MonthsStart(n_months, date[, period_no [, first_month_of_year]])
```

Возвращаемые типы данных: двойное значение

Аргументы:

Аргументы

Аргумент	Описание
n_months	Число месяцев, обозначающее период. Целое число или выражение, определяемое по целому числу, которое должно быть одним из следующих значений: 1 (эквивалентно функции inmonth()), 2 (двухмесячный период), 3 (эквивалентно функции inquarter()), 4 (триместр) или 6 (полугодие).
date	Дата для вычисления.
period_no	Период можно сместить, задав значение в поле **period_no**, целом числе или выражении, определяемом по целому числу, где 0 обозначает период, включающий значение, указанное в поле **base_date**. Отрицательные значения, заданные в поле **period_no**, означают предшествующие периоды, положительные – последующие.
first_ month_of_ year	Если необходимо работать с годами (финансовыми), которые начинаются не в январе, задайте значение от 2 до 12 в поле **first_month_of_year**.

Примеры и результаты:

В этих примерах используется формат даты **DD/MM/YYYY**. Формат даты указан в операторе **SET DateFormat** в верхней части скрипта загрузки данных. Измените формат в примерах согласно своим пожеланиям.

Примеры написания скриптов

Пример	Результат
`monthsstart(4, '19/10/2013')`	Возвращает 1/09/2013.
`monthsstart(4, '19/10/2013, -1)`	Возвращает 01/05/2013.
`monthsstart(4, '19/10/2013', 0, 2 )`	Возвращает 01/10/2013. Поскольку началом года становится месяц 2.

Добавьте образец скрипта в свое приложение и запустите. Чтобы увидеть результаты, добавьте поля, перечисленные в столбце результатов, в лист приложения.

В этом примере выполняется обнаружение первого дня двухмесячного периода для каждой даты счета, смещенного вперед на один двухмесячный период.

```
TempTable:
LOAD RecNo() as InvID, * Inline [
InvDate
28/03/2012
10/12/2012
5/2/2013
31/3/2013
19/5/2013
15/9/2013
11/12/2013
2/3/2014
14/5/2014
13/6/2014
7/7/2014
4/8/2014
];

InvoiceData:
LOAD *,
MonthsStart(2, InvDate, 1) AS BiMthsStart
Resident TempTable;
Drop table TempTable;
```

Результирующая таблица содержит исходные даты и столбец с возвращенным значением функции MonthsStart().

Результирующая таблица

InvDate	BiMthsStart
28/03/2012	01/05/2012
10/12/2012	01/01/2013
5/2/2013	01/03/2013
31/3/2013	01/05/2013

InvDate	BiMthsStart
19/5/2013	01/07/2013
15/9/2013	01/11/2013
11/12/2013	01/01/2014
2/3/2014	01/05/2014
14/5/2014	01/07/2014
13/6/2014	01/07/2014
7/7/2014	01/09/2014
4/8/2014	01/09/2014

monthstart

Эта функция возвращает значение, соответствующее метке времени, включающей первую миллисекунду первого дня месяца, содержащего значение, указанное в поле **date**. По умолчанию для вывода используется формат **DateFormat**, установленный в скрипте.

Синтаксис:

```
MonthStart(date[, period_no])
```

Возвращаемые типы данных: двойное значение

Аргументы:

Аргументы

Аргумент	Описание
date	Дата для вычисления.
period_no	**period_no** является целым числом, значение 0 которого или отсутствие значения означает месяц, содержащий значение, указанное в поле **date**. Отрицательные значения, заданные в поле **period_no**, означают предшествующие месяцы, положительные – последующие.

Примеры и результаты:

В этих примерах используется формат даты **DD/MM/YYYY**. Формат даты указан в операторе **SET DateFormat** в верхней части скрипта загрузки данных. Измените формат в примерах согласно своим пожеланиям.

Примеры написания скриптов

Пример	Результат
`monthstart('19/10/2001')`	Возвращает 01/10/2001.
`monthstart('19/10/2001', -1)`	Возвращает 01/09/2001.

Пример:

Добавьте образец скрипта в свое приложение и запустите. Чтобы увидеть результаты, добавьте поля, перечисленные в столбце результатов, в лист приложения.

В этом примере выполняется обнаружение первого дня в месяце каждой даты счета в таблице, где base_date смещается на четыре месяца путем указания для элемента period_no значения 4.

```
TempTable:
LOAD RecNo() as InvID, * Inline [
InvDate
28/03/2012
10/12/2012
5/2/2013
31/3/2013
19/5/2013
15/9/2013
11/12/2013
2/3/2014
14/5/2014
13/6/2014
7/7/2014
4/8/2014
];

InvoiceData:
LOAD *,
MonthStart(InvDate, 4) AS MthStart
Resident TempTable;
Drop table TempTable;
```

Результирующая таблица содержит исходные даты и столбец с возвращенным значением функции monthstart(). Чтобы отобразить полную метку времени, укажите форматирование на панели свойств.

Результирующая таблица

InvDate	MthStart
28/03/2012	01/07/2012
10/12/2012	01/04/2013
5/2/2013	01/06/2013
31/3/2013	01/07/2013
19/5/2013	01/09/2013

InvDate	MthStart
15/9/2013	01/01/2014
11/12/2013	01/04/2014
2/3/2014	01/07/2014
14/5/2014	01/09/2014
13/6/2014	01/10/2014
7/7/2014	01/11/2014
4/8/2014	01/12/2014

networkdays

Функция **networkdays** возвращает число рабочих дней (понедельник-пятница) между и включая значения, указанные в поле **start_date** и **end_date**, учитывая выходные, которые можно дополнительно задать в поле **holiday**.

Синтаксис:

```
networkdays (start_date, end_date [, holiday])
```

Возвращаемые типы данных: целое

Аргументы:

Аргументы

Аргумент	Описание
start_date	Начальная дата для вычисления.
end_date	Конечная дата для вычисления.
holiday	Периоды выходных дней для исключения из рабочих дней. Период выходных дней указан как дата начала и дата окончания, разделенные запятыми. **Пример:** `'25/12/2013', '26/12/2013'` Можно указать несколько периодов выходных дней, разделенных запятыми. **Пример:** `'25/12/2013', '26/12/2013', '31/12/2013', '01/01/2014'`

Примеры и результаты:
В этих примерах используется формат даты **DD/MM/YYYY**. Формат даты указан в операторе **SET DateFormat** в верхней части скрипта загрузки данных. Измените формат в примерах согласно своим пожеланиям.

Примеры написания скриптов

Пример	Результат
`networkdays ('19/12/2013', '07/01/2014')`	Возвращает 14. В этом примере выходные дни не учитываются.
`networkdays ('19/12/2013', '07/01/2014', '25/12/2013', '26/12/2013')`	Возвращает 12. В этом примере учитываются выходные в периоде с 25/12/2013 по 26/12/2013.
`networkdays ('19/12/2013', '07/01/2014', '25/12/2013', '26/12/2013', '31/12/2013', '01/01/2014')`	Возвращает 10. В этом примере учитываются двухдневные периоды выходных дней.

Пример:

Добавьте образец скрипта в свое приложение и запустите. Чтобы увидеть результаты, добавьте поля, перечисленные в столбце результатов, в лист приложения.

```
PayTable:
LOAD recno() as InvID, * INLINE [
InvRec|InvPaid
28/03/2012|28/04/2012
10/12/2012|01/01/2013
5/2/2013|5/3/2013
31/3/2013|01/5/2013
19/5/2013|12/6/2013
15/9/2013|6/10/2013
11/12/2013|12/01/2014
2/3/2014|2/4/2014
14/5/2014|14/6/2014
13/6/2014|14/7/2014
7/7/2014|14/8/2014
4/8/2014|4/9/2014
] (delimiter is '|');
NrDays:
Load *,
NetWorkDays(InvRec,InvPaid) As PaidDays
Resident PayTable;
Drop table PayTable;
```

Результирующая таблица показывает возвращенные значения функции NetworkDays для каждой записи в таблице.

Результирующая таблица

InvID	InvRec	InvPaid	PaidDays
1	28/03/2012	28/04/2012	23
2	10/12/2012	01/01/2013	17
3	5/2/2013	5/3/2013	21

InvID	InvRec	InvPaid	PaidDays
4	31/3/2013	01/5/2013	23
5	19/5/2013	12/6/2013	18
6	15/9/2013	6/10/2013	15
7	11/12/2013	12/01/2014	23
8	2/3/2014	2/4/2014	23
9	14/5/2014	14/6/2014	23
10	13/6/2014	14/7/2014	22
11	7/7/2014	14/8/2014	29
12	4/8/2014	4/9/2014	24

now

Эта функция возвращает метку текущего времени по системным часам. Значение по умолчанию – 1.

Синтаксис:

```
now([ timer_mode])
```

Возвращаемые типы данных: двойное значение

Аргументы:

Аргументы

Аргумент	Описание
timer_mode	Может иметь следующие значения: 0 (время последней завершенной загрузки данных) 1 (время вызова функции) 2 (время открытия приложения) *Если вы используете функцию в скрипте загрузки данных, функция **timer_mode=0** выдаст время последней завершенной загрузки данных, а **timer_mode=1** выдаст время вызова функции в текущей загрузке данных.*

Примеры и результаты:

Примеры написания скриптов

Пример	Результат
`now( 0)`	Возвращает время завершения последней загрузки данных.
`now( 1)`	При использовании в выражении визуализации будет возвращено время вызова функции. При использовании в скрипте загрузки данных будет возвращено время вызова функции в текущей загрузке данных.
`now( 2)`	Возвращает время открытия приложения.

quarterend

Эта функция возвращает значение, соответствующее метке времени, включающей последнюю миллисекунду квартала, содержащего значение, указанное в поле **date**. По умолчанию для вывода используется формат **DateFormat**, установленный в скрипте.

Синтаксис:

```
QuarterEnd(date[, period_no[, first_month_of_year]])
```

Возвращаемые типы данных: двойное значение

Аргументы:

Аргументы

Аргумент	Описание
date	Дата для вычисления.
period_no	**period_no** – целое число, где 0 обозначает квартал, включающий значение, указанное в поле **date**. Отрицательные значения, заданные в поле **period_no**, означают предшествующие кварталы, положительные – последующие.
first_ month_of_ year	Если необходимо работать с годами (финансовыми), которые начинаются не в январе, задайте значение от 2 до 12 в поле **first_month_of_year**.

Примеры и результаты:
В этих примерах используется формат даты **DD/MM/YYYY**. Формат даты указан в операторе **SET DateFormat** в верхней части скрипта загрузки данных. Измените формат в примерах согласно своим пожеланиям.

Примеры написания скриптов

Пример	Результат
`quarterend('29/10/2005')`	Возвращает 31/12/2005 23:59:59.
`quarterend('29/10/2005', -1)`	Возвращает 30/09/2005 23:59:59.
`quarterend('29/10/2005', 0, 3)`	Возвращает 30/11/2005 23:59:59.

Пример:

Добавьте образец скрипта в свое приложение и запустите. Чтобы увидеть результаты, добавьте поля, перечисленные в столбце результатов, в лист приложения.

В этом примере выполняется обнаружение последнего дня в квартале каждой даты счета в таблице, где первый месяц в году указан как месяц 3.

```
TempTable:
LOAD RecNo() as InvID, * Inline [
InvDate
28/03/2012
10/12/2012
5/2/2013
31/3/2013
19/5/2013
15/9/2013
11/12/2013
2/3/2014
14/5/2014
13/6/2014
7/7/2014
4/8/2014
];

InvoiceData:
LOAD *,
QuarterEnd(InvDate, 0, 3) AS QtrEnd
Resident TempTable;
Drop table TempTable;
```

Результирующая таблица содержит исходные даты и столбец с возвращенным значением функции quarterend(). Чтобы отобразить полную метку времени, укажите форматирование на панели свойств.

Результирующая таблица

InvDate	QtrEnd
28/03/2012	31/05/2012
10/12/2012	28/02/2013
5/2/2013	28/02/2013

InvDate	QtrEnd
31/3/2013	31/05/2013
19/5/2013	31/05/2013
15/9/2013	30/11/2013
11/12/2013	28/02/2014
2/3/2014	31/05/2014
14/5/2014	31/05/2014
13/6/2014	31/08/2014
7/7/2014	31/08/2014
4/8/2014	31/08/2014

quartername

Эта функция возвращает значение, отображающее месяцы квартала (в формате переменной **MonthNames** скрипта) и год с базовым числовым значением, соответствующим метке времени, включающей первую миллисекунду первого дня квартала.

Синтаксис:

```
QuarterName(date[, period_no[, first_month_of_year]])
```

Возвращаемые типы данных: двойное значение

Аргументы:

Аргументы

Аргумент	Описание
date	Дата для вычисления.
period_no	**period_no** – целое число, где 0 обозначает квартал, включающий значение, указанное в поле **date**. Отрицательные значения, заданные в поле **period_no**, означают предшествующие кварталы, положительные – последующие.
first_ month_of_ year	Если необходимо работать с годами (финансовыми), которые начинаются не в январе, задайте значение от 2 до 12 в поле **first_month_of_year**.

Примеры и результаты:

Примеры написания скриптов

Пример	Результат
`quartername('29/10/2013')`	Возвращает Oct-Dec 2013.
`quartername('29/10/2013', -1)`	Возвращает Jul-Sep 2013.
`quartername('29/10/2013', 0, 3)`	Возвращает Sep-Nov 2013.

Пример:

Добавьте образец скрипта в свое приложение и запустите. Чтобы увидеть результаты, добавьте поля, перечисленные в столбце результатов, в лист приложения.

В этом примере для каждой даты счета в таблице имя квартала создается на основе квартала, содержащего *InvID*. Первый месяц в году указан как месяц 4.

```
TempTable:
LOAD RecNo() as InvID, * Inline [
InvDate
28/03/2012
10/12/2012
5/2/2013
31/3/2013
19/5/2013
15/9/2013
11/12/2013
2/3/2014
14/5/2014
13/6/2014
7/7/2014
4/8/2014
];

InvoiceData:
LOAD *,
QuarterName(InvDate, 0, 4) AS QtrName
Resident TempTable;
Drop table TempTable;
```

Результирующая таблица содержит исходные даты и столбец с возвращенным значением функции quartername().

Результирующая таблица

InvDate	QtrName
28/03/2012	Jan-Mar 2011
10/12/2012	Oct-Dec 2012
5/2/2013	Jan-Mar 2012
31/3/2013	Jan-Mar 2012

InvDate	QtrName
19/5/2013	Apr-Jun 2013
15/9/2013	Jul-Sep 2013
11/12/2013	Oct-Dec 2013
2/3/2014	Jan-Mar 2013
14/5/2014	Apr-Jun 2014
13/6/2014	Apr-Jun 2014
7/7/2014	Jul-Sep 2014
4/8/2014	Jul-Sep 2014

quarterstart

Эта функция возвращает значение, соответствующее метке времени, включающей первую миллисекунду квартала, содержащего значение, указанное в поле **date**. По умолчанию для вывода используется формат **DateFormat**, установленный в скрипте.

Синтаксис:

```
QuarterStart(date[, period_no[, first_month_of_year]])
```

Возвращаемые типы данных: двойное значение

Аргументы:

Аргументы

Аргумент	Описание
date	Дата для вычисления.
period_no	**period_no** – целое число, где 0 обозначает квартал, включающий значение, указанное в поле **date**. Отрицательные значения, заданные в поле **period_no**, означают предшествующие кварталы, положительные – последующие.
first_month_of_year	Если необходимо работать с годами (финансовыми), которые начинаются не в январе, задайте значение от 2 до 12 в поле **first_month_of_year**.

Примеры и результаты:

В этих примерах используется формат даты **DD/MM/YYYY**. Формат даты указан в операторе **SET DateFormat** в верхней части скрипта загрузки данных. Измените формат в примерах согласно своим пожеланиям.

Примеры написания скриптов

Пример	Результат
`quarterstart('29/10/2005')`	Возвращает 01/10/2005.
`quarterstart('29/10/2005', -1 )`	Возвращает 01/07/2005.
`quarterstart('29/10/2005', 0, 3)`	Возвращает 01/09/2005.

Пример:

Добавьте образец скрипта в свое приложение и запустите. Чтобы увидеть результаты, добавьте поля, перечисленные в столбце результатов, в лист приложения.

В этом примере выполняется обнаружение первого дня в квартале каждой даты счета в таблице, где первый месяц в году указан как месяц 3.

```
TempTable:
LOAD RecNo() as InvID, * Inline [
InvDate
28/03/2012
10/12/2012
5/2/2013
31/3/2013
19/5/2013
15/9/2013
11/12/2013
2/3/2014
14/5/2014
13/6/2014
7/7/2014
4/8/2014
];

InvoiceData:
LOAD *,
QuarterStart(InvDate, 0, 3) AS QtrStart
Resident TempTable;
Drop table TempTable;
```

Результирующая таблица содержит исходные даты и столбец с возвращенным значением функции quarterstart(). Чтобы отобразить полную метку времени, укажите форматирование на панели свойств.

Результирующая таблица

InvDate	QtrStart
28/03/2012	01/03/2012
10/12/2012	01/12/2012
5/2/2013	01/12/2012
31/3/2013	01/03/2013

InvDate	QtrStart
19/5/2013	01/03/2013
15/9/2013	01/09/2013
11/12/2013	01/12/2013
2/3/2014	01/03/2014
14/5/2014	01/03/2014
13/6/2014	01/06/2014
7/7/2014	01/06/2014
4/8/2014	01/06/2014

second

Эта функция возвращает время в секундах в виде целого числа, а дробное выражение **expression** интерпретируется как время согласно стандартной интерпретации чисел.

Синтаксис:

```
second (expression)
```

Возвращаемые типы данных: целое

Примеры и результаты:

Примеры написания скриптов

Пример	Результат
`second( '09:14:36' )`	возвращает 36
`second( '0.5555' )`	возвращает 55 (так как 0,5555 = 13:19:55)

setdateyear

Данная функция берет в качестве входных значений **timestamp** и **year** и обновляет значение **timestamp** с учетом указанного входного значения **year** .

Синтаксис:

```
setdateyear (timestamp, year)
```

Возвращаемые типы данных: двойное значение

Аргументы:

Аргументы

Аргумент	Описание
timestamp	Стандартная метка времени Qlik Sense (часто просто дата).
year	Четырехзначный год.

Примеры и результаты:
В этих примерах используется формат даты **DD/MM/YYYY**. Формат даты указан в операторе **SET DateFormat** в верхней части скрипта загрузки данных. Измените формат в примерах согласно своим пожеланиям.

Примеры написания скриптов

Пример	Результат
`setdateyear ('29/10/2005', 2013)`	Возвращает '29/10/2013'
`setdateyear ('29/10/2005 04:26:14', 2013)`	Возвращает «29/10/2013 04:26:14» Чтобы задать время как часть метки времени в визуализации, необходимо задать для форматирования числа значение «Дата» и выбрать значение, которое отображает значения времени, для параметра «Форматирование».

Пример:

Добавьте образец скрипта в свое приложение и запустите. Чтобы увидеть результаты, добавьте поля, перечисленные в столбце результатов, в лист приложения.

```
SetYear:
Load *,
SetDateYear(testdates, 2013) as NewYear
Inline [
testdates
1/11/2012
10/12/2012
1/5/2013
2/1/2013
19/5/2013
15/9/2013
11/12/2013
2/3/2014
14/5/2014
13/6/2014
7/7/2014
4/8/2014
];
```

Результирующая таблица содержит исходные даты и столбец, в котором для года было задано значение 2013.

Результирующая таблица

testdates	NewYear
1/11/2012	1/11/2013
10/12/2012	10/12/2013
2/1/2012	2/1/2013
1/5/2013	1/5/2013
19/5/2013	19/5/2013
15/9/2013	15/9/2013
11/12/2013	11/12/2013
2/3/2014	2/3/2013
14/5/2014	14/5/2013
13/6/2014	13/6/2013
7/7/2014	7/7/2013
4/8/2014	4/8/2013

setdateyearmonth

Данная функция берет в качестве входных значений **timestamp**, **month** и **year** и обновляет значение **timestamp** с учетом указанных входных значений **year** и **month** . .

Синтаксис:

```
SetDateYearMonth (timestamp, year, month)
```

Возвращаемые типы данных: двойное значение

Аргументы:

Аргументы

Аргумент	Описание
timestamp	Стандартная метка времени Qlik Sense (часто просто дата).
year	Четырехзначный год.
month	Месяц, заданный в одно- или двухразрядном формате.

Примеры и результаты:
В этих примерах используется формат даты **DD/MM/YYYY**. Формат даты указан в операторе **SET DateFormat** в верхней части скрипта загрузки данных. Измените формат в примерах согласно своим пожеланиям.

Примеры написания скриптов

Пример	Результат
`setdateyearmonth ('29/10/2005', 2013, 3)`	Возвращает '29/03/2013'
`setdateyearmonth ('29/10/2005 04:26:14', 2013, 3)`	Возвращает «29/03/2013 04:26:14» Чтобы задать время как часть метки времени в визуализации, необходимо задать для форматирования числа значение «Дата» и выбрать значение, которое отображает значения времени, для параметра «Форматирование».

Пример:

Добавьте образец скрипта в свое приложение и запустите. Чтобы увидеть результаты, добавьте поля, перечисленные в столбце результатов, в лист приложения.

```
SetYearMonth:
Load *,
SetDateYearMonth(testdates, 2013,3) as NewYearMonth
Inline [
testdates
1/11/2012
10/12/2012
2/1/2013
19/5/2013
15/9/2013
11/12/2013
14/5/2014
13/6/2014
7/7/2014
4/8/2014
];
```

Результирующая таблица содержит исходные даты и столбец, в котором для года было задано значение 2013.

Результирующая таблица

testdates	NewYearMonth
1/11/2012	1/3/2013
10/12/2012	10/3/2013
2/1/2012	2/3/2013

testdates	NewYearMonth
19/5/2013	19/3/2013
15/9/2013	15/3/2013
11/12/2013	11/3/2013
14/5/2014	14/3/2013
13/6/2014	13/3/2013
7/7/2014	7/3/2013
4/8/2014	4/3/2013

timezone

Эта функция возвращает имя текущего часового пояса, соответствующее имени, используемому в Windows.

Синтаксис:

```
TimeZone( )
```

Возвращаемые типы данных: строка

Пример:

```
timezone( )
```

today

Эта функция возвращает текущую дату по системным часам.

Синтаксис:

```
today([ timer_mode])
```

Возвращаемые типы данных: двойное значение

Аргументы:

Аргументы

Аргумент	Описание
timer_mode	Может иметь следующие значения: 0 (день последней завершенной загрузки данных) 1 (день вызова функции) 2 (день открытия приложения) *Если вы используете функцию в скрипте загрузки данных, функция **timer_mode=0** выдаст день последней завершенной загрузки данных, а **timer_mode=1** выдаст день текущей загрузки данных.*

Примеры и результаты:

Примеры написания скриптов

Пример	Результат
`Today( 0)`	Возвращает день последней завершенной загрузки данных.
`Today( 1)`	При использовании в выражении визуализации будет возвращен день вызова функции. При использовании в скрипте загрузки данных будет возвращен день начала текущей загрузки данных.
`Today( 2)`	Возвращает день открытия приложения.

UTC

Возвращает текущее время Coordinated Universal Time.

Синтаксис:

```
UTC( )
```

Возвращаемые типы данных: dual

Пример:

```
utc( )
```

week

Эта функция возвращает номер недели в виде целого числа согласно стандарту ISO 8601. Номер недели высчитывается на основе интерпретации данных выражения согласно стандартной интерпретации чисел.

Синтаксис:

```
week(timestamp [, first_week_day [, broken_weeks [, reference_day]]])
```

Возвращаемые типы данных: целое

Аргументы

Аргумент	Описание
timestamp	Дата для вычисления в виде метки времени или выражения, определяемого по метке времени, для преобразования, например '2012-10-12'.
first_week_day	Если не указать **first_week_day**, значение переменной **FirstWeekDay** будет использовано как первый день недели. Если необходимо использовать другой день в качестве первого дня недели, установите для элемента **first_week_day** следующее значение: • 0 для понедельника • 1 для вторника • 2 для среды • 3 для четверга • 4 для пятницы • 5 для субботы • 6 для воскресенья Целое число, возвращенное этой функцией, теперь будет использовать первый день недели, заданный параметром **first_week_day**.

Аргумент	Описание
broken_ weeks	Если параметр **broken_weeks** не указан, значение переменной **BrokenWeeks** будет использовано для определения, какими должны быть недели: целыми или разбитыми. По умолчанию в функциях Qlik Sense используются целые недели. Это означает следующее: • В одних годах 1-я неделя начинается в декабре, а в других годах 52-я или 53-я неделя заканчивается в январе. • В 1-ой неделе всегда не менее четырех дней в январе. В качестве альтернативы можно использовать разбиение недель. • 52-я или 53-я неделя не будет продолжена в январе следующего года. • 1-я неделя будет начинаться 1 января и в большинстве случаев она будет неполной. Могут использоваться следующие значения: • 0 (= использовать целые недели) • 1 (= использовать разбитые недели)
reference_ day	Если параметр **reference_day** не указан, значение переменной **ReferenceDay** будет использовано для определения, какой день в январе должен быть задан в качестве дня ссылки, чтобы определить неделю 1. По умолчанию в функциях Qlik Sense используется 4 как день ссылки. Это значит, что неделя 1 должна содержать значение «январь 4» , или, другими словами, в неделе 1 всегда должно быть не меньше 4 дней в январе. Используйте следующие значения, чтобы задать день ссылки: • 1 (= январь 1) • 2 (= январь 2) • 3 (= январь 3) • 4 (= январь 4) • 5 (= январь 5) • 6 (= январь 6) • 7 (= январь 7)

Примеры и результаты:

Примеры написания скриптов

Пример	Результат
`week( '2012-10-12' )`	возвращает 41.

Пример	Результат
`week( '35648' )`	возвращает 32, так как 35 648 = 1997-08-06
`week('2012-10-12', 0, 1)`	возвращает 42

weekday

Эта функция возвращает двойное значение со следующим:

- Имя дня, как определено переменной окружения **DayNames**.
- Целое от 0 до 6, соответствующее номинальному дню недели (0-6).

Синтаксис:

```
weekday(date [,first_week_day=0])
```

Возвращаемые типы данных: двойное значение

Аргументы:

Аргументы

Аргумент	Описание
date	Дата для вычисления.
first_week_day	Если не указать **first_week_day**, значение переменной **FirstWeekDay** будет использовано как первый день недели. Если необходимо использовать другой день в качестве первого дня недели, установите для элемента **first_week_day** следующее значение: • 0 для понедельника • 1 для вторника • 2 для среды • 3 для четверга • 4 для пятницы • 5 для субботы • 6 для воскресенья Целое число, возвращенное этой функцией, теперь будет использовать первый день недели, заданный в элементе **first_week_day** как основной (0). *FirstWeekDay (page 162)*

Пример: Выражение диаграммы
Если не указано иначе, в этих примерах для элемента **FirstWeekDay** установлено значение 0.

Примеры написания скриптов

Пример	Результат
`weekday( '1971-10-12' )`	возвращает 'Tue' (вторник) и 1
`weekday( '1971-10-12' , 6)`	возвращает 'Tue' (вторник) и 2. В этом примере мы используем воскресенье (6) в качестве первого дня недели.

`SET FirstWeekDay = 6;`
 	
`...`
`weekday( '1971-10-12')`
	возвращает 'Tue' (вторник) и 2.

Пример: Скрипт загрузки

Скрипт загрузки

weekday можно использовать в скрипте загрузки для возврата строки и числа, представляющих день недели, даже в том случае, если в скрипте уже установлены значения *FirstWeekDay* и *ReferenceDay*. Приведенный ниже скрипт загрузки заимствует определенные значения *FirstWeekDay* и *ReferenceDay*, а затем использует *weekday* для возврата строк и чисел, представляющих дни недели, из данных столбца *transaction_date*.

В представленных результатах столбец *Day* содержит возвращенные строки, а *Numeric value of Day* и *Numeric value of week starting from Sunday* содержат возвращенные числовые значения. В скрипте загрузки *weekday* умножается на 1. Это обеспечивает возврат числовых данных.

В **Редакторе загрузки данных** создайте новый раздел, добавьте образец скрипта и запустите его. Затем добавьте на лист приложения как минимум поля, указанные в столбце с результатами, чтобы увидеть результаты.

```
SET DateFormat='DD/MM/YYYY';
SET TimestampFormat='DD/MM/YYYY h:mm:ss[.fff] TT';
SET FirstWeekDay=0;
SET BrokenWeeks=1;
SET ReferenceDay=0;
SET DayNames='Mon;Tue;Wed;Thu;Fri;Sat;Sun';
SET LongDayNames='Monday;Tuesday;Wednesday;Thursday;Friday;Saturday;Sunday';

Transactions:
Load
*,
WeekDay(transaction_date) as [Day],
1*WeekDay(transaction_date) as [Numeric value of Day]
1*WeekDay(transaction_date, 6) as [Numeric value of a week starting from Sunday],
;
Load * Inline [
transaction_id, transaction_date, transaction_amount, transaction_quantity, discount,
customer_id, size, color_code
3750, 20180830, 12423.56, 23, 0,2038593, L, Red
```

```
3751, 20180907, 5356.31, 6, 0.1, 203521, m, orange
3752, 20180916, 15.75, 1, 0.22, 5646471, S, blue
3753, 20180922, 1251, 7, 0, 3036491, l, Black
3754, 20180922, 21484.21, 1356, 75, 049681, xs, Red
3756, 20180922, -59.18, 2, 0.333333333333333, 2038593, M, Blue
3757, 20180923, 3177.4, 21, .14, 203521, XL, Black
];
```

Результаты

Таблица Qlik Sense, в которой представлены результаты использования функции weekday в скрипте загрузки.

transaction_id	transaction_ date	День	Numeric value of Day	Numeric value for a week starting from Sunday
3750	20180830	Thu	3	4
3751	20180907	Thu	3	4
3752	20180916	Sat	5	6
3753	20180922	Fri	4	5
3754	20180922	Fri	4	5
3756	20180922	Fri	4	5
3757	20180923	Sat	5	6

weekend

Эта функция возвращает значение, соответствующее метке времени, включающей последнюю миллисекунду последней даты (воскресенья) календарной недели, включающей дату, заданную в поле **date**. По умолчанию для вывода используется формат даты **DateFormat**, установленный в скрипте.

Синтаксис:

```
WeekEnd(date [, period_no[, first_week_day]])
```

Возвращаемые типы данных: двойное значение

Аргументы:

Аргументы

Аргумент	Описание
date	Дата для вычисления.
period_no	**shift** – целое число, где 0 обозначает неделю, включающую значение, указанное в поле **date**. Отрицательные значения, заданные в поле shift, означают предшествующие недели, положительные – последующие.

Аргумент	Описание
first_week_day	Указывает день начала недели. Если не указано, используется значение переменной **FirstWeekDay**. Возможные значения **first_week_day**: • 0 для понедельника • 1 для вторника • 2 для среды • 3 для четверга • 4 для пятницы • 5 для субботы • 6 для воскресенья *FirstWeekDay (page 162)*

Примеры и результаты:
В этих примерах используется формат даты **DD/MM/YYYY**. Формат даты указан в операторе **SET DateFormat** в верхней части скрипта загрузки данных. Измените формат в примерах согласно своим пожеланиям.

Пример	Результат
`weekend('10/01/2013')`	Возвращает 12/01/2013 23:59:59.
`weekend('10/01/2013', -1)`	Возвращает 06/01/2013 23:59:59.
`weekend('10/01/2013', 0, 1)`	Возвращает 14/01/2013 23:59:59.

Пример:

Добавьте образец скрипта в свое приложение и запустите. Чтобы увидеть результаты, добавьте поля, перечисленные в столбце результатов, в лист приложения.

В этом примере выполняется обнаружение последнего дня недели после недели с датами каждого счета в таблице.

```
TempTable:
LOAD RecNo() as InvID, * Inline [
InvDate
28/03/2012
10/12/2012
5/2/2013
31/3/2013
19/5/2013
15/9/2013
11/12/2013
2/3/2014
14/5/2014
13/6/2014
7/7/2014
```

```
4/8/2014
];

InvoiceData:
LOAD *,
WeekEnd(InvDate, 1) AS WkEnd
Resident TempTable;
Drop table TempTable;
```

Результирующая таблица содержит исходные даты и столбец с возвращенным значением функции weekend(). Чтобы отобразить полную метку времени, укажите форматирование на панели свойств.

Результирующая таблица

InvDate	WkEnd
28/03/2012	08/04/2012
10/12/2012	23/12/2012
5/2/2013	17/02/2013
31/3/2013	07/04/2013
19/5/2013	26/05/2013
15/9/2013	22/09/2013
11/12/2013	22/12/2013
2/3/2014	09/03/2014
14/5/2014	25/05/2014
13/6/2014	22/06/2014
7/7/2014	20/07/2014
4/8/2014	17/08/2014

weekname

Эта функция возвращает значение года и номер недели с базовым числовым значением, соответствующим метке времени, включающей первую миллисекунду первого дня недели, содержащего значение, указанное в поле **date**.

Синтаксис:

```
WeekName(date[, period_no[,first_week_day]])
```

Возвращаемые типы данных: двойное значение

Аргументы:

Аргументы

Аргумент	Описание
date	Дата для вычисления.
period_no	**shift** – целое число, где 0 обозначает неделю, включающую значение, указанное в поле **date**. Отрицательные значения, заданные в поле shift, означают предшествующие недели, положительные – последующие.
first_week_day	Указывает день начала недели. Если не указано, используется значение переменной **FirstWeekDay**. Возможные значения **first_week_day**: • 0 для понедельника • 1 для вторника • 2 для среды • 3 для четверга • 4 для пятницы • 5 для субботы • 6 для воскресенья *FirstWeekDay (page 162)*

Примеры и результаты:

Пример	Результат
`weekname('12/01/2013')`	Возвращает 2013/02.
`weekname('12/01/2013', -1)`	Возвращает 2013/01.
`weekname('12/01/2013', 0, 1)`	Возвращает 2013/02.

Пример:

Добавьте образец скрипта в свое приложение и запустите. Чтобы увидеть результаты, добавьте поля, перечисленные в столбце результатов, в лист приложения.

В этом примере для каждой даты счета в таблице имя недели создается на основе года, в котором находится эта неделя, и связанного с ней номера недели, смещенного на одну неделю путем указания для элемента period_no значения 1.

```
TempTable:
LOAD RecNo() as InvID, * Inline [
InvDate
28/03/2012
```

```
10/12/2012
5/2/2013
31/3/2013
19/5/2013
15/9/2013
11/12/2013
2/3/2014
14/5/2014
13/6/2014
7/7/2014
4/8/2014
];

InvoiceData:
LOAD *,
WeekName(InvDate, 1) AS WkName
Resident TempTable;
Drop table TempTable;
```

Результирующая таблица содержит исходные даты и столбец с возвращенным значением функции weekname(). Чтобы отобразить полную метку времени, укажите форматирование на панели свойств.

Результирующая таблица

InvDate	WkName
28/03/2012	2012/14
10/12/2012	2012/51
5/2/2013	2013/07
31/3/2013	2013/14
19/5/2013	2013/21
15/9/2013	2013/38
11/12/2013	2013/51
2/3/2014	2014/10
14/5/2014	2014/21
13/6/2014	2014/25
7/7/2014	2014/29
4/8/2014	2014/33

weekstart

Эта функция возвращает значение, соответствующее метке времени, включающей первую миллисекунду первого дня (понедельника) календарной недели, содержащего значение, указанное в поле **date**. По умолчанию для вывода используется формат **DateFormat**, установленный в скрипте.

Синтаксис:

```
WeekStart(date [, period_no[, first_week_day]])
```

Возвращаемые типы данных: двойное значение

Аргументы:

Аргументы

Аргумент	Описание
date	Дата для вычисления.
period_no	**shift** – целое число, где 0 обозначает неделю, включающую значение, указанное в поле **date**. Отрицательные значения, заданные в поле shift, означают предшествующие недели, положительные – последующие.
first_week_day	Указывает день начала недели. Если не указано, используется значение переменной **FirstWeekDay**. Возможные значения **first_week_day**: • 0 для понедельника • 1 для вторника • 2 для среды • 3 для четверга • 4 для пятницы • 5 для субботы • 6 для воскресенья *FirstWeekDay (page 162)*

Примеры и результаты:
В этих примерах используется формат даты **DD/MM/YYYY**. Формат даты указан в операторе **SET DateFormat** в верхней части скрипта загрузки данных. Измените формат в примерах согласно своим пожеланиям.

Примеры написания скриптов

Пример	Результат
`weekstart('12/01/2013')`	Возвращает 07/01/2013.
`weekstart('12/01/2013', -1 )`	Возвращает 31/11/2012.
`weekstart('12/01/2013', 0, 1)`	Возвращает 08/01/2013.

Пример:

Добавьте образец скрипта в свое приложение и запустите. Чтобы увидеть результаты, добавьте поля, перечисленные в столбце результатов, в лист приложения.

В этом примере выполняется обнаружение первого дня недели после недели с датами каждого счета в таблице.

```
TempTable:
LOAD RecNo() as InvID, * Inline [
InvDate
28/03/2012
10/12/2012
5/2/2013
31/3/2013
19/5/2013
15/9/2013
11/12/2013
2/3/2014
14/5/2014
13/6/2014
7/7/2014
4/8/2014
];

InvoiceData:
LOAD *,
WeekStart(InvDate, 1) AS WkStart
Resident TempTable;
Drop table TempTable;
```

Результирующая таблица содержит исходные даты и столбец с возвращенным значением функции weekstart(). Чтобы отобразить полную метку времени, укажите форматирование на панели свойств.

Результирующая таблица

InvDate	WkStart
28/03/2012	02/04/2012
10/12/2012	17/12/2012
5/2/2013	11/02/2013
31/3/2013	01/04/2013
19/5/2013	20/05/2013
15/9/2013	16/09/2013
11/12/2013	16/12/2013
2/3/2014	03/03/2014
14/5/2014	19/05/2014
13/6/2014	16/06/2014
7/7/2014	14/07/2014
4/8/2014	11/08/2014

weekyear

Эта функция возвращает год, которому принадлежит номер недели согласно стандарту ISO 8601. Номер недели в году может быть установлен в пределах от 1 до 52.

Синтаксис:

```
weekyear(expression)
```

Возвращаемые типы данных: целое

Примеры и результаты:

Примеры написания скриптов

Пример	Результат
`weekyear( '1996-12-30' )`	возвращает 1997, поскольку неделя 1 1997 года начинается 1996-12-30
`weekyear( '1997-01-02' )`	возвращает 1997
`weekyear( '1997-12-28' )`	возвращает 1997
`weekyear( '1997-12-30' )`	возвращает 1998, поскольку неделя 1 1998 года начинается 1997-12-29
`weekyear( '1999-01-02' )`	возвращает 1998, поскольку неделя 53 1998 года оканчивается 1999-01-03

Ограничения:

В определенные годы первая неделя начинается в декабре, например, в декабре 1997 года. В другие годы первая неделя начинается с 53-й недели предыдущего года, как, например, в январе 1999 года. В течение этих нескольких дней, когда номер недели текущего года относится к предыдущему году, функции **year** и **weekyear** возвращают разные значения.

year

Эта функция возвращает год в виде целого числа, а выражение **expression** интерпретируется как дата согласно стандартной интерпретации чисел.

Синтаксис:

```
year(expression)
```

Возвращаемые типы данных: целое

Примеры и результаты:

Примеры написания скриптов

Пример	Результат
`year( '2012-10-12' )`	возвращает 2012
`year( '35648' )`	возвращает 1997, так как 35648 = 1997-08-06

yearend

Эта функция возвращает значение, соответствующее метке времени, включающей последнюю миллисекунду последнего дня года, содержащего значение, указанное в поле **date**. По умолчанию для вывода используется формат **DateFormat**, установленный в скрипте.

Синтаксис:

```
YearEnd( date[, period_no[, first_month_of_year = 1]])
```

Возвращаемые типы данных: двойное значение

Аргументы:

Аргументы

Аргумент	Описание
date	Дата для вычисления.
period_no	**period_no** – целое число, где 0 обозначает год, включающий значение, указанное в поле **date**. Отрицательные значения, заданные в поле **period_no**, означают предшествующие годы, положительные – последующие.
first_ month_of_ year	Если необходимо работать с годами (финансовыми), которые начинаются не в январе, задайте значение от 2 до 12 в поле **first_month_of_year**.

Примеры и результаты:

В этих примерах используется формат даты **DD/MM/YYYY**. Формат даты указан в операторе **SET DateFormat** в верхней части скрипта загрузки данных. Измените формат в примерах согласно своим пожеланиям.

Примеры написания скриптов

Пример	Результат
`yearend ( '19/10/2001' )`	Возвращает 31/12/2001 23:59:59.

Пример	Результат
yearend ('19/10/2001', -1)	Возвращает 31/12/2000 23:59:59.
yearend ('19/10/2001', 0, 4)	Возвращает 31/03/2002 23:59:59.

Пример:

Добавьте образец скрипта в свое приложение и запустите. Чтобы увидеть результаты, добавьте поля, перечисленные в столбце результатов, в лист приложения.

В этом примере выполняется обнаружение последнего дня в году каждой даты счета в таблице, где первый месяц в году указан как месяц 4.

```
TempTable:
LOAD RecNo() as InvID, * Inline [
InvDate
28/03/2012
10/12/2012
5/2/2013
31/3/2013
19/5/2013
15/9/2013
11/12/2013
2/3/2014
14/5/2014
13/6/2014
7/7/2014
4/8/2014
];

InvoiceData:
LOAD *,
YearEnd(InvDate, 0, 4) AS YrEnd
Resident TempTable;
Drop table TempTable;
```

Результирующая таблица содержит исходные даты и столбец с возвращенным значением функции yearend(). Чтобы отобразить полную метку времени, укажите форматирование на панели свойств.

Результирующая таблица

InvDate	YrEnd
28/03/2012	31/03/2011
10/12/2012	31/03/2012
5/2/2013	31/03/2013
31/3/2013	31/03/2013
19/5/2013	31/03/2014

15/9/2013	31/03/2014
11/12/2013	31/03/2014
2/3/2014	31/03/2014
14/5/2014	31/03/2015
13/6/2014	31/03/2015
7/7/2014	31/03/2015
4/8/2014	31/03/2015

yearname

Эта функция возвращает 4-значное значение года с базовым числовым значением, соответствующим метке времени с первой миллисекундой первого дня года, содержащего значение, указанное в поле **date**.

Синтаксис:

```
YearName(date[, period_no[, first_month_of_year]] )
```

Возвращаемые типы данных: двойное значение

Аргументы:

Аргумент	Описание
date	Дата для вычисления.
period_no	**period_no** – целое число, где 0 обозначает год, включающий значение, указанное в поле **date**. Отрицательные значения, заданные в поле **period_no**, означают предшествующие годы, положительные – последующие.
first_month_of_year	Если необходимо работать с годами (финансовыми), которые начинаются не в январе, задайте значение от 2 до 12 в поле **first_month_of_year**. Отображаемое значение будет строчным, показывающим два года.

Примеры и результаты:
В этих примерах используется формат даты **DD/MM/YYYY**. Формат даты указан в операторе **SET DateFormat** в верхней части скрипта загрузки данных. Измените формат в примерах согласно своим пожеланиям.

Примеры скриптов

Пример	Результат
`yearname ( '19/10/2001')`	Возвращает 2001.

Пример	Результат
`yearname ( '19/10/2001', -1 )`	Возвращает 2000.
`yearname ( '19/10/2001', 0, 4)`	Возвращает 2001-2002.

Пример:

Добавьте образец скрипта в свое приложение и запустите. Чтобы увидеть результаты, добавьте поля, перечисленные в столбце результатов, в лист приложения.

В этом примере выполняется обнаружение первого дня в году каждой даты счета в таблице, где первый месяц в году указан как месяц 4.

В этом примере создается цифровое имя четыре-плюс-четыре для годов, в которых обнаруживается каждая дата счета в таблице. Это является следствием того, что первый месяц в году указан как месяц 4.

```
TempTable:
LOAD RecNo() as InvID, * Inline [
InvDate
28/03/2012
10/12/2012
5/2/2013
31/3/2013
19/5/2013
15/9/2013
11/12/2013
2/3/2014
14/5/2014
13/6/2014
7/7/2014
4/8/2014
];

InvoiceData:
LOAD *,
YearName(InvDate, 0, 4) AS YrName
Resident TempTable;
Drop table TempTable;
```

Результирующая таблица содержит исходные даты и столбец с возвращенным значением функции yearname().

Результирующая таблица

InvDate	YrName
28/03/2012	2011-2012
10/12/2012	2012-2013
5/2/2013	2012-2013

InvDate	YrName
31/3/2013	2012-2013
19/5/2013	2013-2014
15/9/2013	2013-2014
11/12/2013	2013-2014
2/3/2014	2013-2014
14/5/2014	2014-2015
13/6/2014	2014-2015
7/7/2014	2014-2015
4/8/2014	2014-2015

yearstart

Эта функция возвращает метку времени, соответствующую началу первого дня года, содержащего значение **date**. По умолчанию для вывода используется формат **DateFormat**, установленный в скрипте.

Синтаксис:

```
YearStart(date[, period_no[, first_month_of_year]])
```

Возвращаемые типы данных: двойное значение

Аргументы:

Аргументы

Аргумент	Описание
date	Дата для вычисления.
period_no	**period_no** – целое число, где 0 обозначает год, включающий значение, указанное в поле **date**. Отрицательные значения, заданные в поле **period_no**, означают предшествующие годы, положительные – последующие.
first_ month_of_ year	Если необходимо работать с годами (финансовыми), которые начинаются не в январе, задайте значение от 2 до 12 в поле **first_month_of_year**.

Примеры и результаты:

В этих примерах используется формат даты **DD/MM/YYYY**. Формат даты указан в операторе **SET DateFormat** в верхней части скрипта загрузки данных. Измените формат в примерах согласно своим пожеланиям.

Примеры написания скриптов

Пример	Результат
`yearstart ('19/10/2001')`	Возвращает 01/01/2001.
`yearstart ('19/10/2001', -1)`	Возвращает 01/01/2000.
`yearstart ('19/10/2001', 0, 4)`	Возвращает 01/04/2001.

Пример:

Добавьте образец скрипта в свое приложение и запустите. Чтобы увидеть результаты, добавьте поля, перечисленные в столбце результатов, в лист приложения.

В этом примере выполняется обнаружение первого дня в году каждой даты счета в таблице, где первый месяц в году указан как месяц 4.

```
TempTable:
LOAD RecNo() as InvID, * Inline [
InvDate
28/03/2012
10/12/2012
5/2/2013
31/3/2013
19/5/2013
15/9/2013
11/12/2013
2/3/2014
14/5/2014
13/6/2014
7/7/2014
4/8/2014
];

InvoiceData:
LOAD *,
YearStart(InvDate, 0, 4) AS YrStart
Resident TempTable;
Drop table TempTable;
```

Результирующая таблица содержит исходные даты и столбец с возвращенным значением функции yearstart(). Чтобы отобразить полную метку времени, укажите форматирование на панели свойств.

Результирующая таблица

InvDate	YrStart
28/03/2012	01/04/2011
10/12/2012	01/04/2012
5/2/2013	01/04/2012
31/3/2013	01/04/2012

InvDate	YrStart
19/5/2013	01/04/2013
15/9/2013	01/04/2013
11/12/2013	01/04/2013
2/3/2014	01/04/2013
14/5/2014	01/04/2014
13/6/2014	01/04/2014
7/7/2014	01/04/2014
4/8/2014	01/04/2014

yeartodate

Эта функция определяет, находится ли введенная метка времени в том году, в котором находится дата последней загрузки скрипта, и возвращает значение True, если это так, и False если это не так.

Синтаксис:

```
YearToDate(timestamp[ , yearoffset [ , firstmonth [ , todaydate] ] ])
```

Возвращаемые типы данных: Булево значение

Если дополнительные параметры не используются, то значение данной функции может быть любой датой в пределах одного календарного года с 1 января до даты последнего выполнения скрипта включительно.

Аргументы:

Аргументы

Аргумент	Описание
timestamp	Метка времени, подлежащая оценке, например '2012-10-12'.
yearoffset	При указании элемента **yearoffset**, элемент **yeartodate** возвращает значение True для того же периода в другом году. Отрицательное значение смещения **yearoffset** указывает предыдущий год, положительное значение смещения – будущий год. Наиболее поздняя дата с начала года до последнего момента достигается путем указания yearoffset = -1. Если значение не указано, принимается 0.
firstmonth	Если в поле **firstmonth** задать значение от 1 до 12 (1, если значение не указано), то начало года может быть передвинуто вперед на первый день любого месяца. Если, например, необходимо работать в рамках финансового года, начинающегося 1 мая, задайте **firstmonth** = 5.

Аргумент	Описание
todaydate	Задав значение **todaydate** (метка времени последнего выполнения скрипта, если не указано), можно сместить день, используемый в качестве верхней границы периода.

Примеры и результаты:
В следующих примерах предполагается время последней перезагрузки = 2011-11-18

Примеры скриптов

Пример	Результат
`yeartodate( '2010-11-18')`	возвращает False
`yeartodate( '2011-02-01')`	возвращает True
`yeartodate( '2011-11-18')`	возвращает True
`yeartodate( '2011-11-19')`	возвращает False
`yeartodate( '2011-11-19', 0, 1, '2011-12-31')`	возвращает True
`yeartodate( '2010-11-18', -1)`	возвращает True
`yeartodate( '2011-11-18', -1)`	возвращает False
`yeartodate( '2011-04-30', 0, 5)`	возвращает False
`yeartodate( '2011-05-01', 0, 5)`	возвращаетTrue

5.8 Экспоненциальные и логарифмические функции

В этом разделе описаны функции, которые относятся к экспоненциальным и логарифмическим вычислениям. Все функции можно использовать как в скрипте загрузки данных, так и в выражениях диаграмм.

Параметры в приведенных ниже функциях – это выражения, в которых переменные **x** и **y** должны интерпретироваться как действительные числа.

exp

Натуральная экспоненциальная функция, e^x, использующая натуральный логарифм **e** в качестве основы. Результат – положительное число.

```
exp(x )
```

Примеры и результаты:

Элемент `exp(3)` возвращает 20,085.

log

Натуральный логарифм числа **x**. Функция определена, только если **x**> 0. Результат – число.

```
log(x )
```

Примеры и результаты:

Элемент `log(3)` возвращает 1,0986

log10
Десятичный логарифм (с основанием 10) числа **x**. Функция определена, только если **x**> 0. Результат – число.

```
log10(x )
```

Примеры и результаты:

`log10(3)` возвращает 0,4771

pow
Возвращает **x** в степени **y**. Результат – число.

```
pow(x,y )
```

Примеры и результаты:

`pow(3, 3)` возвращает 27

sqr
Возвращает **x** в квадрате (**x** в степени 2). Результат – число.

```
sqr (x )
```

Примеры и результаты:

`sqr(3)` возвращает 9

sqrt
Квадратный корень из **x**. Функция определена, только если **x** >= 0. Результат – положительное число.

```
sqrt(x )
```

Примеры и результаты:

`sqrt(3)` возвращает 1,732

5.9 Функции поля

Эти функции могут использоваться только в выражениях диаграмм.

Функции полей возвращают целые числа или строки, выявляя различные аспекты выборок поля.

Функции счетчика

GetAlternativeCount

GetAlternativeCount() используется для обнаружения альтернативных (светло-серых) значений в указанном поле.

```
GetAlternativeCount — функция диаграммы (field_name)
```

GetExcludedCount

GetExcludedCount() находит количество уникальных исключенных (темно-серых) значений в указанном поле. К числу исключенных значений относятся следующие поля: альтернативные (светло-серые), исключенные (темно-серые) и выбранные исключенные (темно-серые с отметкой).

```
GetExcludedCount — функция диаграммы (page 605)(field_name)
```

GetNotSelectedCount

Эта функция диаграммы возвращает число невыбранных значений в поле с именем **fieldname**. Для применимости этой функции поле должно находиться в режиме логического «И».

```
GetNotSelectedCount — функция диаграммы(fieldname [, includeexcluded=false])
```

GetPossibleCount

GetPossibleCount() используется для обнаружения количества возможных значений в указанном поле. Если указанное поле включает выборки, то выбранные (зеленые) поля учитываются. В противном случае учитываются связанные (белые) значения.

```
GetPossibleCount — функция диаграммы(field_name)
```

GetSelectedCount

GetSelectedCount() находит выбранные (зеленые) значения в поле.

```
GetSelectedCount — функция диаграммы (field_name [, include_excluded])
```

Функции поля и выборки

GetCurrentSelections

GetCurrentSelections() возвращает список текущих выборок в приложении. Если выборки были выполнены при помощи строки поиска в окне поиска, **GetCurrentSelections()** возвращает строку поиска.

```
GetCurrentSelections — функция диаграммы([record_sep [,tag_sep [,value_sep
[,max_values]]]])
```

GetFieldSelections

Функция **GetFieldSelections()** возвращает **строку** с текущими выборками в поле.

```
GetFieldSelections — функция диаграммы ( field_name [, value_sep [, max_
values]])
```

GetObjectDimension

GetObjectDimension() возвращает имя измерения. **Index** – дополнительное целое число, обозначающее измерение, которое необходимо вернуть.

```
GetObjectDimension – функция диаграммы ([index])
```

GetObjectField

GetObjectField() возвращает имя измерения. **Index** – дополнительное целое число, обозначающее измерение, которое необходимо вернуть.

```
GetObjectField – функция диаграммы ([index])
```

GetObjectMeasure

GetObjectMeasure() возвращает имя меры. **Index** – дополнительное целое число, обозначающее меру, которую необходимо вернуть.

```
GetObjectMeasure – функция диаграммы ([index])
```

GetAlternativeCount – функция диаграммы

GetAlternativeCount() используется для обнаружения альтернативных (светло-серых) значений в указанном поле.

Синтаксис:

```
GetAlternativeCount (field_name)
```

Возвращаемые типы данных: целое

Аргументы:

Аргументы

Аргумент	Описание
field_name	Поле, содержащее диапазон данных для измерения.

Примеры и результаты:

В следующем примере используется значение поля **First name**, загруженное в фильтр.

Примеры и результаты

Примеры	Результаты
При условии, что элемент **John** выбран в элементе **First name**. `GetAlternativeCount ([First name])`	Значение 4, поскольку существует 4 уникальных и исключенных (серых) значения в элементе **First name**.

Примеры	Результаты
При условии выбора элементов **John** и **Peter**. `GetAlternativeCount ([First name])`	Значение 3, поскольку существует 3 уникальных и исключенных (серых) значения в элементе **First name**.
При условии, что в элементе **First name** значения не выбраны. `GetAlternativeCount ([First name])`	Значение 0, поскольку выборок нет.

Данные, используемые в примере:

```
Names:
LOAD * inline [
First name|Last name|Initials|Has cellphone
John|Anderson|JA|Yes
Sue|Brown|SB|Yes
Mark|Carr|MC|No
Peter|Devonshire|PD|No
Jane|Elliot|JE|Yes
Peter|Franc|PF|Yes ] (delimiter is '|');
```

GetCurrentSelections – функция диаграммы

GetCurrentSelections() возвращает список текущих выборок в приложении. Если выборки были выполнены при помощи строки поиска в окне поиска, **GetCurrentSelections()** возвращает строку поиска.

Если параметры используются, необходимо указать record_sep. Чтобы указать новый размер строки, установите для параметра **record_sep** значение **chr(13)&chr(10)**.

Если выбраны все значения, кроме двух или одного значения, будет использован формат «NOT x,y» или «NOT y» соответственно. Если выбраны все значения, и число всех значений больше, чем max_values, будет возвращен текст ALL.

Синтаксис:

```
GetCurrentSelections ([record_sep [, tag_sep [, value_sep [, max_values [, state_name]]]]])
```

Возвращаемые типы данных: строка

Аргументы:

Аргументы

Аргументы	Описание
record_sep	Разделитель должен стоять между записями в поле. Значение по умолчанию <CR><LF> означает новую строку.
tag_sep	Разделитель должен стоять между тегом имени поля и значениями поля. По умолчанию используется «: ».
value_sep	Разделитель значений в поле. По умолчанию используется «, ».
max_values	Максимальное число отдельно отображаемых значений, введенных в поле. При вводе большого числа значений используется формат «x из y значений». По умолчанию установлено 6.
state_name	Имя альтернативного состояния, выбранное для определенной визуализации. Если используется аргумент **state_name**, учитываются только выборки, связанные с указанным именем состояния.

Примеры и результаты:

В следующем примере используются два поля, загруженные в разные поля фильтра, одно для имени **First name**, а второе для **Initials**.

Примеры и результаты

Примеры	Результаты
При условии, что элемент **John** выбран в элементе **First name**. `GetCurrentSelections ()`	'First name: John'
При условии выбора элементов **John** и **Peter** в элементе **First name**. `GetCurrentSelections ()`	'First name: John, Peter'
При условии выбора элементов **John** и **Peter** в элементе **First name** и выбора элемента **JA** в элементе **Initials**. `GetCurrentSelections ()`	'First name: John, Peter Initials: JA'
При условии выбора элемента **John** в элементе **First name**, а **JA** в элементе **Initials**. `GetCurrentSelections ( chr(13)&chr(10) , ' = ' )`	'First name = John Initials = JA'
При условии выбора всех имен, кроме Sue, в элементе **First name** и отсутствии выборок в элементе **Initials**. `GetCurrentSelections (chr(13)&chr(10), '=', ',' ,3)`	'First name=NOT Sue'

Данные, используемые в примере:

```
Names:
LOAD * inline [
First name|Last name|Initials|Has cellphone
John|Anderson|JA|Yes
Sue|Brown|SB|Yes
Mark|Carr|MC|No
Peter|Devonshire|PD|No
Jane|Elliot|JE|Yes
Peter|Franc|PF|Yes ] (delimiter is '|');
```

GetExcludedCount – функция диаграммы

GetExcludedCount() находит количество уникальных исключенных (темно-серых) значений в указанном поле. К числу исключенных значений относятся следующие поля: альтернативные (светло-серые), исключенные (темно-серые) и выбранные исключенные (темно-серые с отметкой).

Синтаксис:

```
GetExcludedCount (field_name)
```

Возвращаемые типы данных: строка

Аргументы:

Аргументы

Аргументы	Описание
field_name	Поле, содержащее диапазон данных для измерения.

Примеры и результаты:

В следующем примере используется три поля, загруженных в разные фильтры: одно для элемента **First name**, второе для элемента **Last name**, а третье для элемента **Initials**.

Примеры и результаты

Примеры	Результаты
При условии, что в элементе **First name** значения не выбраны.	`GetExcludedCount (Initials) = 0` Выборки отсутствуют.
При условии, что элемент **John** выбран в элементе **First name**.	`GetExcludedCount (Initials) = 5` Поле **Initials** содержит 5 исключенных значений, отображающихся темно-серым цветом. Шестая ячейка (JA) будет белой, поскольку она связана с выборкой John в элементе **First name**.
При условии, что выбраны элементы **John** и **Peter**.	`GetExcludedCount (Initials) = 3` Элемент John связан с 1 значением, элемент Peter связан с 2 значениями в элементе **Initials**.

Примеры	Результаты
При условии, что выбраны элементы **John** и **Peter** в элементе **First name**, а затем выбран элемент **Franc** в элементе **Last name**.	`GetExcludedCount ([First name]) = 4` Элемент **First name** содержит 4 исключенных значения, отображающихся темно-серым цветом. Элемент **GetExcludedCount()** оценивает поля с исключенными значениями, в том числе поля с альтернативными и выбранными исключенными значениями.
При условии, что выбраны элементы **John** и **Peter** в элементе **First name**, а затем выбраны элементы **Franc** и **Anderson** в элементе **Last name**.	`GetExcludedCount (Initials) = 4` Элемент **Initials** содержит 4 исключенных значения, отображающихся темно-серым цветом. Две другие ячейки (JA и PF) будут белыми, поскольку они связаны с выборками John и Peter в элементе **First name**.
При условии, что выбраны элементы **John** и **Peter** в элементе **First name**, а затем выбраны элементы **Franc** и **Anderson** в элементе **Last name**.	`GetExcludedCount ([Last name]) = 4` Элемент **Initials** содержит 4 исключенных значения. Элемент Devonshire отображается светло-серым цветом, элементы Brown, Carr и Elliot отображаются темно-серым цветом.

Данные, используемые в примере:

```
Names:
LOAD * inline [
First name|Last name|Initials|Has cellphone
John|Anderson|JA|Yes
Sue|Brown|SB|Yes
Mark|Carr|MC|No
Peter|Devonshire|PD|No
Jane|Elliot|JE|Yes
Peter|Franc|PF|Yes ] (delimiter is '|');
```

GetFieldSelections – функция диаграммы

Функция **GetFieldSelections()** возвращает **строку** с текущими выборками в поле.

Если выбраны все значения, кроме двух или одного значения, будет использован формат «NOT x,y» или «NOT y» соответственно. Если выбраны все значения и число всех значений больше, чем max_values, будет возвращен текст ALL.

Синтаксис:

```
GetFieldSelections ( field_name [, value_sep [, max_values [, state_name]]])
```

Возвращаемые типы данных: строка

Форматы возвращаемой строки

Формат	Описание
'a, b, c'	Количество выбранных значений – max_values или меньше, возвращенная строка представляет собой список выбранных значений. В качестве разделителя значений используется value_sep.
'NOT a, b, c'	Количество невыбранных значений – max_values или меньше, возвращенная строка представляет собой список невыбранных значений с префиксом NOT. В качестве разделителя значений используется value_sep.
'x of y'	x = количество выбранных значений y = общее количество значений Этот результат возвращается, когда max_values < x < (y - max_values).
'ALL'	Возвращается, если выбраны все значения.
'-'	Возвращается, если значения не выбраны.
<search string>	Если выборка выполнена с помощью поиска, возвращается строка поиска.

Аргументы:

Аргументы

Аргументы	Описание
field_name	Поле, содержащее диапазон данных для измерения.
value_sep	Разделитель значений в поле. По умолчанию используется «, ».
max_values	Максимальное число отдельно отображаемых значений, введенных в поле. При вводе большого числа значений используется формат «x из y значений». По умолчанию установлено 6.
state_name	Имя альтернативного состояния, выбранное для определенной визуализации. Если используется аргумент **state_name**, учитываются только выборки, связанные с указанным именем состояния.

Примеры и результаты:

В следующем примере используется значение поля **First name**, загруженное в фильтр.

Примеры и результаты

Примеры	Результаты
При условии, что элемент **John** выбран в элементе **First name**. `GetFieldSelections ([First name])`	«John»
При условии выбора элементов **John** и **Peter**. `GetFieldSelections ([First name])`	«John,Peter»
При условии выбора элементов **John** и **Peter**. `GetFieldSelections ([First name],'; ')`	«John; Peter»
При условии выбора элементов **John**, **Sue**, **Mark** в элементе **First name**. `GetFieldSelections ([First name],';',2)`	«NOT Jane;Peter», поскольку значение 2 указано, как значение аргумента max_values. В противном случае результат был бы John; Sue; Mark.

Данные, используемые в примере:

```
Names:
LOAD * inline [
First name|Last name|Initials|Has cellphone
John|Anderson|JA|Yes
Sue|Brown|SB|Yes
Mark|Carr|MC|No
Peter|Devonshire|PD|No
Jane|Elliot|JE|Yes
Peter|Franc|PF|Yes ] (delimiter is '|');
```

GetNotSelectedCount – функция диаграммы

Эта функция диаграммы возвращает число невыбранных значений в поле с именем **fieldname**. Для применимости этой функции поле должно находиться в режиме логического «И».

Синтаксис:

```
GetNotSelectedCount(fieldname [, includeexcluded=false])
```

Аргументы:

Аргументы

Аргумент	Описание
fieldname	Имя поля для оценки.
includeexcluded	Если для элемента **includeexcluded** установлено значение True, число будет включать в себя выбранные значения, исключенные выборками в другом поле.

Примеры:

```
GetNotSelectedCount( Country )
GetNotSelectedCount( Country, true )
```

GetObjectDimension – функция диаграммы

GetObjectDimension() возвращает имя измерения. **Index** – дополнительное целое число, обозначающее измерение, которое необходимо вернуть.

Данная функция не поддерживается в следующих расположениях диаграммы: заголовок, подзаголовок, нижний колонтитул, выражение опорной линии.

Создание ссылок на имя измерения или меры в другом объекте при помощи Object ID не поддерживается.

Синтаксис:

```
GetObjectDimension ([index])
```

Пример:

```
GetObjectDimension(1)
```

Пример: Выражение диаграммы

В таблице Qlik Sense представлены примеры использования функции GetObjectDimension в выражении диаграммы

transaction_date	customer_id	transaction_quantity	=GetObjectDimension ()	=GetObjectDimension (0)	=GetObjectDimension (1)
2018/08/30	049681	13	transaction_date	transaction_date	customer_id
2018/08/3	203521	6	transaction_date	transaction_date	customer_id

transaction_date	customer_id	transaction_quantity	=GetObjectDimension ()	=GetObjectDimension (0)	=GetObjectDimension (1)
0					
2018/08/30	203521	21	transaction_date	transaction_date	customer_id

Для возврата имени меры используйте функцию **GetObjectMeasure**.

GetObjectField – функция диаграммы

GetObjectField() возвращает имя измерения. **Index** – дополнительное целое число, обозначающее измерение, которое необходимо вернуть.

Данная функция не поддерживается в следующих расположениях диаграммы: заголовок, подзаголовок, нижний колонтитул, выражение опорной линии.

Создание ссылок на имя измерения или меры в другом объекте при помощи Object ID не поддерживается.

Синтаксис:

```
GetObjectField ([index])
```

Пример:

```
GetObjectField(1)
```

Пример: Выражение диаграммы

В таблице Qlik Sense представлены примеры использования функции GetObjectField в выражении диаграммы.

transaction_date	customer_id	transaction_quantity	=GetObjectField ()	=GetObjectField (0)	=GetObjectField (1)
2018/08/30	049681	13	transaction_date	transaction_date	customer_id
2018/08/30	203521	6	transaction_date	transaction_date	customer_id
2018/08/30	203521	21	transaction_date	transaction_date	customer_id

Для возврата имени меры используйте функцию **GetObjectMeasure**.

GetObjectMeasure – функция диаграммы

GetObjectMeasure() возвращает имя меры. **Index** – дополнительное целое число, обозначающее меру, которую необходимо вернуть.

Данная функция не поддерживается в следующих расположениях диаграммы: заголовок, подзаголовок, нижний колонтитул, выражение опорной линии.

Создание ссылок на имя измерения или меры в другом объекте при помощи Object ID не поддерживается.

Синтаксис:

```
GetObjectMeasure ([index])
```

Пример:

```
GetObjectMeasure(1)
```

Пример: Выражение диаграммы

В таблице Qlik Sense представлены примеры использования функции GetObjectMeasure в выражении диаграммы

customer_id	sum (transaction_quantity)	Avg (transaction_quantity)	=GetObjectMeasure ()	=GetObjectMeasure(0)	=GetObjectMeasure(1)
49681	13	13	sum(transaction_quantity)	sum(transaction_quantity)	Avg(transaction_quantity)
203521	27	13.5	sum(transaction_quantity)	sum(transaction_quantity)	Avg(transaction_quantity)

Для возврата имени измерения используйте функцию **GetObjectField**.

GetPossibleCount – функция диаграммы

GetPossibleCount() используется для обнаружения количества возможных значений в указанном поле. Если указанное поле включает выборки, то выбранные (зеленые) поля учитываются. В противном случае учитываются связанные (белые) значения. .

Для полей с выборками функция **GetPossibleCount()** возвращает число выбранных (зеленых) полей.

Возвращаемые типы данных: целое

Синтаксис:

```
GetPossibleCount (field_name)
```

Аргументы:

Аргументы

Аргументы	Описание
field_name	Поле, содержащее диапазон данных для измерения.

Примеры и результаты:

В следующем примере используются два поля, загруженные в разные поля фильтра, одно для имени **First name**, а второе для **Initials**.

Примеры и результаты

Примеры	Результаты
При условии, что элемент **John** выбран в элементе **First name**. `GetPossibleCount ([Initials])`	Значение 1, поскольку в элементе «Инициалы» значение 1 связано с выборкой, элементом **John**, в элементе **First name**.
При условии, что элемент **John** выбран в элементе **First name**. `GetPossibleCount ([First name])`	Значение 1, поскольку существует 1 выборка, элемент **John**, в элементе **First name**.
При условии, что элемент **Peter** выбран в элементе **First name**. `GetPossibleCount ([Initials])`	Значение 2, поскольку элемент Peter связан с 2 значениями в элементе **Initials**.
При условии, что в элементе **First name** значения не выбраны. `GetPossibleCount ([First name])`	Значение 5, поскольку выборок нет, но есть 5 уникальных значений в элементе **First name**.
При условии, что в элементе **First name** значения не выбраны. `GetPossibleCount ([Initials])`	Значение 6, поскольку выборок нет, но есть 6 уникальных значений в элементе **Initials**.

Данные, используемые в примере:

```
Names:
LOAD * inline [
First name|Last name|Initials|Has cellphone
John|Anderson|JA|Yes
Sue|Brown|SB|Yes
Mark|Carr|MC|No
Peter|Devonshire|PD|No
```

```
Jane|Elliot|JE|Yes
Peter|Franc|PF|Yes ] (delimiter is '|');
```

GetSelectedCount – функция диаграммы

GetSelectedCount() находит выбранные (зеленые) значения в поле.

Синтаксис:

```
GetSelectedCount (field_name [, include_excluded [, state_name]])
```

Возвращаемые типы данных: целое

Аргументы:

Аргументы

Аргументы	Описание
field_name	Поле, содержащее диапазон данных для измерения.
include_ excluded	Если установлено значение **True()**, при подсчете будут учитываться выбранные значения, которые в настоящее время исключаются выборками в других полях. Если значения являются False или опущены, эти значения не будут включены.
state_name	Имя альтернативного состояния, выбранное для определенной визуализации. Если используется аргумент **state_name**, учитываются только выборки, связанные с указанным именем состояния.

Примеры и результаты:

В следующем примере используется три поля, загруженных в разные фильтры: одно для элемента **First name**, второе для элемента **Initials**, а третье для элемента **Has cellphone**.

Примеры и результаты

Примеры	Результаты
При условии, что элемент **John** выбран в элементе **First name**. `GetSelectedCount ([First name])`	Значение 1, поскольку одно значение выбрано в элементе **First name**.
При условии, что элемент **John** выбран в элементе **First name**. `GetSelectedCount ([Initials])`	Значение 0, поскольку в элементе **Initials** значения не выбраны.
При отсутствии выборок в элементе **First name** выберите все значения в элементе **Initials**, а затем выберите значение **Yes** в элементе **Has cellphone**. `GetSelectedCount ([Initials], True())`	6. Хотя при выборках элемента **Initials**MC и PD имеют значение **Has cellphone**, заданное как **No**, результатом все равно будет 6, поскольку для аргумента include_excluded задано значение True().

Данные, используемые в примере:

```
Names:
LOAD * inline [
First name|Last name|Initials|Has cellphone
John|Anderson|JA|Yes
Sue|Brown|SB|Yes
Mark|Carr|MC|No
Peter|Devonshire|PD|No
Jane|Elliot|JE|Yes
Peter|Franc|PF|Yes ] (delimiter is '|');
```

5.10 Функции файлов

Функции файлов (доступны только в выражениях скрипта) возвращают информацию о табличном файле, читаемом в настоящее время. Эти функции возвращают значение NULL для всех источников данных, кроме табличных файлов (исключение: ConnectString( )).

Обзор функций файла

Каждая функция подробно описана после обзора. Также можно щелкнуть имя функции в синтаксисе, чтобы получить немедленный доступ к подробной информации об этой конкретной функции.

Attribute

Эта функция скрипта возвращает значение мета-тегов различных медиафайлов в виде текста. Поддерживаются следующие форматы файлов: MP3, WMA, WMV, PNG и JPG. Если файл **filename** не существует, не является поддерживаемым форматом файла или не содержит мета-тег с именем **attributename**, в таком случае возвращается значение NULL.

```
Attribute (filename, attributename)
```

ConnectString

Функция **ConnectString()** возвращает имя активного подключения к данным для подключений ODBC или OLE DB. Функция возвращает пустую строку, если не выполнен оператор **connect**, или после оператора **disconnect**.

```
ConnectString ()
```

FileBaseName

Функция **FileBaseName** возвращает строку с именем табличного файла, читаемого в текущий момент, без пути или расширения.

```
FileBaseName ()
```

FileDir

Функция **FileDir** возвращает строку, содержащую путь к каталогу табличного файла, читаемого в текущий момент.

```
FileDir ()
```

FileExtension

Функция **FileExtension** возвращает строку, содержащую расширение табличного файла, читаемого в текущий момент.

```
FileExtension ()
```

FileName

Функция **FileName** возвращает строку с именем табличного файла, читаемого в текущий момент, без пути, но с расширением.

```
FileName ()
```

FilePath

Функция **FilePath** возвращает строку, содержащую полный путь табличного файла, читаемого в текущий момент.

```
FilePath ()
```

FileSize

Функция **FileSize** возвращает целое, содержащее размер в байтах файла filename, или, если не указан файл filename, табличного файла, читаемого в текущий момент.

```
FileSize ()
```

FileTime

Функция **FileTime** возвращает метку времени для даты и времени последнего исправления файла filename. Если не указан файл в поле filename, функция ссылается на табличный файл, читаемый в текущий момент.

```
FileTime ([ filename ])
```

GetFolderPath

Функция **GetFolderPath** возвращает значение функции Microsoft Windows *SHGetFolderPath*. Данная функция берет в качестве значения ввода имя папки Microsoft Windows и возвращает полный путь папки.

```
GetFolderPath ()
```

QvdCreateTime

Эта функция скрипта возвращает метку времени верхнего колонтитула XML из файла QVD при его наличии, в противном случае она возвращает значение NULL.

```
QvdCreateTime (filename)
```

QvdFieldName

Эта функция скрипта возвращает имя числа поля **fieldno** в файле QVD. Если поле не существует, возвращается значение NULL.

```
QvdFieldName (filename , fieldno)
```

QvdNoOfFields

Эта функция скрипта возвращает число полей в файле QVD.

```
QvdNoOfFields (filename)
```

QvdNoOfRecords

Эта функция скрипта возвращает число записей, находящихся в настоящее время в файле QVD.

```
QvdNoOfRecords (filename)
```

QvdTableName

Эта функция скрипта возвращает имя таблицы, хранящейся в файле QVD.

```
QvdTableName (filename)
```

Attribute

Эта функция скрипта возвращает значение мета-тегов различных медиафайлов в виде текста. Поддерживаются следующие форматы файлов: MP3, WMA, WMV, PNG и JPG. Если файл **filename** не существует, не является поддерживаемым форматом файла или не содержит мета-тег с именем **attributename**, в таком случае возвращается значение NULL.

Синтаксис:

```
Attribute(filename, attributename)
```

Может быть прочитано большое количество мета-тегов. В этой теме показаны примеры, в которых видно, какие теги могут быть прочитаны для соответствующих поддерживаемых типов файлов.

*Для чтения доступны только мета-теги, сохраненные в файле согласно соответствующей спецификации, например, ID2v3 для файлов MP3 или EXIF для файлов JPG, но не мета-информация, сохраненная в **Windows File Explorer**.*

Аргументы:

Аргументы

Аргумент	Описание
filename	Имя медиафайла, включающее при необходимости путь, в качестве подключения к данным папки. **Пример:** *'lib://Table Files/'* В прежней версии режима написания скриптов следующие форматы пути тоже поддерживаются: • абсолютный **Пример:** *c:\data* • относительно рабочего каталога приложения Qlik Sense. **Пример:** *data*
attributename	Имя мета-тега.

В примерах используется функция **GetFolderPath** для обнаружения путей к медиафайлам. **GetFolderPath** поддерживается только в устаревшем режиме, поэтому, если эта функция используется в стандартном режиме или в Qlik Sense SaaS, вместо ссылок на **GetFolderPath** необходимо указать путь подключения к данным lib://.

Ограничение доступа к файловой системе (page 869)

Example 1: Файлы MP3

Этот скрипт предназначен для чтения всех возможных мета-тегов MP3 в папке *MyMusic*.

```
// Script to read MP3 meta tags for each vExt in 'mp3' for each vFoundFile in filelist(
GetFolderPath('MyMusic') & '\*.'& vExt ) FileList: LOAD FileLongName,     subfield
(FileLongName,'\',-1) as FileShortName,     num(FileSize(FileLongName),'# ### ### ###',',','
') as FileSize,     FileTime(FileLongName) as FileTime,      // ID3v1.0 and ID3v1.1 tags
Attribute(FileLongName, 'Title') as Title,     Attribute(FileLongName, 'Artist') as Artist,
 Attribute(FileLongName, 'Album') as Album,     Attribute(FileLongName, 'Year') as Year,
Attribute(FileLongName, 'Comment') as Comment,     Attribute(FileLongName, 'Track') as Track,
   Attribute(FileLongName, 'Genre') as Genre,

     // ID3v2.3 tags     Attribute(FileLongName, 'AENC') as AENC, // Audio encryption
Attribute(FileLongName, 'APIC') as APIC, // Attached picture     Attribute(FileLongName,
'COMM') as COMM, // Comments     Attribute(FileLongName, 'COMR') as COMR, // Commercial frame
   Attribute(FileLongName, 'ENCR') as ENCR, // Encryption method registration     Attribute
(FileLongName, 'EQUA') as EQUA, // Equalization     Attribute(FileLongName, 'ETCO') as ETCO,
// Event timing codes     Attribute(FileLongName, 'GEOB') as GEOB, // General encapsulated
object     Attribute(FileLongName, 'GRID') as GRID, // Group identification registration
Attribute(FileLongName, 'IPLS') as IPLS, // Involved people list     Attribute(FileLongName,
'LINK') as LINK, // Linked information     Attribute(FileLongName, 'MCDI') as MCDI, // Music
```

```
CD identifier     Attribute(FileLongName, 'MLLT') as MLLT, // MPEG location lookup table
Attribute(FileLongName, 'OWNE') as OWNE, // Ownership frame     Attribute(FileLongName,
'PRIV') as PRIV, // Private frame     Attribute(FileLongName, 'PCNT') as PCNT, // Play counter
    Attribute(FileLongName, 'POPM') as POPM, // Popularimeter

     Attribute(FileLongName, 'POSS') as POSS, // Position synchronisation frame     Attribute
(FileLongName, 'RBUF') as RBUF, // Recommended buffer size     Attribute(FileLongName, 'RVAD')
as RVAD, // Relative volume adjustment     Attribute(FileLongName, 'RVRB') as RVRB, // Reverb
   Attribute(FileLongName, 'SYLT') as SYLT, // Synchronized lyric/text     Attribute
(FileLongName, 'SYTC') as SYTC, // Synchronized tempo codes     Attribute(FileLongName,
'TALB') as TALB, // Album/Movie/Show title     Attribute(FileLongName, 'TBPM') as TBPM, // BPM
(beats per minute)     Attribute(FileLongName, 'TCOM') as TCOM, // Composer     Attribute
(FileLongName, 'TCON') as TCON, // Content type     Attribute(FileLongName, 'TCOP') as TCOP,
// Copyright message     Attribute(FileLongName, 'TDAT') as TDAT, // Date     Attribute
(FileLongName, 'TDLY') as TDLY, // Playlist delay

     Attribute(FileLongName, 'TENC') as TENC, // Encoded by     Attribute(FileLongName,
'TEXT') as TEXT, // Lyricist/Text writer     Attribute(FileLongName, 'TFLT') as TFLT, // File
type     Attribute(FileLongName, 'TIME') as TIME, // Time     Attribute(FileLongName, 'TIT1')
as TIT1, // Content group description     Attribute(FileLongName, 'TIT2') as TIT2, //
Title/songname/content description     Attribute(FileLongName, 'TIT3') as TIT3, //
Subtitle/Description refinement     Attribute(FileLongName, 'TKEY') as TKEY, // Initial key
 Attribute(FileLongName, 'TLAN') as TLAN, // Language(s)     Attribute(FileLongName, 'TLEN')
as TLEN, // Length     Attribute(FileLongName, 'TMED') as TMED, // Media type

     Attribute(FileLongName, 'TOAL') as TOAL, // Original album/movie/show title     Attribute
(FileLongName, 'TOFN') as TOFN, // Original filename     Attribute(FileLongName, 'TOLY') as
TOLY, // Original lyricist(s)/text writer(s)     Attribute(FileLongName, 'TOPE') as TOPE, //
Original artist(s)/performer(s)     Attribute(FileLongName, 'TORY') as TORY, // Original
release year     Attribute(FileLongName, 'TOWN') as TOWN, // File owner/licensee     Attribute
(FileLongName, 'TPE1') as TPE1, // Lead performer(s)/Soloist(s)     Attribute(FileLongName,
'TPE2') as TPE2, // Band/orchestra/accompaniment

     Attribute(FileLongName, 'TPE3') as TPE3, // Conductor/performer refinement     Attribute
(FileLongName, 'TPE4') as TPE4, // Interpreted, remixed, or otherwise modified by
Attribute(FileLongName, 'TPOS') as TPOS, // Part of a set     Attribute(FileLongName, 'TPUB')
as TPUB, // Publisher     Attribute(FileLongName, 'TRCK') as TRCK, // Track number/Position in
set     Attribute(FileLongName, 'TRDA') as TRDA, // Recording dates     Attribute
(FileLongName, 'TRSN') as TRSN, // Internet radio station name     Attribute(FileLongName,
'TRSO') as TRSO, // Internet radio station owner

     Attribute(FileLongName, 'TSIZ') as TSIZ, // Size     Attribute(FileLongName, 'TSRC') as
TSRC, // ISRC (international standard recording code)     Attribute(FileLongName, 'TSSE') as
TSSE, // Software/Hardware and settings used for encoding     Attribute(FileLongName, 'TYER')
as TYER, // Year     Attribute(FileLongName, 'TXXX') as TXXX, // User defined text information
frame     Attribute(FileLongName, 'UFID') as UFID, // Unique file identifier     Attribute
(FileLongName, 'USER') as USER, // Terms of use     Attribute(FileLongName, 'USLT') as USLT,
// Unsychronized lyric/text transcription     Attribute(FileLongName, 'WCOM') as WCOM, //
Commercial information     Attribute(FileLongName, 'WCOP') as WCOP, // Copyright/Legal
information

     Attribute(FileLongName, 'WOAF') as WOAF, // Official audio file webpage     Attribute
(FileLongName, 'WOAR') as WOAR, // Official artist/performer webpage     Attribute
(FileLongName, 'WOAS') as WOAS, // Official audio source webpage     Attribute(FileLongName,
'WORS') as WORS, // Official internet radio station homepage     Attribute(FileLongName,
```

```
'WPAY') as WPAY, // Payment     Attribute(FileLongName, 'WPUB') as WPUB, // Publishers
official webpage     Attribute(FileLongName, 'WXXX') as WXXX; // User defined URL link frame
LOAD @1:n as FileLongName Inline "$(vFoundFile)" (fix, no labels); Next vFoundFile Next vExt
```

Example 2: JPEG

Этот скрипт предназначен для чтения всех возможных мета-тегов EXIF из файлов JPG в папке *MyPictures*.

```
// Script to read Jpeg Exif meta tags for each vExt in 'jpg', 'jpeg', 'jpe', 'jfif', 'jif',
'jfi' for each vFoundFile in filelist( GetFolderPath('MyPictures') & '\*.'& vExt )

 FileList: LOAD FileLongName,     subfield(FileLongName,'\',-1) as FileShortName,     num
(FileSize(FileLongName),'# ### ### ###',',',' ') as FileSize,     FileTime(FileLongName) as
FileTime,      // ************   Exif Main (IFD0) Attributes   ************     Attribute
(FileLongName, 'ImageWidth') as ImageWidth,     Attribute(FileLongName, 'ImageLength') as
ImageLength,     Attribute(FileLongName, 'BitsPerSample') as BitsPerSample,     Attribute
(FileLongName, 'Compression') as Compression,

    //  examples: 1=uncompressed, 2=CCITT, 3=CCITT 3, 4=CCITT 4,

    //5=LZW, 6=JPEG (old style), 7=JPEG, 8=Deflate, 32773=PackBits RLE,      Attribute
(FileLongName, 'PhotometricInterpretation') as PhotometricInterpretation,

    //  examples: 0=WhiteIsZero, 1=BlackIsZero, 2=RGB, 3=Palette, 5=CMYK, 6=YCbCr,
Attribute(FileLongName, 'ImageDescription') as ImageDescription,     Attribute(FileLongName,
'Make') as Make,     Attribute(FileLongName, 'Model') as Model,     Attribute(FileLongName,
'StripOffsets') as StripOffsets,     Attribute(FileLongName, 'Orientation') as Orientation,

    //  examples: 1=TopLeft, 2=TopRight, 3=BottomRight, 4=BottomLeft,

    // 5=LeftTop, 6=RightTop, 7=RightBottom, 8=LeftBottom,      Attribute(FileLongName,
'SamplesPerPixel') as SamplesPerPixel,     Attribute(FileLongName, 'RowsPerStrip') as
RowsPerStrip,     Attribute(FileLongName, 'StripByteCounts') as StripByteCounts,     Attribute
(FileLongName, 'XResolution') as XResolution,     Attribute(FileLongName, 'YResolution') as
YResolution,     Attribute(FileLongName, 'PlanarConfiguration') as PlanarConfiguration,

    //  examples: 1=chunky format, 2=planar format,      Attribute(FileLongName,
'ResolutionUnit') as ResolutionUnit,

    //  examples: 1=none, 2=inches, 3=centimeters,      Attribute(FileLongName,
'TransferFunction') as TransferFunction,     Attribute(FileLongName, 'Software') as Software,
   Attribute(FileLongName, 'DateTime') as DateTime,     Attribute(FileLongName, 'Artist') as
Artist,     Attribute(FileLongName, 'HostComputer') as HostComputer,     Attribute
(FileLongName, 'WhitePoint') as WhitePoint,     Attribute(FileLongName,
'PrimaryChromaticities') as PrimaryChromaticities,     Attribute(FileLongName,
'YCbCrCoefficients') as YCbCrCoefficients,     Attribute(FileLongName, 'YCbCrSubSampling') as
YCbCrSubSampling,     Attribute(FileLongName, 'YCbCrPositioning') as YCbCrPositioning,

    //  examples: 1=centered, 2=co-sited,      Attribute(FileLongName, 'ReferenceBlackWhite')
as ReferenceBlackWhite,     Attribute(FileLongName, 'Rating') as Rating,     Attribute
(FileLongName, 'RatingPercent') as RatingPercent,     Attribute(FileLongName,
'ThumbnailFormat') as ThumbnailFormat,
```

```
    //  examples: 0=Raw Rgb, 1=Jpeg,      Attribute(FileLongName, 'Copyright') as Copyright,
  Attribute(FileLongName, 'ExposureTime') as ExposureTime,     Attribute(FileLongName,
'FNumber') as FNumber,     Attribute(FileLongName, 'ExposureProgram') as ExposureProgram,

    //  examples: 0=Not defined, 1=Manual, 2=Normal program, 3=Aperture priority, 4=Shutter
priority,

    // 5=Creative program, 6=Action program, 7=Portrait mode, 8=Landscape mode, 9=Bulb,
Attribute(FileLongName, 'ISOSpeedRatings') as ISOSpeedRatings,     Attribute(FileLongName,
'TimeZoneOffset') as TimeZoneOffset,     Attribute(FileLongName, 'SensitivityType') as
SensitivityType,

    //  examples: 0=Unknown, 1=Standard output sensitivity (SOS), 2=Recommended exposure index
(REI),

    // 3=ISO speed, 4=Standard output sensitivity (SOS) and Recommended exposure index (REI),

    //5=Standard output sensitivity (SOS) and ISO Speed, 6=Recommended exposure index (REI)
and ISO Speed,

    // 7=Standard output sensitivity (SOS) and Recommended exposure index (REI) and ISO speed,
     Attribute(FileLongName, 'ExifVersion') as ExifVersion,     Attribute(FileLongName,
'DateTimeOriginal') as DateTimeOriginal,     Attribute(FileLongName, 'DateTimeDigitized') as
DateTimeDigitized,     Attribute(FileLongName, 'ComponentsConfiguration') as
ComponentsConfiguration,

    //  examples: 1=Y, 2=Cb, 3=Cr, 4=R, 5=G, 6=B,      Attribute(FileLongName,
'CompressedBitsPerPixel') as CompressedBitsPerPixel,     Attribute(FileLongName,
'ShutterSpeedValue') as ShutterSpeedValue,     Attribute(FileLongName, 'ApertureValue') as
ApertureValue,     Attribute(FileLongName, 'BrightnessValue') as BrightnessValue, //
examples: -1=Unknown,      Attribute(FileLongName, 'ExposureBiasValue') as ExposureBiasValue,
   Attribute(FileLongName, 'MaxApertureValue') as MaxApertureValue,     Attribute
(FileLongName, 'SubjectDistance') as SubjectDistance,

    //  examples: 0=Unknown, -1=Infinity,      Attribute(FileLongName, 'MeteringMode') as
MeteringMode,

    //  examples: 0=Unknown, 1=Average, 2=CenterWeightedAverage, 3=Spot,

    // 4=MultiSpot, 5=Pattern, 6=Partial, 255=Other,      Attribute(FileLongName,
'LightSource') as LightSource,

    //  examples: 0=Unknown, 1=Daylight, 2=Fluorescent, 3=Tungsten, 4=Flash, 9=Fine weather,

    // 10=Cloudy weather, 11=Shade, 12=Daylight fluorescent,

    // 13=Day white fluorescent, 14=Cool white fluorescent,

    // 15=White fluorescent, 17=Standard light A, 18=Standard light B, 19=Standard light C,

    // 20=D55, 21=D65, 22=D75, 23=D50, 24=ISO studio tungsten, 255=other light source,
Attribute(FileLongName, 'Flash') as Flash,     Attribute(FileLongName, 'FocalLength') as
FocalLength,     Attribute(FileLongName, 'SubjectArea') as SubjectArea,     Attribute
(FileLongName, 'MakerNote') as MakerNote,     Attribute(FileLongName, 'UserComment') as
UserComment,     Attribute(FileLongName, 'SubSecTime') as SubSecTime,
```

```
    Attribute(FileLongName, 'SubsecTimeOriginal') as SubsecTimeOriginal,    Attribute
(FileLongName, 'SubsecTimeDigitized') as SubsecTimeDigitized,    Attribute(FileLongName,
'XPTitle') as XPTitle,    Attribute(FileLongName, 'XPComment') as XPComment,

    Attribute(FileLongName, 'XPAuthor') as XPAuthor,    Attribute(FileLongName,
'XPKeywords') as XPKeywords,    Attribute(FileLongName, 'XPSubject') as XPSubject,
Attribute(FileLongName, 'FlashpixVersion') as FlashpixVersion,    Attribute(FileLongName,
'ColorSpace') as ColorSpace, //  examples: 1=sRGB, 65535=Uncalibrated,    Attribute
(FileLongName, 'PixelXDimension') as PixelXDimension,    Attribute(FileLongName,
'PixelYDimension') as PixelYDimension,    Attribute(FileLongName, 'RelatedSoundFile') as
RelatedSoundFile,

    Attribute(FileLongName, 'FocalPlaneXResolution') as FocalPlaneXResolution,    Attribute
(FileLongName, 'FocalPlaneYResolution') as FocalPlaneYResolution,    Attribute(FileLongName,
'FocalPlaneResolutionUnit') as FocalPlaneResolutionUnit,

   //  examples: 1=None, 2=Inch, 3=Centimeter,    Attribute(FileLongName, 'ExposureIndex')
as ExposureIndex,    Attribute(FileLongName, 'SensingMethod') as SensingMethod,

   //  examples: 1=Not defined, 2=One-chip color area sensor, 3=Two-chip color area sensor,

   // 4=Three-chip color area sensor, 5=Color sequential area sensor,

   // 7=Trilinear sensor, 8=Color sequential linear sensor,    Attribute(FileLongName,
'FileSource') as FileSource,

   //  examples: 0=Other, 1=Scanner of transparent type,

   // 2=Scanner of reflex type, 3=Digital still camera,    Attribute(FileLongName,
'SceneType') as SceneType,

   //  examples: 1=A directly photographed image,    Attribute(FileLongName, 'CFAPattern')
as CFAPattern,    Attribute(FileLongName, 'CustomRendered') as CustomRendered,

   //  examples: 0=Normal process, 1=Custom process,    Attribute(FileLongName,
'ExposureMode') as ExposureMode,

   //  examples: 0=Auto exposure, 1=Manual exposure, 2=Auto bracket,    Attribute
(FileLongName, 'WhiteBalance') as WhiteBalance,

   //  examples: 0=Auto white balance, 1=Manual white balance,    Attribute(FileLongName,
'DigitalZoomRatio') as DigitalZoomRatio,    Attribute(FileLongName, 'FocalLengthIn35mmFilm')
as FocalLengthIn35mmFilm,    Attribute(FileLongName, 'SceneCaptureType') as SceneCaptureType,

   //  examples: 0=Standard, 1=Landscape, 2=Portrait, 3=Night scene,    Attribute
(FileLongName, 'GainControl') as GainControl,

   //  examples: 0=None, 1=Low gain up, 2=High gain up, 3=Low gain down, 4=High gain down,
  Attribute(FileLongName, 'Contrast') as Contrast,

   //  examples: 0=Normal, 1=Soft, 2=Hard,    Attribute(FileLongName, 'Saturation') as
Saturation,

   //  examples: 0=Normal, 1=Low saturation, 2=High saturation,    Attribute(FileLongName,
'Sharpness') as Sharpness,
```

```
    //  examples: 0=Normal, 1=Soft, 2=Hard,      Attribute(FileLongName,
'SubjectDistanceRange') as SubjectDistanceRange,

    //  examples: 0=Unknown, 1=Macro, 2=Close view, 3=Distant view,      Attribute
(FileLongName, 'ImageUniqueID') as ImageUniqueID,     Attribute(FileLongName,
'BodySerialNumber') as BodySerialNumber,     Attribute(FileLongName, 'CMNT_GAMMA') as CMNT_
GAMMA,     Attribute(FileLongName, 'PrintImageMatching') as PrintImageMatching,     Attribute
(FileLongName, 'OffsetSchema') as OffsetSchema,

     // ************   Interoperability Attributes   ************     Attribute(FileLongName,
'InteroperabilityIndex') as InteroperabilityIndex,     Attribute(FileLongName,
'InteroperabilityVersion') as InteroperabilityVersion,     Attribute(FileLongName,
'InteroperabilityRelatedImageFileFormat') as InteroperabilityRelatedImageFileFormat,
Attribute(FileLongName, 'InteroperabilityRelatedImageWidth') as
InteroperabilityRelatedImageWidth,     Attribute(FileLongName,
'InteroperabilityRelatedImageLength') as InteroperabilityRelatedImageLength,     Attribute
(FileLongName, 'InteroperabilityColorSpace') as InteroperabilityColorSpace,

    //  examples: 1=sRGB, 65535=Uncalibrated,      Attribute(FileLongName,
'InteroperabilityPrintImageMatching') as InteroperabilityPrintImageMatching,     //
************   GPS Attributes   ************     Attribute(FileLongName, 'GPSVersionID') as
GPSVersionID,     Attribute(FileLongName, 'GPSLatitudeRef') as GPSLatitudeRef,     Attribute
(FileLongName, 'GPSLatitude') as GPSLatitude,     Attribute(FileLongName, 'GPSLongitudeRef')
as GPSLongitudeRef,     Attribute(FileLongName, 'GPSLongitude') as GPSLongitude,     Attribute
(FileLongName, 'GPSAltitudeRef') as GPSAltitudeRef,

    //  examples: 0=Above sea level, 1=Below sea level,      Attribute(FileLongName,
'GPSAltitude') as GPSAltitude,     Attribute(FileLongName, 'GPSTimeStamp') as GPSTimeStamp,
 Attribute(FileLongName, 'GPSSatellites') as GPSSatellites,     Attribute(FileLongName,
'GPSStatus') as GPSStatus,     Attribute(FileLongName, 'GPSMeasureMode') as GPSMeasureMode,
 Attribute(FileLongName, 'GPSDOP') as GPSDOP,     Attribute(FileLongName, 'GPSSpeedRef') as
GPSSpeedRef,

     Attribute(FileLongName, 'GPSSpeed') as GPSSpeed,     Attribute(FileLongName,
'GPSTrackRef') as GPSTrackRef,     Attribute(FileLongName, 'GPSTrack') as GPSTrack,
Attribute(FileLongName, 'GPSImgDirectionRef') as GPSImgDirectionRef,     Attribute
(FileLongName, 'GPSImgDirection') as GPSImgDirection,     Attribute(FileLongName,
'GPSMapDatum') as GPSMapDatum,     Attribute(FileLongName, 'GPSDestLatitudeRef') as
GPSDestLatitudeRef,

     Attribute(FileLongName, 'GPSDestLatitude') as GPSDestLatitude,     Attribute
(FileLongName, 'GPSDestLongitudeRef') as GPSDestLongitudeRef,     Attribute(FileLongName,
'GPSDestLongitude') as GPSDestLongitude,     Attribute(FileLongName, 'GPSDestBearingRef') as
GPSDestBearingRef,     Attribute(FileLongName, 'GPSDestBearing') as GPSDestBearing,
Attribute(FileLongName, 'GPSDestDistanceRef') as GPSDestDistanceRef,

     Attribute(FileLongName, 'GPSDestDistance') as GPSDestDistance,     Attribute
(FileLongName, 'GPSProcessingMethod') as GPSProcessingMethod,     Attribute(FileLongName,
'GPSAreaInformation') as GPSAreaInformation,     Attribute(FileLongName, 'GPSDateStamp') as
GPSDateStamp,     Attribute(FileLongName, 'GPSDifferential') as GPSDifferential;

    //  examples: 0=No correction, 1=Differential correction,  LOAD @1:n as FileLongName
Inline "$(vFoundFile)" (fix, no labels); Next vFoundFile Next vExt
```

Example 3: Медиафайлы Windows

Этот скрипт предназначен для чтения всех возможных мета-тегов WMA/WMV ASF в папке *MyMusic*.

```
// Script to read WMA/WMV ASF meta tags for each vExt in 'asf', 'wma', 'wmv' for each
vFoundFile in filelist( GetFolderPath('MyMusic') & '\*.'& vExt )

 FileList: LOAD FileLongName,     subfield(FileLongName,'\',-1) as FileShortName,     num
(FileSize(FileLongName),'# ### ### ###',',',' ') as FileSize,     FileTime(FileLongName) as
FileTime,      Attribute(FileLongName, 'Title') as Title,     Attribute(FileLongName,
'Author') as Author,     Attribute(FileLongName, 'Copyright') as Copyright,     Attribute
(FileLongName, 'Description') as Description,

     Attribute(FileLongName, 'Rating') as Rating,     Attribute(FileLongName, 'PlayDuration')
as PlayDuration,     Attribute(FileLongName, 'MaximumBitrate') as MaximumBitrate,
Attribute(FileLongName, 'WMFSDKVersion') as WMFSDKVersion,     Attribute(FileLongName,
'WMFSDKNeeded') as WMFSDKNeeded,     Attribute(FileLongName, 'IsVBR') as IsVBR,     Attribute
(FileLongName, 'ASFLeakyBucketPairs') as ASFLeakyBucketPairs,

     Attribute(FileLongName, 'PeakValue') as PeakValue,     Attribute(FileLongName,
'AverageLevel') as AverageLevel; LOAD @1:n as FileLongName Inline "$(vFoundFile)" (fix, no
labels); Next vFoundFile Next vExt
```

Example 4: PNG

Этот скрипт предназначен для чтения всех возможных мета-тегов PNG в папке *MyPictures*.

```
// Script to read PNG meta tags for each vExt in 'png' for each vFoundFile in filelist(
GetFolderPath('MyPictures') & '\*.'& vExt )

 FileList: LOAD FileLongName,     subfield(FileLongName,'\',-1) as FileShortName,     num
(FileSize(FileLongName),'# ### ### ###',',',' ') as FileSize,     FileTime(FileLongName) as
FileTime,      Attribute(FileLongName, 'Comment') as Comment,

     Attribute(FileLongName, 'Creation Time') as Creation_Time,     Attribute(FileLongName,
'Source') as Source,     Attribute(FileLongName, 'Title') as Title,     Attribute
(FileLongName, 'Software') as Software,     Attribute(FileLongName, 'Author') as Author,
Attribute(FileLongName, 'Description') as Description,

     Attribute(FileLongName, 'Copyright') as Copyright; LOAD @1:n as FileLongName Inline
"$(vFoundFile)" (fix, no labels); Next vFoundFile Next vExt
```

ConnectString

Функция **ConnectString()** возвращает имя активного подключения к данным для подключений ODBC или OLE DB. Функция возвращает пустую строку, если не выполнен оператор **connect**, или после оператора **disconnect**.

Синтаксис:

```
ConnectString()
```

Примеры и результаты:

Примеры написания скриптов

Пример	Результат

`LIB CONNECT TO 'Tutorial ODBC';`
`ConnectString:`
`Load ConnectString() as ConnectString AutoGenerate 1;`
	Возвращает «Tutorial ODBC» в поле ConnectString
	
	В этом примере предполагается, что у вас есть доступное подключение к данным под названием Tutorial ODBC.

FileBaseName

Функция **FileBaseName** возвращает строку с именем табличного файла, читаемого в текущий момент, без пути или расширения.

Синтаксис:

```
FileBaseName()
```

Примеры и результаты:

Примеры написания скриптов

Пример	Результат
`LOAD *, filebasename( ) as X from C:\UserFiles\abc.txt`	Возвращает 'abc' в поле X в каждой прочитанной записи.

FileDir

Функция **FileDir** возвращает строку, содержащую путь к каталогу табличного файла, читаемого в текущий момент.

Синтаксис:

```
FileDir()
```

Эта функция поддерживает только подключения к данным из папки в стандартном режиме.

Примеры и результаты:

Примеры написания скриптов

Пример	Результат

`Load *, filedir( ) as X from`
 	
`C:\UserFiles\abc.txt`
	Возвращает 'C:\UserFiles' в поле X в каждой прочитанной записи.

FileExtension

Функция **FileExtension** возвращает строку, содержащую расширение табличного файла, читаемого в текущий момент.

Синтаксис:

```
FileExtension()
```

Примеры и результаты:

Примеры написания скриптов

Пример	Результат
`LOAD *, FileExtension( ) as X from C:\UserFiles\abc.txt`	Возвращает 'txt' в поле X в каждой прочитанной записи.

FileName

Функция **FileName** возвращает строку с именем табличного файла, читаемого в текущий момент, без пути, но с расширением.

Синтаксис:

```
FileName()
```

Примеры и результаты:

Примеры написания скриптов

Пример	Результат

`LOAD *, FileName( ) as X from`
 	
`C:\UserFiles\abc.txt`
	Будет возвращено значение 'abc.txt' в поле X в каждой прочитанной записи.

FilePath

Функция **FilePath** возвращает строку, содержащую полный путь табличного файла, читаемого в текущий момент.

Синтаксис:

```
FilePath()
```

Эта функция поддерживает только подключения к данным из папки в стандартном режиме.

Примеры и результаты:

Примеры написания скриптов

Пример	Результат

`Load *, FilePath( ) as X from`
 	
`C:\UserFiles\abc.txt`
	Будет возвращено значение 'C:\UserFiles\abc.txt' в поле X в каждой прочитанной записи.

FileSize

Функция **FileSize** возвращает целое, содержащее размер в байтах файла filename, или, если не указан файл filename, табличного файла, читаемого в текущий момент.

Синтаксис:

FileSize([filename]**)**

Аргументы:

Аргументы

Аргумент	Описание
filename	Имя файла, включающее путь при необходимости, в виде папки или подключения к данным веб-файла. Если имя файла не будет указано, будет использоваться табличный файл, считываемый в данный момент. **Пример:** *'lib://Table Files/'* В прежней версии режима написания скриптов следующие форматы пути тоже поддерживаются: • абсолютный **Пример:** *c:\data* • относительно рабочего каталога приложения Qlik Sense. **Пример:** *data* • URL-адрес (HTTP или FTP), указывающий на местоположение в Интернете или интрасети. **Пример:** *http://www.qlik.com*

Примеры и результаты:

Примеры написания скриптов

Пример	Результат
`LOAD *, FileSize( ) as X from abc.txt;`	Возвращает размер указанного файла (abc.txt) в виде целого числа в поле X в каждой прочитанной записи.
`FileSize( 'lib://DataFiles/xyz.xls' )`	Возвращает размер файла xyz.xls.

FileTime

Функция **FileTime** возвращает метку времени для даты и времени последнего исправления файла filename. Если не указан файл в поле filename, функция ссылается на табличный файл, читаемый в текущий момент.

Синтаксис:

```
FileTime([ filename ])
```

Аргументы:

Аргументы

Аргумент	Описание
filename	Имя файла, включающее путь при необходимости, в виде папки или подключения к данным веб-файла. **Пример:** *'lib://Table Files/'* В прежней версии режима написания скриптов следующие форматы пути тоже поддерживаются: • абсолютный **Пример:** *c:\data* • относительно рабочего каталога приложения Qlik Sense. **Пример:** *data* • URL-адрес (HTTP или FTP), указывающий на местоположение в Интернете или интрасети. **Пример:** *http://www.qlik.com*

Примеры и результаты:

Примеры скриптов

Пример	Результат
`LOAD *, FileTime( ) as X from abc.txt;`	Возвращает дату и время последнего исправления файла (abc.txt) в виде метки времени в поле X в каждой прочитанной записи.
FileTime('xyz.xls')	Возвращает метку времени последнего исправления файла xyz.xls.

GetFolderPath

Функция **GetFolderPath** возвращает значение функции Microsoft Windows *SHGetFolderPath*. Данная функция берет в качестве значения ввода имя папки Microsoft Windows и возвращает полный путь папки.

Эта функция не поддерживается в стандартном режиме.

Синтаксис:

```
GetFolderPath(foldername)
```

Аргументы:

Аргументы

Аргумент	Описание
foldername	Имя папки Microsoft Windows. Имя папки не должно содержать пробелов. Из имени папки, отображающегося в Windows Explorer, следует удалить все пробелы. Примеры: *MyMusic* *MyDocuments*

Примеры и результаты:

Назначение этого примера – получить пути следующих папок Microsoft Windows: *MyMusic*, *MyPictures* and *Windows*. Добавьте образец скрипта в свое приложение и перезагрузите.

```
LOAD   GetFolderPath('MyMusic') as MyMusic,  GetFolderPath('MyPictures') as MyPictures,
GetFolderPath('Windows') as Windows AutoGenerate 1;
```

После перезагрузки приложения в модель данных будут добавлены поля *MyMusic*, *MyPictures* и *Windows* . Каждое поле содержит путь к папке, определенной во время ввода. Пример.

- *C:\Users\smu\Music* for the folder *MyMusic*
- *C:\Users\smu\Pictures* for the folder *MyPictures*
- *C:\Windows* for the folder *Windows*

QvdCreateTime

Эта функция скрипта возвращает метку времени верхнего колонтитула XML из файла QVD при его наличии, в противном случае она возвращает значение NULL.

Синтаксис:

```
QvdCreateTime(filename)
```

Аргументы:

Аргументы

Аргумент	Описание
filename	Имя файла QVD, включающее путь при необходимости, в виде папки или подключения к данным веб-файла. **Пример:** *'lib://Table Files/'* В прежней версии режима написания скриптов следующие форматы пути тоже поддерживаются: • абсолютный **Пример:** *c:\data* • относительно рабочего каталога приложения Qlik Sense. **Пример:** *data* • URL-адрес (HTTP или FTP), указывающий на местоположение в Интернете или интрасети. **Пример:** *http://www.qlik.com*

Пример:

```
QvdCreateTime('MyFile.qvd')

QvdCreateTime('C:\MyDir\MyFile.qvd')

QvdCreateTime('lib://DataFiles/MyFile.qvd')
```

QvdFieldName

Эта функция скрипта возвращает имя числа поля **fieldno** в файле QVD. Если поле не существует, возвращается значение NULL.

Синтаксис:

```
QvdFieldName(filename , fieldno)
```

Аргументы:

Аргументы

Аргумент	Описание
filename	Имя файла QVD, включающее путь при необходимости, в виде папки или подключения к данным веб-файла. **Пример:** *'lib://Table Files/'* В прежней версии режима написания скриптов следующие форматы пути тоже поддерживаются: • абсолютный **Пример:** *c:\data* • относительно рабочего каталога приложения Qlik Sense. **Пример:** *data* • URL-адрес (HTTP или FTP), указывающий на местоположение в Интернете или интрасети. **Пример:** *http://www.qlik.com*
fieldno	Номер поля внутри таблицы, находящейся в файле QVD.

Примеры:

```
QvdFieldName ('MyFile.qvd', 5)

QvdFieldName ('C:\MyDir\MyFile.qvd', 5)

QvdFieldName ('lib://DataFiles/MyFile.qvd', 5)
```

Все три примера возвращают имя пятого поля таблицы, находящейся в файле QVD.

QvdNoOfFields

Эта функция скрипта возвращает число полей в файле QVD.

Синтаксис:

```
QvdNoOfFields(filename)
```

Аргументы:

Аргументы

Аргумент	Описание
filename	Имя файла QVD, включающее путь при необходимости, в виде папки или подключения к данным веб-файла. **Пример:** *'lib://Table Files/'* В прежней версии режима написания скриптов следующие форматы пути тоже поддерживаются: • абсолютный **Пример:** *c:\data* • относительно рабочего каталога приложения Qlik Sense. **Пример:** *data* • URL-адрес (HTTP или FTP), указывающий на местоположение в Интернете или интрасети. **Пример:** *http://www.qlik.com*

Примеры:

```
QvdNoOfFields ('MyFile.qvd')

QvdNoOfFields ('C:\MyDir\MyFile.qvd')

QvdNoOfFields ('lib://DataFiles/MyFile.qvd')
```

QvdNoOfRecords

Пример: Эта функция скрипта возвращает число записей, находящихся в настоящее время в файле QVD.

Синтаксис:

```
QvdNoOfRecords(filename)
```

Аргументы:

Аргументы

Аргумент	Описание
filename	Имя файла QVD, включающее путь при необходимости, в виде папки или подключения к данным веб-файла. **Пример:** *'lib://Table Files/'* В прежней версии режима написания скриптов следующие форматы пути тоже поддерживаются: • абсолютный **Пример:** *c:\data* • относительно рабочего каталога приложения Qlik Sense. **Пример:** *data* • URL-адрес (HTTP или FTP), указывающий на местоположение в Интернете или интрасети. **Пример:** *http://www.qlik.com*

Примеры:

```
QvdNoOfRecords ('MyFile.qvd')

QvdNoOfRecords ('C:\MyDir\MyFile.qvd')

QvdNoOfRecords ('lib://DataFiles/MyFile.qvd')
```

QvdTableName

Эта функция скрипта возвращает имя таблицы, хранящейся в файле QVD.

Синтаксис:

```
QvdTableName(filename)
```

Аргументы:

Аргументы

Аргумент	Описание
filename	Имя файла QVD, включающее путь при необходимости, в виде папки или подключения к данным веб-файла. **Пример:** *'lib://Table Files/'* В прежней версии режима написания скриптов следующие форматы пути тоже поддерживаются: • абсолютный **Пример:** *c:\data* • относительно рабочего каталога приложения Qlik Sense. **Пример:** *data* • URL-адрес (HTTP или FTP), указывающий на местоположение в Интернете или интрасети. **Пример:** *http://www.qlik.com*

Примеры:

```
QvdTableName ('MyFile.qvd')
QvdTableName ('C:\MyDir\MyFile.qvd')
QvdTableName ('lib://data\MyFile.qvd')
```

5.11 Финансовые функции

Финансовые функции можно использовать в скрипте загрузки данных и в выражениях диаграммы для вычисления платежей и процентных ставок.
Во всех аргументах выплачиваемые наличные представлены отрицательными числами.
Полученные наличные представлены положительными числами.
Здесь перечислены те аргументы, которые используются в финансовых функциях (кроме тех, которые начинаются с элемента **range**-):

*Для всех финансовых функций очень важно, чтобы согласованно указывались единицы для **rate** и **nper**. При совершении месячных выплат по пятилетнему кредиту под 6% годовых используйте 0,005 (6%/12) для элемента **rate** и 60 (5*12) для элемента **nper**. Если по тому же кредиту совершаются ежегодные выплаты, используйте 6% для элемента **rate** и 5 для элемента **nper**.*

Обзор финансовых функций

Каждая функция подробно описана после обзора. Также можно щелкнуть имя функции в синтаксисе, чтобы получить немедленный доступ к подробной информации об этой конкретной функции.

FV

Эта функция возвращает будущую стоимость вложения на основе периодических, постоянных платежей и простой годовой процентной ставки.

```
FV (rate, nper, pmt [ ,pv [ , type ] ])
```

nPer

Эта функция возвращает число периодов вложения на основе периодических, постоянных платежей и постоянной процентной ставки.

```
nPer (rate, pmt, pv [ ,fv [ , type ] ])
```

Pmt

Эта функция возвращает платеж по кредиту на основе периодических, постоянных платежей и постоянной процентной ставки. Она не может меняться в течение аннуитета. Платеж указан как отрицательное число, например -20.

```
Pmt (rate, nper, pv [ ,fv [ , type ] ] )
```

PV

Эта функция возвращает текущую стоимость вложения.

```
PV (rate, nper, pmt [ ,fv [ , type ] ])
```

Rate

Эта функция возвращает процентную ставку за период по аннуитету. Результат имеет формат числа по умолчанию **Fix**, два десятичных и %.

```
Rate (nper, pmt , pv [ ,fv [ , type ] ])
```

BlackAndSchole

Модель Black and Scholes – это математическая модель для производных инструментов финансовых рынков. Данная формула используется для расчета теоретической стоимости опциона. В программе Qlik Sense функция **BlackAndSchole** возвращает стоимость по немодифицированной формуле Black and Scholes (на европейские опционы).

```
BlackAndSchole(strike , time_left , underlying_price , vol , risk_free_rate ,
type)
```

Возвращаемые типы данных: числовое значение

Аргументы:

Аргументы

Аргумент	Описание
strike	Будущая цена покупки акции.
time_left	Число периодов до истечения опциона.
underlying_price	Текущая цена акции.
vol	Волатильность (курса акций) выражается в процентах в десятичной форме за период времени.
risk_free_rate	Безрисковая процентная ставка выражается в процентах в десятичной форме за период времени.
call_or_put	Тип опциона: 'c', 'call' или любое ненулевое числовое значение для опционов call 'p', 'put' или 0 для параметров put.

Ограничения:

Значение параметров strike, time_left и underlying_price должно быть >0.

Значение параметров vol и risk_free_rate должно быть <0 или >0.

Примеры и результаты:

Примеры написания скриптов

Пример	Результат
`BlackAndScholе(130, 4, 68.5, 0.4, 0.04, 'call')` Рассчитывается теоретическая цена опциона для покупки акции, стоимость которой в настоящее время составляет 68,5, по цене 130 через 4 года. В этой формуле используется волатильность 0,4 (40%) в год при безрисковой процентной ставке 0,04 (4%).	Возвращает 11,245

FV

Эта функция возвращает будущую стоимость вложения на основе периодических, постоянных платежей и простой годовой процентной ставки.

Синтаксис:

```
FV(rate, nper, pmt [ ,pv [ , type ] ])
```

Возвращаемые типы данных: числовой. Результат имеет числовой денежный формат по умолчанию. .

Аргументы:

Аргументы

Аргумент	Описание
rate	Процентная ставка за период.
nper	Итоговое число сроков оплаты аннуитета.
pmt	Оплата, совершаемая в каждый период. Она не может меняться в течение аннуитета. Платеж указан как отрицательное число, например -20.
pv	Текущая стоимость или единовременно выплачиваемая сумма, которая соответствует ряду будущих выплат в настоящее время. Если **pv** отсутствует, он соответствует 0 (нулю).
type	Должно быть значение 0, если платежи осуществляются в конце периода, и 1, если платежи осуществляются в начале периода. Если **type** отсутствует, он соответствует 0.

Примеры и результаты:

Пример написания скрипта

Пример	Результат
Вы выплачиваете стоимость нового бытового электроприбора путем ежемесячных взносов в размере 20 долл. США в течение 36 месяцев. Процентная ставка составляет 6% годовых. Счет приходит в конце каждого месяца. Каким будет итоговое значение вложенных денег на момент оплаты последнего счета? `FV(0.005,36,-20)`	Возвращает $786.72

nPer

Эта функция возвращает число периодов вложения на основе периодических, постоянных платежей и постоянной процентной ставки.

Синтаксис:

```
nPer(rate, pmt, pv [ ,fv [ , type ] ])
```

Возвращаемые типы данных: числовое значение

Аргументы:

Аргументы

Аргумент	Описание
rate	Процентная ставка за период.
nper	Итоговое число сроков оплаты аннуитета.
pmt	Оплата, совершаемая в каждый период. Она не может меняться в течение аннуитета. Платеж указан как отрицательное число, например -20.
pv	Текущая стоимость или единовременно выплачиваемая сумма, которая соответствует ряду будущих выплат в настоящее время. Если **pv** отсутствует, он соответствует 0 (нулю).
fv	Будущая стоимость, или остаток денежных средств, который вы хотели бы достичь после совершения последнего платежа. Если **fv** отсутствует, он соответствует 0.
type	Должно быть значение 0, если платежи осуществляются в конце периода, и 1, если платежи осуществляются в начале периода. Если **type** отсутствует, он соответствует 0.

Примеры и результаты:

Пример написания скрипта

Пример	Результат
Вы хотите продать бытовой электроприбор путем ежемесячных взносов в размере 20 долл. США. Процентная ставка составляет 6% годовых. Счет приходит в конце каждого месяца. Сколько периодов требуется, если значение полученных денежных средств после оплаты последнего счета должно быть равным 800 долл. США? `nPer(0.005,-20,0,800)`	Возвращает 36,56

Pmt

Эта функция возвращает платеж по кредиту на основе периодических, постоянных платежей и постоянной процентной ставки. Она не может меняться в течение аннуитета. Платеж указан как отрицательное число, например -20.

```
Pmt(rate, nper, pv [ ,fv [ , type ] ] )
```

Возвращаемые типы данных: числовой. Результат имеет числовой денежный формат по умолчанию. .

Чтобы узнать итоговую сумму, уплаченную в течение действия кредита, умножьте возвращенное значение **pmt** на **nper**.

Аргументы:

Аргументы

Аргумент	Описание
rate	Процентная ставка за период.
nper	Итоговое число сроков оплаты аннуитета.
pv	Текущая стоимость или единовременно выплачиваемая сумма, которая соответствует ряду будущих выплат в настоящее время. Если **pv** отсутствует, он соответствует 0 (нулю).
fv	Будущая стоимость, или остаток денежных средств, который вы хотели бы достичь после совершения последнего платежа. Если **fv** отсутствует, он соответствует 0.
type	Должно быть значение 0, если платежи осуществляются в конце периода, и 1, если платежи осуществляются в начале периода. Если **type** отсутствует, он соответствует 0.

Примеры и результаты:

Примеры написания скриптов

Пример	Результат
Следующая формула возвращает ежемесячный платеж по кредиту в 20 тыс. долл. США под 10% годовых, которые необходимо выплатить за 8 месяцев: `Pmt(0.1/12,8,20000)`	Возвращает - $2,594.66
Для того же кредита, если платеж необходимо осуществлять в начале периода, размер платежа составляет: `Pmt(0.1/12,8,20000,0,1)`	Возвращает - $2,573.21

PV

Эта функция возвращает текущую стоимость вложения.

```
PV(rate, nper, pmt [ ,fv [ , type ] ])
```

Возвращаемые типы данных: числовой. Результат имеет числовой денежный формат по умолчанию. .

Текущая стоимость – это итоговая сумма, которая соответствует ряду будущих выплат в настоящее время. Например, если Вы берете деньги взаймы, для кредитора сумма кредита является текущей стоимостью.

Аргументы:

Аргументы

Аргумент	Описание
rate	Процентная ставка за период.
nper	Итоговое число сроков оплаты аннуитета.
pmt	Оплата, совершаемая в каждый период. Она не может меняться в течение аннуитета. Платеж указан как отрицательное число, например -20.
fv	Будущая стоимость, или остаток денежных средств, который вы хотели бы достичь после совершения последнего платежа. Если **fv** отсутствует, он соответствует 0.
type	Должно быть значение 0, если платежи осуществляются в конце периода, и 1, если платежи осуществляются в начале периода. Если **type** отсутствует, он соответствует 0.

Примеры и результаты:

Пример написания скрипта

Пример	Результат
Каково текущее значение долга, если необходимо платить по 100 долл. США в конце каждого месяца в течение 5-летнего периода при процентной ставке 7%? `PV(0.07/12,12*5,-100,0,0)`	Возвращает $5,050.20

Rate

Эта функция возвращает процентную ставку за период по аннуитету. Результат имеет формат числа по умолчанию **Fix**, два десятичных и %.

Синтаксис:

```
Rate(nper, pmt , pv [ ,fv [ , type ] ])
```

Возвращаемые типы данных: числовой.

Элемент **rate** вычисляется циклично и обладает нулевым или несколькими решениями. Если последовательные результаты элемента **rate** не сходятся, возвращается значение NULL.

Аргументы:

Аргументы

Аргумент	Описание
nper	Итоговое число сроков оплаты аннуитета.
pmt	Оплата, совершаемая в каждый период. Она не может меняться в течение аннуитета. Платеж указан как отрицательное число, например -20.
pv	Текущая стоимость или единовременно выплачиваемая сумма, которая соответствует ряду будущих выплат в настоящее время. Если **pv** отсутствует, он соответствует 0 (нулю).
fv	Будущая стоимость, или остаток денежных средств, который вы хотели бы достичь после совершения последнего платежа. Если **fv** отсутствует, он соответствует 0.
type	Должно быть значение 0, если платежи осуществляются в конце периода, и 1, если платежи осуществляются в начале периода. Если **type** отсутствует, он соответствует 0.

Примеры и результаты:

Пример написания скрипта

Пример	Результат
Какова процентная ставка 5-летнего кредита по аннуитету в размере 10 тыс. долл. США с ежемесячными платежами 300 долл. США? `Rate(60,-300,10000)`	Возвращает 2.00%

5.12 Функции форматирования

Функции форматирования применяют формат отображения к числовым полям ввода и выражениям. В зависимости от типа данных можно указать символы для десятичного разделителя, разделителя разрядов и т. д.

Все функции возвращают двойное значение, состоящее из строкового и числового значения, но могут использоваться для преобразования числа в строку. Функция **Dual()** – это особый случай, но другие функции форматирования берут числовое значение входного выражения и создают строку, представляющую собой число.

В отличие от них, функции интерпретации делают все наоборот. Они берут строковые выражения и интерпретируют их в качестве чисел, определяя формат полученного числа.

Функции можно использовать как в скриптах загрузки данных, так и в выражениях диаграмм.

Во всех представлениях чисел в качестве десятичного разделителя используется десятичная точка.

Обзор функций форматирования

Каждая функция подробно описана после обзора. Также можно щелкнуть имя функции в синтаксисе, чтобы получить немедленный доступ к подробной информации об этой конкретной функции.

ApplyCodepage

ApplyCodepage() применяет другой набор символов кодовой страницы к полю или тексту, указанному в выражении. Аргумент **codepage** должен иметь числовой формат.

```
ApplyCodepage (text, codepage)
```

Date

Date() преобразует формат выражения в значение даты, используя формат, указанный в системных переменных в скрипте загрузки данных, в операционной системе или в строке форматирования (если указана).

```
Date (number[, format])
```

Dual

Dual() объединяет число и строку в одной записи таким образом, что число, представляющее строку, можно использовать для сортировки и вычислений, а значение строки может использоваться для отображения.

```
Dual (text, number)
```

Interval

Interval() преобразует формат числа в интервал времени, используя формат системных переменных в скрипте загрузки данных или в операционной системе (либо строку форматирования, если указана).

```
Interval (number[, format])
```

Money

Money() преобразует формат выражения в цифровую форму денежного значения в формат, установленный в системных переменных в скрипте загрузки данных или в операционной системе (если не указана строка форматирования), дополнительно разделяет десятые и сотые доли.

```
Money (number[, format[, dec_sep [, thou_sep]]])
```

Num

Num() форматирует число, то есть преобразует числовое значение ввода в отображаемый текст, используя формат, указанный во втором параметре. Если второй параметр опущен, то используются десятичные и тысячные разделители, установленные в скрипте загрузки данных.

Пользовательские символы разделителей десятичных разрядов и тысяч являются дополнительными параметрами.

```
Num (number[, format[, dec_sep [, thou_sep]]])
```

Time

Time() преобразует формат выражения в значение времени, используя формат системных переменных в скрипте загрузки данных или в операционной системе (если не указана строка форматирования).

```
Time (number[, format])
```

Timestamp

TimeStamp() преобразует формат выражения в значение времени и даты, используя формат системных переменных в скрипте загрузки данных или в операционной системе (если не указана строка форматирования).

```
Timestamp (number[, format])
```

См. также:

Функции интерпретации (page 677)

ApplyCodepage

ApplyCodepage() применяет другой набор символов кодовой страницы к полю или тексту, указанному в выражении. Аргумент **codepage** должен иметь числовой формат.

ApplyCodepage можно использовать в выражениях диаграмм, однако эта функция чаще применяется в качестве функции скрипта в редакторе загрузки данных. Например, при загрузке файлов, которые бесконтрольно могли быть сохранены в разных наборах символов, можно применить кодовую страницу, представляющую необходимый набор символов.

Синтаксис:

```
ApplyCodepage(text, codepage)
```

Возвращаемые типы данных: строка

Аргументы:

Аргументы

Аргумент	Описание
text	Поле или текст, к которым необходимо применить другую кодовую страницу, заданную аргументом **codepage**.
codepage	Число, представляющее кодовую страницу для применения к полю или выражению, заданным аргументом **text**.

Примеры и результаты:

Примеры написания скриптов

Пример	Результат
`LOAD ApplyCodepage (ROWX,1253) as GreekProduct, ApplyCodepage (ROWY, 1255) as HebrewProduct, ApplyCodepage (ROWZ, 65001) as EnglishProduct; SQL SELECT ROWX, ROWY, ROWZ From Products;`	При загрузке из SQL источник может содержать символы из разных наборов: «Кириллица», «Иврит» и т. д. в формате UTF-8. В этой ситуации может потребоваться загрузка по строкам с применением разных кодовых страниц к каждой строке. Значение **codepage** 1253 соответствует набору символов «Windows: греческие», значение 1255 – набору символов «Иврит», значение 65001 – стандартному набору символов латинского алфавита UTF-8.

См. также: *Набор символов (page 115)*

Date

Date() преобразует формат выражения в значение даты, используя формат, указанный в системных переменных в скрипте загрузки данных, в операционной системе или в строке форматирования (если указана).

Синтаксис:

```
Date(number[, format])
```

Возвращаемые типы данных: двойное значение

Аргументы:

Аргументы

Аргумент	Описание
number	Число для изменения формата.
format	Строка, описывающая формат результирующей строки. Если строка формата отсутствует, формат даты устанавливается в системных переменных в скрипте загрузки данных или используются данные операционной системы.

Примеры и результаты:
В указанных ниже примерах используются следующие параметры по умолчанию:

- Параметр даты 1: YY-MM-DD
- Параметр даты 2: M/D/YY

Пример:

```
Date( A )
```
где A=35648

Результирующая таблица

Результаты	Параметр 1	Параметр 2
Строка:	97-08-06	8/6/97
Число:	35648	35648

Пример:

```
Date( A, 'YY.MM.DD' )
```
где A=35648

Результирующая таблица

Результаты	Параметр 1	Параметр 2
Строка:	97.08.06	97.08.06
Число:	35648	35648

Пример:

```
Date( A, 'DD.MM.YYYY' )
```
где A=35648.375

Результирующая таблица

Результаты	Параметр 1	Параметр 2
Строка:	06.08.1997	06.08.1997
Число:	35648.375	35648.375

Пример:

```
Date( A, 'YY.MM.DD' )
```
где A=8/6/97

Результирующая таблица

Результаты	Параметр 1	Параметр 2
Строка:	NULL (ничего)	97.08.06
Число:	NULL	35648

Dual

Dual() объединяет число и строку в одной записи таким образом, что число, представляющее строку, можно использовать для сортировки и вычислений, а значение строки может использоваться для отображения.

Синтаксис:

```
Dual(text, number)
```

Возвращаемые типы данных: двойное значение

Аргументы:

Аргументы

Аргумент	Описание
text	Значение строки, которое будет использоваться вместе с числовым аргументом.
number	Число, которое будет использоваться вместе со строкой в строковом аргументе.

В программе Qlik Sense все значения полей потенциально являются двойными. Это означает, что значения полей могут иметь как числовое, так и текстовое значения. Примером служит дата, которая может иметь числовое значение 40908 и текстовое представление '2011-12-31'.

Если несколько элементов данных, переданных в одно поле, имеют разные строковые представления, но одно действительное числовое представление, то все они будут использовать первое найденное строковое представление.

*Функция **dual**, как правило, используется на ранней стадии выполнения скрипта до передачи других данных в соответствующее поле для создания первого строкового представления, которое будет отображено в фильтре.*

Примеры и результаты:

Примеры написания скриптов

Пример	Описание
Добавьте следующие примеры в скрипт и запустите его. `Load dual ( NameDay,NumDay ) as DayOfWeek inline [ NameDay,NumDay Monday,0 Tuesday,1 Wednesday,2 Thursday,3 Friday,4 Saturday,5 Sunday,6 ];`	Поле DayOfWeek можно использовать в визуализации, например в качестве измерения. В таблице дни недели автоматически сортируются в правильной числовой последовательности, а не по алфавиту.
`Load Dual('Q' & Ceil(Month(Now ())/3), Ceil(Month (Now())/3)) as Quarter AutoGenerate 1;`	В этом примере выполняется определение текущего квартала. Это значение отображается как Q1, если функция **Now()** запускается в первые три месяца года, Q2 – для вторых трех месяцев и так далее. Однако при использовании в сортировке поле Quarter будет вести себя как числовое значение: от 1 до 4.
`Dual('Q' & Ceil (Month(Date)/3), Ceil(Month (Date)/3)) as Quarter`	Как и в предыдущем примере, поле Quarter создается с текстовыми значениями от 'Q1' до 'Q4', и ему назначаются числовые значения от 1 до 4. Для использования в скрипте необходимо загрузить значения для параметра Date.
`Dual(WeekYear(Date) & '-W' & Week (Date), WeekStart (Date)) as YearWeek`	В этом примере создается поле YearWeek с текстовыми значениями вида '2012-W22' и в то же время присваивает числовое значение в соответствии с числом даты первого дня недели, например: 41057. Для использования в скрипте необходимо загрузить значения для параметра Date.

Interval

Interval() преобразует формат числа в интервал времени, используя формат системных переменных в скрипте загрузки данных или в операционной системе (либо строку форматирования, если указана).

Интервалы можно форматировать как время, дни или комбинацию дней, часов, минут, секунд и долей секунд.

Синтаксис:

```
Interval(number[, format])
```

Возвращаемые типы данных: двойное значение

Аргументы:

Аргументы

Аргумент	Описание
number	Число для изменения формата.
format	Строка, описывающая, как будет отформатирована полученная строка интервала. Если пропущено, то используется краткий формат даты, формат времени и десятичный разделитель из операционной системы.

Примеры и результаты:
В указанных ниже примерах используются следующие параметры по умолчанию:

- Параметр формата даты 1: YY-MM-DD
- Параметр формата даты 2: hh:mm:ss
- Десятичный разделитель числа: .

Результирующая таблица

Пример	Строка	Число
`Interval( A )` , где A=0,375	09:00:00	0,375
`Interval( A )` , где A=1,375	33:00:00	1,375
`Interval( A, 'D hh:mm' )` , где A=1,375	1 09:00	1,375
`Interval( A-B, 'D hh:mm' )` , где A=97-08-06 09:00:00 и B=96-08-06 00:00:00	365 09:00	365,375

Money

Money() преобразует формат выражения в цифровую форму денежного значения в формат, установленный в системных переменных в скрипте загрузки данных или в операционной системе (если не указана строка форматирования), дополнительно разделяет десятые и сотые доли.

Синтаксис:

```
Money(number[, format[, dec_sep[, thou_sep]]])
```

Возвращаемые типы данных: двойное значение

Аргументы:

Аргументы

Аргумент	Описание
number	Число для изменения формата.
format	Строка, описывающая, как будет отформатирована полученная строка денежных единиц.
dec_sep	Строка, определяющая десятичный разделитель.
thou_sep	Строка, определяющая разделитель тысяч.

Если аргументы 2-4 не заданы, то используется формат для валюты, установленный в операционной системе.

Примеры и результаты:
В указанных ниже примерах используются следующие параметры по умолчанию:

- Параметр формата денежных единиц 1: kr ##0,00, MoneyThousandSep' '
- Параметр формата денежных единиц 2: $ #,##0.00, MoneyThousandSep','

Пример:

```
Money( A )
```
, где A=35648

Результирующая таблица

Результаты	Параметр 1	Параметр 2
Строка:	kr 35 648,00	$ 35 648,00
Число:	35 648,00	35 648,00

Пример:

```
Money( A, '#,##0 ¥', '.' , ',' )
```
, где A=3564800

Результирующая таблица

Результаты	Параметр 1	Параметр 2
Строка:	3 564 800 ¥	3 564 800 ¥
Число:	3564800	3564800

Num

Num() форматирует число, то есть преобразует числовое значение ввода в отображаемый текст, используя формат, указанный во втором параметре. Если второй параметр опущен, то используются десятичные и тысячные разделители, установленные в скрипте загрузки данных. Пользовательские символы разделителей десятичных разрядов и тысяч являются дополнительными параметрами.

Синтаксис:

```
Num(number[, format[, dec_sep [, thou_sep]]])
```

Возвращаемые типы данных: двойное значение

Функция Num возвращает двойное значение, которое включает строковое и числовое значения. Функция берет числовое значение входного выражения и создает строку, представляющую собой число.

Аргументы:

Аргументы

Аргумент	Описание
number	Число для изменения формата.
format	Строка, указывающая, как будет отформатирована полученная строка. Если не указано, то используются десятичные и тысячные разделители, установленные в скрипте загрузки данных.
dec_sep	Строка, определяющая десятичный разделитель. Если не указано, используется значение переменной DecimalSep, установленное в скрипте загрузки данных.
thou_sep	Строка, определяющая разделитель тысяч. Если не указано, используется значение переменной ThousandSep, установленное в скрипте загрузки данных.

Пример: Выражение диаграммы

Пример:

В следующей таблице показаны результаты, когда поле A равно 35648.312.

Результаты

Символ	Результат
Num(A)	35648.312 (зависит от переменных среды в скрипте)
Num(A, '0.0', '.')	35648.3
Num(A, '0,00', ',')	35648,31

Символ	Результат
Num(A, '#,##0.0', '.', ',')	35,648.3
Num(A, '# ##0', ',', ' ')	35 648

Пример: Скрипт загрузки

Скрипт загрузки

Num может использоваться в скрипте загрузки для форматирования числа даже в том случае, если в скрипте уже указаны десятичный разделитель и разделитель тысяч. Приведенный ниже скрипт загрузки содержит десятичный разделитель и разделитель тысяч, однако *Num* используется в нем для форматирования данных разными способами.

В **Редакторе загрузки данных** создайте новый раздел, добавьте образец скрипта и запустите его. Затем добавьте на лист приложения как минимум поля, указанные в столбце с результатами, чтобы увидеть результаты.

```
SET ThousandSep=','; SET DecimalSep='.'; Transactions: Load *, Num(transaction_amount) as [No formatting], Num(transaction_amount,'0') as [0], Num(transaction_amount,'#,##0') as [#,##0], Num(transaction_amount,'# ###,00') as [# ###,00], Num(transaction_amount,'# ###,00',',',' ') as [# ###,00 , ',' , ' '], Num(transaction_amount,'#,###.00','.',',') as [#,###.00 , '.' , ','], Num(transaction_amount,'$#,###.00') as [$#,###.00], ; Load * Inline [ transaction_id, transaction_date, transaction_amount, transaction_quantity, discount, customer_id, size, color_code 3750, 20180830, 12423.56, 23, 0,2038593, L, Red 3751, 20180907, 5356.31, 6, 0.1, 203521, m, orange 3752, 20180916, 15.75, 1, 0.22, 5646471, S, blue 3753, 20180922, 1251, 7, 0, 3036491, l, Black 3754, 20180922, 21484.21, 1356, 75, 049681, xs, Red 3756, 20180922, -59.18, 2, 0.333333333333333, 2038593, M, Blue 3757, 20180923, 3177.4, 21, .14, 203521, XL, Black ];
```

Таблица Qlik Sense, в которой представлены результаты применения разных способов использования функции *Num* в скрипте загрузки. В четвертом столбце таблицы представлен пример некорректного использования форматирования.

Без форматирования	0	#,##0	# ###,00	# ###,00 , ',' , ''	#,###.00 , '' , ','	$#,###.00
-59.18	-59	-59	-59###,00	-59,18	-59.18	$-59,18
15.75	16	16	16###,00	15,75	15.75	$15,75
1251	1251	1,251	1251###,00	1 251,00	1,251.00	$1,251.00
3177.4	3177	3,177	3177###,00	3 177,40	3,177.40	$3,177.40
5356.31	5356	5,356	5356###,00	5 356,31	5,356.31	$5,356.31
12423.56	12424	12,424	12424###,00	12 423,56	12,423.56	$12,423.56
21484.21	21484	21,484	21484###,00	21 484,21	21,484.21	$21,484.21

Пример: Скрипт загрузки

Скрипт загрузки

Num может использоваться в скрипте загрузки для форматирования числа в качестве процентного значения.

В **Редакторе загрузки данных** создайте новый раздел, добавьте образец скрипта и запустите его. Затем добавьте на лист приложения как минимум поля, указанные в столбце с результатами, чтобы увидеть результаты.

```
SET ThousandSep=','; SET DecimalSep='.'; Transactions: Load *, Num(discount,'#,##0%') as
[Discount #,##0%] ; Load * Inline [ transaction_id, transaction_date, transaction_amount,
transaction_quantity, discount, customer_id, size, color_code 3750, 20180830, 12423.56, 23,
0,2038593, L, Red 3751, 20180907, 5356.31, 6, 0.1, 203521, m, orange 3752, 20180916, 15.75, 1,
0.22, 5646471, S, blue 3753, 20180922, 1251, 7, 0, 3036491, l, Black 3754, 20180922, 21484.21,
1356, 75, 049681, xs, Red 3756, 20180922, -59.18, 2, 0.333333333333333, 2038593, M, Blue 3757,
20180923, 3177.4, 21, .14, 203521, XL, Black ];
```

Таблица Qlik Sense, в которой представлены результаты использования функции *Num* в скрипте загрузки для форматирования процентных значений.

Скидка	Discount #,##0%
0.333333333333333	33%
0.22	22%
0	0%
.14	14%
0.1	10%
0	0%
75	7,500%

Time

Time() преобразует формат выражения в значение времени, используя формат системных переменных в скрипте загрузки данных или в операционной системе (если не указана строка форматирования).

Синтаксис:

```
Time(number[, format])
```

Возвращаемые типы данных: двойное значение

Аргументы:

Аргументы

Аргумент	Описание
number	Число для изменения формата.
format	Строка, описывающая, как будет отформатирована полученная строка времени. Если пропущено, то используется краткий формат даты, формат времени и десятичный разделитель из операционной системы.

Примеры и результаты:
В указанных ниже примерах используются следующие параметры по умолчанию:

- Параметр формата времени 1: hh:mm:ss
- Параметр формата времени 2: hh.mm.ss

Пример:

```
Time( A )
```
, где A=0,375

Результирующая таблица

Результаты	Параметр 1	Параметр 2
Строка:	09:00:00	09.00.00
Число:	0.375	0.375

Пример:

```
Time( A )
```
, где A=35648,375

Результирующая таблица

Результаты	Параметр 1	Параметр 2
Строка:	09:00:00	09.00.00
Число:	35648.375	35648.375

Пример:

```
Time( A, 'hh-mm' )
```
, где A=0,99999

Результирующая таблица

Результаты	Параметр 1	Параметр 2
Строка:	23-59	23-59
Число:	0.99999	0.99999

Timestamp

TimeStamp() преобразует формат выражения в значение времени и даты, используя формат системных переменных в скрипте загрузки данных или в операционной системе (если не указана строка форматирования).

Синтаксис:

```
Timestamp(number[, format])
```

Возвращаемые типы данных: двойное значение

Аргументы:

Аргументы

Аргумент	Описание
number	Число для изменения формата.
format	Строка, описывающая, как будет отформатирована полученная строка метки времени. Если пропущено, то используется краткий формат даты, формат времени и десятичный разделитель из операционной системы.

Примеры и результаты:
В указанных ниже примерах используются следующие параметры по умолчанию:

- Параметр TimeStampFormat 1: YY-MM-DD hh:mm:ss
- Параметр TimeStampFormat 2: M/D/YY hh:mm:ss

Пример:

```
Timestamp( A )
```
, где A=35648,375

Результирующая таблица

Результаты	Параметр 1	Параметр 2
Строка:	97-08-06 09:00:00	8/6/97 09:00:00
Число:	35648.375	35648.375

Пример:

```
Timestamp( A,'YYYY-MM-DD hh.mm')
```
, где A=35648

Результирующая таблица

Результаты	Параметр 1	Параметр 2
Строка:	1997-08-06 00.00	1997-08-06 00.00
Число:	35648	35648

5.13 Общие числовые функции

Аргументы в этих общих числовых функциях – это выражения, в которых переменная **x** должна интерпретироваться как действительное число. Все функции можно использовать как в скриптах загрузки данных, так и в выражениях диаграмм.

Обзор общих числовых функций

Каждая функция подробно описана после обзора. Также можно щелкнуть имя функции в синтаксисе, чтобы получить немедленный доступ к подробной информации об этой конкретной функции.

bitcount

BitCount() возвращает количество битов в двоичном эквиваленте, для которых установлено значение 1. То есть эта функция возвращает число битов набора в **integer_number**, где **integer_number** интерпретируется как 32-разрядное целое со знаком.

```
BitCount(integer_number)
```

div

Div() возвращает целую часть арифметического деления первого аргумента на второй аргумент. Оба параметра интерпретируются как действительные числа, то есть они не обязательно должны быть целыми числами.

```
Div (integer_number1, integer_number2)
```

fabs

Fabs() возвращает абсолютное значение **x**. Результат – положительное число.

```
Fabs (x)
```

fact

Fact() возвращает факториал положительного целого числа **x**.

```
Fact (x)
```

frac

Frac() возвращает дробную часть **x**.

```
Frac (x)
```

sign

Sign() возвращает 1, 0 или -1 в зависимости от того, чем является **x** – положительным, отрицательным числом или 0.

```
Sign (x)
```

Функции сочетаний и перестановок

combin

Combin() возвращает число комбинаций **q** элементов, которые могут быть получены из набора элементов **p**. Как видно из формулы: `Combin(p,q) = p! / q!(p-q)!` порядок выбора элементов не имеет значения.

```
Combin (p, q)
```

permut

Permut() возвращает число перестановок элементов **q**, которые могут быть выбраны из набора элементов **p**. Как видно из формулы: `Permut(p,q) = (p)! / (p - q)!` порядок выбора элементов имеет значение.

```
Permut (p, q)
```

Функции Modulo

fmod

fmod() является обобщенной функцией modulo, которая возвращает оставшуюся часть целочисленного деления первого аргумента (делимого) на второй аргумент (делитель). Результат – действительное число. Оба аргумента интерпретируются как действительные числа, то есть они не обязательно должны быть целыми числами.

```
Fmod (a, b)
```

mod

Mod() является математической функцией modulo, которая возвращает неотрицательный остаток целочисленного деления. Первый аргумент – делимое, второй аргумент – делитель. Оба аргумента должны иметь целые значения.

```
Mod (integer_number1, integer_number2)
```

Функции четности

even

Even() возвращает значение True (-1), если **integer_number** – четное целое число или ноль. Возвращает False (0), если **integer_number** – нечетное целое число, и NULL, если **integer_number** – нецелое число.

```
Even (integer_number)
```

odd

Odd() возвращает значение True (-1), если **integer_number** – нечетное целое число или ноль. Возвращает False (0), если **integer_number** – четное целое число, и NULL, если **integer_number** – нецелое число.

```
Odd (integer_number)
```

Функции округления

ceil

Ceil() используется для округления чисел в большую сторону до ближайших нескольких чисел интервала **step**, смещенного в соответствии со значением **offset** .

```
Ceil (x[, step[, offset]])
```

floor

Floor() используется для округления чисел в меньшую сторону до ближайших нескольких чисел интервала **step**, смещенного в соответствии со значением **offset** .

```
Floor (x[, step[, offset]])
```

round

Round() возвращает результат округления числа в большую или меньшую сторону до ближайших нескольких чисел интервала **step**, смещенного в соответствии со значением **offset** .

```
Round ( x [ , step [ , offset ]])
```

BitCount

BitCount() возвращает количество битов в двоичном эквиваленте, для которых установлено значение 1. То есть эта функция возвращает число битов набора в **integer_number**, где **integer_number** интерпретируется как 32-разрядное целое со знаком.

Синтаксис:

```
BitCount(integer_number)
```

Возвращаемые типы данных: целое

Примеры и результаты:

Примеры и результаты

Примеры	Результаты
`BitCount ( 3 )`	3 является двоичным числом 11, поэтому возвращается значение 2
`BitCount ( -1 )`	-1 является 64 числами в двоичном числе, поэтому возвращается значение 64

Ceil

Ceil() используется для округления чисел в большую сторону до ближайших нескольких чисел интервала **step**, смещенного в соответствии со значением **offset** .

Сравните с функцией **floor**, которая округляет числа ввода в меньшую сторону.

Синтаксис:

```
Ceil(x[, step[, offset]])
```

Возвращаемые типы данных: числовое значение

Аргументы:

Аргументы

Аргумент	Описание
x	Число ввода.
step	Приращение интервала. Значение по умолчанию – 1.
offset	Определяет базовое значение интервала шага. Значение по умолчанию – 0.

Примеры и результаты:

Примеры и результаты

Примеры	Результаты
`Ceil(2.4 )`	Возвращает 3 В данном примере значение размера шага – 1, базовое значение интервала шага – 0. Интервалы: ...0 < x <=1, 1 < x <= 2, **2< x <=3**, 3< x <=4...
`Ceil(4.2 )`	Возвращает 5
`Ceil(3.88 ,0.1)`	Возвращает 3,9 В данном примере значение размера интервала – 0,1, базовое значение интервала – 0. Интервалы: ... 3.7 < x <= 3.8, **3.8 < x <= 3.9**, 3.9 < x <= 4.0...
`Ceil(3.88 ,5)`	Возвращает 5
`Ceil(1.1 ,1)`	Возвращает 2

Примеры	Результаты
`Ceil(1.1 ,1,0.5)`	Возвращает 1,5 В данном примере значение размера шага – 1, значение смещения – 0,5. Это означает, что базовое значение интервала шага составляет 0,5, а не 0. Интервалы: ...**0.5 < x <=1.5**, 1.5 < x <= 2.5, 2.5< x <=3.5, 3.5< x <=4.5...
`Ceil(1.1 ,1,-0.01)`	Возвращает 1,99 Интервалы: ...-0.01< x <= 0.99, **0.99< x <= 1.99**, 1.99 < x <=2.99...

Combin

Combin() возвращает число комбинаций **q** элементов, которые могут быть получены из набора элементов **p**. Как видно из формулы: `Combin(p,q) = p! / q!(p-q)!` порядок выбора элементов не имеет значения.

Синтаксис:

```
Combin(p, q)
```

Возвращаемые типы данных: целое

Ограничения:

Нецелые элементы будут усечены.

Примеры и результаты:

Примеры и результаты

Примеры	Результаты
Сколько сочетаний 7 чисел может быть получено из 35 чисел лотереи? `Combin( 35,7 )`	Возвращает 6 724 520

Div

Div() возвращает целую часть арифметического деления первого аргумента на второй аргумент. Оба параметра интерпретируются как действительные числа, то есть они не обязательно должны быть целыми числами.

Синтаксис:

```
Div(integer_number1, integer_number2)
```

Возвращаемые типы данных: целое

Примеры и результаты:

Примеры и результаты

Примеры	Результаты
`Div( 7,2 )`	Возвращает 3
`Div( 7.1,2.3 )`	Возвращает 3
`Div( 9,3 )`	Возвращает 3
`Div( -4,3 )`	Возвращает -1
`Div( 4,-3 )`	Возвращает -1
`Div( -4,-3 )`	Возвращает 1

Even

Even() возвращает значение True (-1), если **integer_number** – четное целое число или ноль. Возвращает False (0), если **integer_number** – нечетное целое число, и NULL, если **integer_number** – нецелое число.

Синтаксис:

```
Even(integer_number)
```

Возвращаемые типы данных: Булево значение

Примеры и результаты:

Примеры и результаты

Примеры	Результаты
`Even( 3 )`	Возвращает 0, False
`Even( 2 * 10 )`	Возвращает -1, True
`Even( 3.14 )`	Возвращает NULL

Fabs

Fabs() возвращает абсолютное значение **x**. Результат – положительное число.

Синтаксис:

```
fabs(x)
```

Возвращаемые типы данных: числовое значение

Примеры и результаты:

Примеры и результаты

Примеры	Результаты
`fabs( 2.4 )`	Возвращает 2,4
`fabs( -3.8 )`	Возвращает 3,8

Fact

Fact() возвращает факториал положительного целого числа **x**.

Синтаксис:

```
Fact(x)
```

Возвращаемые типы данных: целое

Ограничения:

Если число **x** не является целым, оно будет обрезано. Неположительные числа возвращают значение NULL.

Примеры и результаты:

Примеры и результаты

Примеры	Результаты
`Fact( 1 )`	Возвращает 1
`Fact( 5 )`	Возвращает 120 (1 * 2 * 3 * 4 * 5 = 120)
`Fact( -5 )`	Возвращает NULL

Floor

Floor() используется для округления чисел в меньшую сторону до ближайших нескольких чисел интервала **step**, смещенного в соответствии со значением **offset** .

Сравните с функцией **ceil**, которая округляет числа ввода в большую сторону.

Синтаксис:

```
Floor(x[, step[, offset]])
```

Возвращаемые типы данных: числовое значение

Аргументы:

Аргументы

Аргумент	Описание
x	Число ввода.
step	Приращение интервала. Значение по умолчанию – 1.
offset	Определяет базовое значение интервала шага. Значение по умолчанию – 0.

Примеры и результаты:

Примеры и результаты

Примеры	Результаты
`Floor(2.4)`	Возвращает 2 In this example, the size of the step is 1 and the base of the step interval is 0. The intervals are ...0 <= x <1, 1 <= x < 2, **2<= x <3**, 3<= x <4....
`Floor(4.2)`	Возвращает 4
`Floor(3.88 ,0.1)`	Возвращает 3,8 В данном примере значение размера интервала – 0,1, базовое значение интервала – 0. Интервалы: ... 3.7 <= x < 3.8, **3.8 <= x < 3.9**, 3.9 <= x < 4.0...
`Floor(3.88 ,5)`	Возвращает 0
`Floor(1.1 ,1)`	Возвращает 1
`Floor(1.1 ,1,0.5)`	Возвращает 0,5 В данном примере значение размера шага – 1, значение смещения – 0,5. Это означает, что базовое значение интервала шага составляет 0,5, а не 0. Интервалы: ...**0.5 <= x <1.5**, 1.5 <= x < 2.5, 2.5<= x <3.5,...

Fmod

fmod() является обобщенной функцией modulo, которая возвращает оставшуюся часть целочисленного деления первого аргумента (делимого) на второй аргумент (делитель). Результат – действительное число. Оба аргумента интерпретируются как действительные числа, то есть они не обязательно должны быть целыми числами.

Синтаксис:

```
fmod(a, b)
```

Возвращаемые типы данных: числовое значение

Аргументы:

Аргументы

Аргумент	Описание
a	Делимое
b	Делитель

Примеры и результаты:

Примеры и результаты

Примеры	Результаты
`fmod( 7,2 )`	Возвращает 1
`fmod( 7.5,2 )`	Возвращает 1,5
`fmod( 9,3 )`	Возвращает 0
`fmod( -4,3 )`	Возвращает -1
`fmod( 4,-3 )`	Возвращает 1
`fmod( -4,-3 )`	Возвращает -1

Frac

Frac() возвращает дробную часть **x**.

Десятичная дробь определяется следующим образом: `Frac(x ) + Floor(x ) = x`. Говоря простым языком, это значит, что дробная часть положительного числа является разницей между числом (x) и целым числом, предшествующим ему.

Пример. Дробная часть 11,43 = 11,43 - 11 = 0,43

Для отрицательного числа допустим, что -1,4, Floor(-1.4) = -2, что приведет к следующему результату.

Дробная часть -1,4 = -1,4 - (-2) = -1,4 + 2 = 0,6

Синтаксис:

```
Frac(x)
```

Возвращаемые типы данных: числовое значение

Аргументы:

Аргументы

Аргумент	Описание
x	Число, для которого возвращается дробная часть.

Примеры и результаты:

Примеры и результаты

Примеры	Результаты
`Frac( 11.43 )`	Возвращает 0,43
`Frac( -1.4 )`	Возвращает 0,6
Извлеките компонент времени из числового представления метки времени, таким образом, пропуская дату. `Time(Frac(44518.663888889))`	Возвращает 3:56:00 PM

Mod

Mod() является математической функцией modulo, которая возвращает неотрицательный остаток целочисленного деления. Первый аргумент – делимое, второй аргумент – делитель. Оба аргумента должны иметь целые значения.

Синтаксис:

```
Mod(integer_number1, integer_number2)
```

Возвращаемые типы данных: целое

Ограничения:

Значение **integer_number2** должно быть больше 0.

Примеры и результаты:

Примеры и результаты

Примеры	Результаты
`Mod( 7,2 )`	Возвращает 1
`Mod( 7.5,2 )`	Возвращает NULL

Примеры	Результаты
`Mod( 9,3 )`	Возвращает 0
`Mod( -4,3 )`	Возвращает 2
`Mod( 4,-3 )`	Возвращает NULL
`Mod( -4,-3 )`	Возвращает NULL

Odd

Odd() возвращает значение True (-1), если **integer_number** – нечетное целое число или ноль. Возвращает False (0), если **integer_number** – четное целое число, и NULL, если **integer_number** – нецелое число.

Синтаксис:

```
Odd(integer_number)
```

Возвращаемые типы данных: Булево значение

Примеры и результаты:

Примеры и результаты

Примеры	Результаты
`Odd( 3 )`	Возвращает -1, True
`Odd( 2 * 10 )`	Возвращает 0, False
`Odd( 3.14 )`	Возвращает NULL

Permut

Permut() возвращает число перестановок элементов **q**, которые могут быть выбраны из набора элементов **p**. Как видно из формулы: `Permut(p,q) = (p)! / (p - q)!` порядок выбора элементов имеет значение.

Синтаксис:

```
Permut(p, q)
```

Возвращаемые типы данных: целое

Ограничения:

Нецелые аргументы будут усечены.

Примеры и результаты:

Примеры и результаты

Примеры	Результаты
Сколько существует вариантов распределения золотой, серебряной и бронзовой медалей после финального забега на 100 м среди 8 участников? `Permut( 8,3 )`	Возвращает 336

Round

Round() возвращает результат округления числа в большую или меньшую сторону до ближайших нескольких чисел интервала **step**, смещенного в соответствии со значением **offset** .

Если число, подлежащее округлению, находится точно посередине интервала, выполняется округление в большую сторону.

Синтаксис:

```
Round(x[, step[, offset]])
```

Возвращаемые типы данных: числовое значение

> *При округлении числа с плавающей запятой результаты могут быть неверными. Обычно такие ошибки округления возникают потому, что числа с плавающей запятой отображаются ограниченным числом двоичных значений. Следовательно, вычисление результатов осуществляется с использованием уже округленного числа. Если ошибки округления могут повлиять на результаты вашей работы, перед округлением выполните умножение чисел для преобразования их в целые числа.*

Аргументы:

Аргументы

Аргумент	Описание
x	Число ввода.
step	Приращение интервала. Значение по умолчанию – 1.
offset	Определяет базовое значение интервала шага. Значение по умолчанию – 0.

Примеры и результаты:

Примеры и результаты

Примеры	Результаты
`Round(3.8 )`	Возвращает 4 В данном примере значение размера шага – 1, базовое значение интервала шага – 0. Интервалы: ...0 <= x <1, 1 <= x < 2, 2<= x <3, **3<= x <4**...
`Round(3.8,4 )`	Возвращает 4
`Round(2.5 )`	Возвращает 3. В данном примере значение размера шага – 1, базовое значение интервала шага – 0. Интервалы: ...0 <= x <1, 1 <= x <2, **2<= x <3**
`Round(2,4 )`	Возвращает 4. Округляется в большую сторону, поскольку значение 2 находится ровно посередине интервала шага, равного 4. В данном примере значение размера шага – 4, базовое значение интервала шага – 0. Интервалы: ...**0 <= x <4**, 4 <= x <8, 8<= x <12
`Round(2,6 )`	Возвращает 0. Округляется в меньшую сторону, поскольку значение 2 меньше половины интервала шага, равного 6. В данном примере значение размера шага – 6, базовое значение интервала шага – 0. Интервалы: ...**0 <= x <6**, 6 <= x <12, 12<= x <18
`Round(3.88 ,0.1)`	Возвращает 3,9 В данном примере значение размера шага – 0,1, базовое значение интервала шага – 0. Интервалы: ... 3.7 <= x <3.8, **3.8 <= x <3.9**, 3.9 <= x < 4.0...
`Round (3.88875,1/1000)`	Возвращает 3,889 В данном примере величина шага составляет 0,001, вследствие чего число округляется, а число знаков после десятичной запятой ограничивается до трех.
`Round(3.88 ,5)`	Возвращает 5

Примеры	Результаты
`Round(1.1 ,1,0.5)`	Возвращает 1,5 В данном примере значение размера шага – 1, базовое значение интервала шага – 0,5. Интервалы: ...**0.5 <= x <1.5**, 1.5 <= x <2.5, 2.5<= x <3.5...

Sign

Sign() возвращает 1, 0 или -1 в зависимости от того, чем является **x** – положительным, отрицательным числом или 0.

Синтаксис:

```
Sign(x)
```

Возвращаемые типы данных: числовое значение

Ограничения:

Если числовые значения не найдены, возвращается значение NULL.

Примеры и результаты:

Примеры и результаты

Примеры	Результаты
`Sign( 66 )`	Возвращает 1
`Sign( 0 )`	Возвращает 0
`Sign( - 234 )`	Возвращает -1

5.14 Геопространственные функции

Эти функции используются при работе с геопространственными данными в визуализациях карт. Qlik Sense соответствует спецификациям геопространственных данных GeoJSON и поддерживает следующие виды представления данных:

- Point
- Linestring
- Polygon
- Multipolygon

Для получения дополнительных сведений о спецификациях GeoJSON см.:
GeoJSON.org

Обзор геопространственных функций

Каждая функция подробно описана после обзора. Также можно щелкнуть имя функции в синтаксисе, чтобы получить немедленный доступ к подробной информации об этой конкретной функции.

Существует две категории геопространственных функций: агрегирование и неагрегирование.

Функции агрегирования получают набор геометрии (точки или области) в качестве входных данных и возвращают единую геометрию. Например, можно объединить несколько областей и изобразить на карте одну границу для агрегирования.

Функция неагрегирования получает одну геометрию и возвращает одну геометрию. Например, если при использовании функции GeoGetPolygonCenter() в качестве входных данных задана геометрия границы одной области, будет возвращена точка геометрии (долгота и широта) для центра этой области.

Ниже приведены функции агрегирования.

GeoAggrGeometry

GeoAggrGeometry() используется для агрегирования нескольких областей в одну большую область, например, агрегирование нескольких подрегионов в один регион.

```
GeoAggrGeometry (field_name)
```

GeoBoundingBox

GeoBoundingBox() используется для агрегирования геометрии в область и вычисления наименьшего ограничивающего прямоугольника, содержащего все координаты.

```
GeoBoundingBox (field_name)
```

GeoCountVertex

GeoCountVertex() используется для вычисления количества вершин у многоугольной геометрии.

```
GeoCountVertex(field_name)
```

GeoInvProjectGeometry

GeoInvProjectGeometry() используется для агрегирования геометрии в область и применения обратного значения проекции.

```
GeoInvProjectGeometry(type, field_name)
```

GeoProjectGeometry

GeoProjectGeometry() используется для агрегирования геометрии в область и применения проекции.

```
GeoProjectGeometry(type, field_name)
```

GeoReduceGeometry

GeoReduceGeometry() используется для сокращения количества вершин геометрии и агрегирования нескольких областей в одну область с отображением границ отдельных областей.

```
GeoReduceGeometry (geometry)
```

Ниже приведены функции неагрегирования.

GeoGetBoundingBox

GeoGetBoundingBox() используется в скриптах и выражениях диаграмм для вычисления наименьшего геопространственного ограничивающего прямоугольника, содержащего все координаты геометрии.

```
GeoGetBoundingBox (geometry)
```

GeoGetPolygonCenter

GeoGetPolygonCenter() используется в скриптах и выражениях диаграмм для вычисления и возврата центральной точки геометрии.

```
GeoGetPolygonCenter (geometry)
```

GeoMakePoint

GeoMakePoint() используется в скриптах и выражениях диаграмм для создания и обозначения широты и долготы точки.

```
GeoMakePoint (lat_field_name, lon_field_name)
```

GeoProject

GeoProject() используется в скриптах и выражениях диаграмм для применения проекции к геометрии.

```
GeoProject (type, field_name)
```

GeoAggrGeometry

GeoAggrGeometry() используется для агрегирования нескольких областей в одну большую область, например, агрегирование нескольких подрегионов в один регион.

Синтаксис:

```
GeoAggrGeometry(field_name)
```

Возвращаемые типы данных: строка

Аргументы:

Аргументы

Аргумент	Описание
field_name	Поле или выражение, относящееся к полю, в котором содержится геометрия для представления. Может быть представлена в виде точки (или набора точек), задающей долготу и широту, или области.

Обычно **GeoAggrGeometry()** используется для объединения данных геопространственных границ. Например, у вас могут быть области с почтовыми кодами для окраины города и доходы от продаж для каждой области. Если территория менеджера по продажам включает в себя несколько областей с почтовыми кодами, может понадобиться представить общий объем продаж для территории ведения продаж, а не для отдельных областей, и отобразить эти результаты на карте с заливкой цветом.

GeoAggrGeometry() может вычислить агрегирование отдельных геометрий окраины и создать геометрию объединенной территории в модели данных. В случае последующей настройки границ территории продаж после перезагрузки данных на карте отобразятся новые объединенные границы и доходы.

Так как функция **GeoAggrGeometry()** является функцией агрегирования, для ее использования в скрипте требуется оператор **LOAD** с предложением **Group by**.

*Линии границы на карте, созданной с помощью **GeoAggrGeometry()**, представляют собой объединенные области. Чтобы отобразить линии отдельной границы предварительно агрегированных областей, используйте **GeoReduceGeometry()**.*

Примеры:
В данном примере показан порядок загрузки файла KML с данными области и таблицы с агрегированными данными области.

```
[MapSource]: LOAD [world.Name], [world.Point], [world.Area] FROM [lib://Downloads/world.kml]
(kml, Table is [World.shp/Features]); Map: LOAD world.Name, GeoAggrGeometry(world.Area) as
[AggrArea] resident MapSource Group By world.Name;

 Drop Table MapSource;
```

GeoBoundingBox

GeoBoundingBox() используется для агрегирования геометрии в область и вычисления наименьшего ограничивающего прямоугольника, содержащего все координаты.

Элемент GeoBoundingBox представлен в виде списка из четырех значений: левого, правого, верхнего и нижнего.

Синтаксис:

```
GeoBoundingBox(field_name)
```

Возвращаемые типы данных: строка

Аргументы:

Аргументы

Аргумент	Описание
field_name	Поле или выражение, относящееся к полю, в котором содержится геометрия для представления. Может быть представлена в виде точки (или набора точек), задающей долготу и широту, или области.

GeoBoundingBox() агрегирует набор геометрий и возвращает четыре координаты для наименьшего прямоугольника, в котором содержатся все координаты агрегированной геометрии.

Для визуализации результата на карте переместите результирующую строку с четырьмя координатами в полигональный формат, нанесите на перемещенное поле метку геополигонного формата и перетащите данное поле в объект карты. После этого прямоугольные поля отобразятся в визуализации карты.

GeoCountVertex

GeoCountVertex() используется для вычисления количества вершин у многоугольной геометрии.

Синтаксис:

```
GeoCountVertex(field_name)
```

Возвращаемые типы данных: целое

Аргументы:

Аргументы

Аргумент	Описание
field_name	Поле или выражение, относящееся к полю, в котором содержится геометрия для представления. Может быть представлена в виде точки (или набора точек), задающей долготу и широту, или области.

GeoGetBoundingBox

GeoGetBoundingBox() используется в скриптах и выражениях диаграмм для вычисления наименьшего геопространственного ограничивающего прямоугольника, содержащего все координаты геометрии.

Геопространственный ограничивающий прямоугольник, созданный с помощью функции GeoBoundingBox(), представлен в виде списка из четырех значений: слева, справа, сверху, снизу.

Синтаксис:

```
GeoGetBoundingBox(field_name)
```

Возвращаемые типы данных: строка

Аргументы:

Аргументы

Аргумент	Описание
field_name	Поле или выражение, относящееся к полю, в котором содержится геометрия для представления. Может быть представлена в виде точки (или набора точек), задающей долготу и широту, или области.

*Не используйте предложение **Group by** в редакторе загрузки данных с этой и другими неагрегирующими геопространственными функциями, так как это приведет к возникновению ошибки загрузки.*

GeoGetPolygonCenter

GeoGetPolygonCenter() используется в скриптах и выражениях диаграмм для вычисления и возврата центральной точки геометрии.

В некоторых случаях требуется наносить на карту точку вместо цветной заливки. Если существующие геопространственные данные доступны только в виде геометрии области (например, границы), используйте **GeoGetPolygonCenter()** для извлечения пары, состоящей из долготы и широты, для центра области.

Синтаксис:

```
GeoGetPolygonCenter(field_name)
```

Возвращаемые типы данных: строка

Аргументы:

Аргументы

Аргумент	Описание
field_name	Поле или выражение, относящееся к полю, в котором содержится геометрия для представления. Может быть представлена в виде точки (или набора точек), задающей долготу и широту, или области.

*Не используйте предложение **Group by** в редакторе загрузки данных с этой и другими неагрегирующими геопространственными функциями, так как это приведет к возникновению ошибки загрузки.*

GeoInvProjectGeometry

GeoInvProjectGeometry() используется для агрегирования геометрии в область и применения обратного значения проекции.

Синтаксис:

```
GeoInvProjectGeometry(type, field_name)
```

Возвращаемые типы данных: строка

Аргументы:

Аргументы

Аргумент	Описание
type	Тип проекции, используемый для преобразования геометрии карты. Может присутствовать одно из двух значений: «единица» (по умолчанию), которое создает проекцию 1:1, или «проекция Меркатора», которое использует стандартную проекцию Меркатора.
field_name	Поле или выражение, относящееся к полю, в котором содержится геометрия для представления. Может быть представлена в виде точки (или набора точек), задающей долготу и широту, или области.

Пример:

Пример написания скрипта

Пример	Результат
В операторе `Load`: `GeoInvProjectGeometry ('mercator',AreaPolygon) as InvProjectGeometry`	Геометрия, загруженная как **AreaPolygon**, преобразуется методом обратного преобразования проекции Меркатора и сохраняется как **InvProjectGeometry** для использования в визуализациях.

GeoMakePoint

GeoMakePoint() используется в скриптах и выражениях диаграмм для создания и обозначения широты и долготы точки. Функция GeoMakePoint возвращает точки в порядке долготы и широты.

Синтаксис:

```
GeoMakePoint(lat_field_name, lon_field_name)
```

Возвращаемые типы данных: строка, с форматированием [долгота, широта]

Аргументы:

Аргументы

Аргумент	Описание
lat_field_name	Поле или выражение, относящееся к полю, в котором представлена широта точки.
lon_field_ name	Поле или выражение, относящееся к полю, в котором представлена долгота точки.

*Не используйте предложение **Group by** в редакторе загрузки данных с этой и другими неагрегирующими геопространственными функциями, так как это приведет к возникновению ошибки загрузки.*

GeoProject

GeoProject() используется в скриптах и выражениях диаграмм для применения проекции к геометрии.

Синтаксис:

```
GeoProject(type, field_name)
```

Возвращаемые типы данных: строка

Аргументы:

Аргументы

Аргумент	Описание
type	Тип проекции, используемый для преобразования геометрии карты. Может присутствовать одно из двух значений: «единица» (по умолчанию), которое создает проекцию 1:1, или «проекция Меркатора», которое использует стандартную проекцию Меркатора.
field_name	Поле или выражение, относящееся к полю, в котором содержится геометрия для представления. Может быть представлена в виде точки (или набора точек), задающей долготу и широту, или области.

*Не используйте предложение **Group by** в редакторе загрузки данных с этой и другими неагрегирующими геопространственными функциями, так как это приведет к возникновению ошибки загрузки.*

Пример:

Примеры скриптов

Пример	Результат
В операторе `Load`: `GeoProject ('mercator',Area) as GetProject`	Проекция Меркатора применяется к геометрии, загруженной в качестве **Area**, а результат сохраняется в качестве **GetProject**.

GeoProjectGeometry

GeoProjectGeometry() используется для агрегирования геометрии в область и применения проекции.

Синтаксис:

```
GeoProjectGeometry(type, field_name)
```

Возвращаемые типы данных: строка

Аргументы:

Аргументы

Аргумент	Описание
type	Тип проекции, используемый для преобразования геометрии карты. Может присутствовать одно из двух значений: «единица» (по умолчанию), которое создает проекцию 1:1, или «проекция Меркатора», которое использует стандартную проекцию Меркатора.
field_name	Поле или выражение, относящееся к полю, в котором содержится геометрия для представления. Может быть представлена в виде точки (или набора точек), задающей долготу и широту, или области.

Пример:

Пример	Результат
В операторе `Load`: `GeoProjectGeometry ('mercator',AreaPolygon) as ProjectGeometry`	Геометрия, загруженная как **AreaPolygon**, преобразуется методом проекции Меркатора и сохраняется как **ProjectGeometry** для использования в визуализациях.

GeoReduceGeometry

GeoReduceGeometry() используется для сокращения количества вершин геометрии и агрегирования нескольких областей в одну область с отображением границ отдельных областей.

Синтаксис:

```
GeoReduceGeometry(field_name[, value])
```

Возвращаемые типы данных: строка

Аргументы:

Аргументы

Аргумент	Описание
field_name	Поле или выражение, относящееся к полю, в котором содержится геометрия для представления. Может быть представлена в виде точки (или набора точек), задающей долготу и широту, или области.

Аргумент	Описание
value	Значение сокращения, которое необходимо применить к геометрии. В диапазон входят значения от 0 до 1, при этом значение 0 не влечет сокращения количества вершин, а значение 1 влечет максимальное сокращение. *В случае использования при работе со сложным набором данных значения value, равного 0,9 или больше, количество вершин сокращается до уровня, на котором визуальное отображение может быть неточным.*

GeoReduceGeometry() также выполняет функцию, схожую с **GeoAggrGeometry()**, агрегируя несколько областей в одну область. Различие заключается в том, что в случае использования **GeoReduceGeometry()** на карте отображаются линии отдельной границы из данных предварительного агрегирования.

Так как функция **GeoReduceGeometry()** является функцией агрегирования, для ее использования в скрипте требуется оператор **LOAD** с предложением **Group by**.

Примеры:
В данном примере показан порядок загрузки файла KML с данными области и таблицы с сокращенными и агрегированными данными области.

```
[MapSource]: LOAD [world.Name], [world.Point], [world.Area] FROM [lib://Downloads/world.kml]
(kml, Table is [World.shp/Features]); Map: LOAD world.Name, GeoReduceGeometry(world.Area,0.5)
as [ReducedArea] resident MapSource Group By world.Name;

 Drop Table MapSource;
```

5.15 Функции интерпретации

Функции интерпретации оценивают содержимое текстовых полей ввода или выражений и применяют указанный формат данных к полученному числовому значению. Эти функции позволяют указывать формат числа в соответствии с типом данных, включая такие атрибуты, как разделители разрядов и формат даты.

Функции интерпретации возвращают двойное значение, состоящее из строкового и числового значения, но могут использоваться для преобразования строки в число. Эти функции берут текстовое значение входного выражения и создают число, представляющую собой строку.

В отличие от них, функции форматирования делают все наоборот. Они берут числовые выражения и интерпретируют их в качестве строк, определяя формат полученного текста.

Если функции интерпретации не используются, программа Qlik Sense интерпретирует данные как комбинацию чисел, дат, времени, меток времени и строк с помощью настроек по умолчанию для формата чисел, даты и времени, заданных переменными скрипта и операционной системой.

Все функции интерпретации можно использовать как в скриптах загрузки данных, так и в выражениях диаграмм.

Во всех представлениях чисел в качестве десятичного разделителя используется десятичная точка.

Обзор функций интерпретации

Каждая функция подробно описана после обзора. Также можно щелкнуть имя функции в синтаксисе, чтобы получить немедленный доступ к подробной информации об этой конкретной функции.

Date#

Date# оценивает выражение в качестве даты в формате, указанном во втором аргументе (если указан). Если код формата не указан, используется формат даты, установленный в операционной системе по умолчанию.

```
Date# (page 679)(text[, format])
```

Interval#

Interval#() преобразует текстовое выражение в интервал времени в формате, установленном в операционной системе (по умолчанию) или в формате, указанном во втором аргументе, если имеется.

```
Interval# (page 680)(text[, format])
```

Money#

Money#() преобразует текстовую строку в денежное значение, используя формат, установленный в скрипте загрузки или в операционной системе (если не указана строка форматирования). Пользовательские символы разделителей десятичных разрядов и тысяч являются дополнительными параметрами.

```
Money# (page 681)(text[, format[, dec_sep[, thou_sep ] ] ])
```

Num#

Num() интерпретирует текстовую строку как числовое значение, то есть преобразует входную строку в число, используя формат, указанный во втором параметре. Если второй параметр опущен, то используются десятичные и тысячные разделители, установленные в скрипте загрузки данных. Пользовательские символы разделителей десятичных разрядов и тысяч являются дополнительными параметрами.

```
Num# (page 682)(text[ , format[, dec_sep[ , thou_sep]]])
```

Text

Text() преобразует выражение в текстовый вид даже при возможности обработки его в качестве числа.

```
Text(expr)
```

Time#

Time#() преобразует выражение в значение времени, используя формат, установленный в скрипте загрузки данных или в операционной системе (если не указана строка форматирования)..

```
Time# (page 683)(text[, format])
```

Timestamp#

Timestamp#() преобразует выражение в значение времени и даты, используя формат метки времени, установленный в скрипте загрузки данных или в операционной системе (если не указана строка форматирования).

```
Timestamp# (page 685)(text[, format])
```

См. также:

- *Функции форматирования (page 640)*

Date#

Date# оценивает выражение в качестве даты в формате, указанном во втором аргументе (если указан).

Синтаксис:

```
Date#(text[, format])
```

Возвращаемые типы данных: двойное значение

Аргументы:

Аргументы

Аргумент	Описание
text	Текстовая строка для оценки.
format	Строка, описывающая формат текстовой строки, подлежащей оценке. Если строка пропущена, формат даты устанавливается в системных переменных в скрипте загрузки данных или используются данные операционной системы.

Примеры и результаты:

В следующем примере используется формат даты **M/D/YYYY**. Формат даты указан в операторе **SET DateFormat** в верхней части скрипта загрузки данных.

Добавьте образец скрипта в свое приложение и запустите.

```
Load *,
Num(Date#(StringDate)) as Date;
LOAD * INLINE [
StringDate
8/7/97
8/6/1997
```

При создании таблицы с помощью **StringDate** и **Date** в качестве измерений результаты выглядят следующим образом:

Результаты

StringDate	Дата
8/7/97	35649
8/6/1997	35648

Interval#

Interval#() преобразует текстовое выражение в интервал времени в формате, установленном в операционной системе (по умолчанию) или в формате, указанном во втором аргументе, если имеется.

Синтаксис:

```
Interval#(text[, format])
```

Возвращаемые типы данных: двойное значение

Аргументы:

Аргументы

Аргумент	Описание
text	Текстовая строка для оценки.
format	Строка, описывающая ожидаемый формат ввода для использования при преобразовании строки в числовой интервал. Если пропущено, то используется краткий формат даты, формат времени и десятичный разделитель из операционной системы.

Функция **interval#** преобразует текстовый интервал времени в числовой.

Примеры и результаты:
В указанных ниже примерах используются следующие настройки операционной системы:

- Краткий формат даты: YY-MM-DD
- Формат времени: M/D/YY
- Десятичный разделитель числа: .

Результаты

Пример	Результат
`Interval#( A, 'D hh:mm' )` где A='1 09:00'	1.375

Money#

Money#() преобразует текстовую строку в денежное значение, используя формат, установленный в скрипте загрузки или в операционной системе (если не указана строка форматирования). Пользовательские символы разделителей десятичных разрядов и тысяч являются дополнительными параметрами.

Синтаксис:

```
Money#(text[, format[, dec_sep [, thou_sep ] ] ])
```

Возвращаемые типы данных: двойное значение

Аргументы:

Аргументы

Аргумент	Описание
text	Текстовая строка для оценки.
format	Строка, описывающая ожидаемый формат ввода для использования при преобразовании строки в числовой интервал. Если не указано, то используется денежный формат, заданный в операционной системе.
dec_sep	Строка, определяющая десятичный разделитель. Если не указано, в скрипте загрузки данных используется набор значений MoneyDecimalSep.
thou_sep	Строка, определяющая разделитель тысяч. Если не указано, в скрипте загрузки данных используется набор значений MoneyThousandSep.

Функция **money#** выполняется почти так же, как функция **num#**, но использует значения, заданные по умолчанию для разделителей десятичных дробей и тысяч в переменных скрипта для денежного формата, или соответствующие системные настройки для валюты.

Примеры и результаты:
В рассматриваемых ниже примерах предполагается использование двух следующих настроек операционной системы:

- Параметр формата денежных единиц по умолчанию 1: kr # ##0,00
- Параметр формата денежных единиц по умолчанию 2: $ #,##0.00

```
Money#(A , '# ##0,00 kr' )
```
, где A=35 648,37 kr

Результаты

Результаты	Параметр 1	Параметр 2
Строка	35 648,37 kr	35 648,37 kr
Число	35648.37	3564837

```
Money#( A, ' $#', '.', ',' )
```
, где A= $35 648,37

Результаты

Результаты	Параметр 1	Параметр 2
Строка	$35,648.37	$35,648.37
Число	35648.37	35648.37

Num#

Num() интерпретирует текстовую строку как числовое значение, то есть преобразует входную строку в число, используя формат, указанный во втором параметре. Если второй параметр опущен, то используются десятичные и тысячные разделители, установленные в скрипте загрузки данных. Пользовательские символы разделителей десятичных разрядов и тысяч являются дополнительными параметрами.

Синтаксис:

```
Num#(text[, format[, dec_sep [, thou_sep ] ] ])
```

Возвращаемые типы данных: двойное значение

Функция **Num#()** возвращает двойное значение, которое включает строковое и числовое значения. Функция берет текстовое представление входного выражения и создает число. Формат числа не изменяется: выходное число форматируется так же, как и входное.

Аргументы:

Аргументы

Аргумент	Описание
text	Текстовая строка для оценки.
format	Строка, указывающая формат числа, используемый в первом параметре. Если не указано, то используются десятичные и тысячные разделители, установленные в скрипте загрузки данных.
dec_sep	Строка, определяющая десятичный разделитель. Если не указано, используется значение переменной DecimalSep, установленной в скрипте загрузки данных.
thou_sep	Строка, определяющая разделитель тысяч. Если не указано, используется значение переменной ThousandSep, установленной в скрипте загрузки данных.

Примеры и результаты:
В следующей таблице показан результат *Num#(A, '#', '.' , ',')* для различных значений A.

	Результаты	
A	**Строковое представление**	**Числовое значение (здесь отображается с десятичной точкой)**
35,648.31	35,648.31	35648.31
35 648.312	35 648.312	35648.312
35.648,3123	35.648,3123	-
35 648,31234	35 648,31234	-

Text

Text() преобразует выражение в текстовый вид даже при возможности обработки его в качестве числа.

Синтаксис:

```
Text (expr)
```

Возвращаемые типы данных: двойное значение

Пример:

```
Text( A )
```
, где A=1234

Результаты	
Строка	**Число**
1234	-

Пример:

```
Text( pi( ) )
```

Результаты	
Строка	**Число**
3.1415926535898	-

Time#

Time#() преобразует выражение в значение времени, используя формат, установленный в скрипте загрузки данных или в операционной системе (если не указана строка форматирования)..

Синтаксис:

```
time#(text[, format])
```

Возвращаемые типы данных: двойное значение

Аргументы:

Аргументы

Аргумент	Описание
text	Текстовая строка для оценки.
format	Строка, описывающая формат текстовой строки, подлежащей оценке. Если пропущено, то используется краткий формат даты, формат времени и десятичный разделитель из операционной системы.

Пример:

- Параметр формата времени по умолчанию 1: hh:mm:ss
- Параметр формата времени по умолчанию 2: hh.mm.ss

```
time#( A )
```
где A=09:00:00

Результаты

Результаты	Параметр 1	Параметр 2
Строка:	09:00:00	09:00:00
Число:	0.375	-

Пример:

- Параметр формата времени по умолчанию 1: hh:mm:ss
- Параметр формата времени по умолчанию 2: hh.mm.ss

```
time#( A, 'hh.mm' )
```
где A=09.00

Результаты

Результаты	Параметр 1	Параметр 2
Строка:	09.00	09.00
Число:	0.375	0.375

Timestamp#

Timestamp#() преобразует выражение в значение времени и даты, используя формат метки времени, установленный в скрипте загрузки данных или в операционной системе (если не указана строка форматирования).

Синтаксис:

```
timestamp#(text[, format])
```

Возвращаемые типы данных: двойное значение

Аргументы:

Аргументы

Аргумент	Описание
text	Текстовая строка для оценки.
format	Строка, описывающая формат текстовой строки, подлежащей оценке. Если пропущено, то используется краткий формат даты, формат времени и десятичный разделитель из операционной системы. Для меток времени поддерживается ISO 8601.

Пример:

В следующем примере используется формат даты **M/D/YYYY**. Формат даты указан в операторе **SET DateFormat** в верхней части скрипта загрузки данных.

Добавьте образец скрипта в свое приложение и запустите.

```
Load *,
Timestamp(Timestamp#(String)) as TS;
LOAD * INLINE [
Строка
2015-09-15T12:13:14
1952-10-16T13:14:00+0200
1109-03-01T14:15
];
```

При создании таблицы с помощью **String** и **TS** в качестве измерений результаты выглядят следующим образом:

Результаты

Строка	TS
2015-09-15T12:13:14	9/15/2015 12:13:14 PM
1952-10-16T13:14:00+0200	10/16/1952 11:14:00 AM
1109-03-01T14:15	3/1/1109 2:15:00 PM

5.16 Функции между записями

Функции между записями используются:

- в скрипте загрузки данных, если для оценки текущей записи требуется значение из ранее загруженных записей данных;
- в выражении диаграммы, если требуется другое значение из набора данных визуализации.

Сортировка по y-значениям в диаграммах или сортировка по столбцам выражений в прямых таблицах не допускается, если в каком-либо из выражений диаграммы используются функции между записями диаграммы. Данные возможности сортировки автоматически отключаются.

Создание корректных определений выражений с рекурсивными ссылками возможно только в таблицах, содержащих менее 100 строк, однако это зависит от аппаратного обеспечения, используемого модулем Qlik.

Функции строки

Эти функции могут использоваться только в выражениях диаграмм.

Above

Функция **Above()** оценивает выражение в строке над текущей строкой в сегменте столбца в таблице. Строка, для которой выполняется вычисление, зависит от значения элемента **offset**, если таковой имеет место, по умолчанию принимается строка непосредственно над текущей строкой. Для диаграмм, за исключением таблиц, функция **Above()** используется для оценки строки над текущей строкой в эквиваленте прямой таблицы диаграммы.

```
Above — функция диаграммы([TOTAL [<fld{,fld}>]] expr [ , offset [,count]])
```

Below

Функция **Below()** оценивает выражение в строке под текущей строкой в сегменте столбца в таблице. Строка, для которой выполняется вычисление, зависит от значения элемента **offset**, если таковой имеет место, по умолчанию принимается строка непосредственно под текущей строкой. Для диаграмм, за исключением таблиц, функция **Below()** используется для оценки строки под текущим столбцом в эквиваленте прямой таблицы диаграммы.

```
Below — функция диаграммы([TOTAL[<fld{,fld}>]] expression [ , offset [,count
]])
```

Bottom

Функция **Bottom()** оценивает выражение в последней (нижней) строке сегмента столбца в таблице. Строка, для которой выполняется вычисление, зависит от значения элемента **offset**, если таковой имеет место, по умолчанию принимается нижняя строка. Для диаграмм, за исключением таблиц, оценка выполняется в последней строке текущего столбца в эквиваленте прямой таблицы

диаграммы.

```
Bottom — функция диаграммы([TOTAL[<fld{,fld}>]] expr [ , offset [,count ]])
```

Top

Функция **Top()** оценивает выражение в первой (верхней) строке сегмента столбца в таблице. Строка, для которой выполняется вычисление, зависит от значения элемента **offset**, если таковой имеет место, по умолчанию принимается верхняя строка. Для диаграмм, за исключением таблиц, функция **Top()** используется для оценки в первой строке текущего столбца в эквиваленте прямой таблицы диаграммы.

```
Top — функция диаграммы([TOTAL [<fld{,fld}>]] expr [ , offset [,count ]])
```

NoOfRows

Функция **NoOfRows()** возвращает строки в текущий сегмент столбца в таблице. Для растровых диаграмм функция **NoOfRows()** возвращает строки в эквивалент прямой таблицы диаграммы.

```
NoOfRows — функция диаграммы([TOTAL])
```

Функции столбца

Эти функции могут использоваться только в выражениях диаграмм.

Column

Функция **Column()** возвращает значение, обнаруженное в столбце, соответствующем элементу **ColumnNo**, в прямую таблицу без учета измерений. Например, элемент **Column(2)** возвращает значение второго столбца мер.

```
Column — функция диаграммы(ColumnNo)
```

Dimensionality

Функция **Dimensionality()** возвращает измерения для текущей строки. В случае со сводными таблицами эта функция возвращает итоговое число столбцов измерений, имеющих неагрегированное содержимое, т. е. не содержащих частичных сумм или свернутых агрегированных показателей.

```
Dimensionality — функция диаграммы ( )
```

Secondarydimensionality

Функция **SecondaryDimensionality()** возвращает количество строк измерений сводной таблицы, имеющих неагрегированное содержимое, т. е. не содержащих частичных сумм или свернутых агрегированных показателей. Данная функция является эквивалентом функции **dimensionality()** для горизонтальных измерений сводной таблицы.

```
SecondaryDimensionality — функция диаграммы ( )
```

Функции поля

FieldIndex

Функция **FieldIndex()**возвращает позицию значения поля **value** в поле **field_name** (в порядке загрузки).

```
FieldIndex (field_name , value)
```

FieldValue

Функция **FieldValue()** возвращает значение, находящееся в позиции **elem_no** поля **field_name** (в порядке загрузки).

```
FieldValue (field_name , elem_no)
```

FieldValueCount

Функция **FieldValueCount()** – это функция **целого числа**, которая возвращает уникальные значения в поле.

```
FieldValueCount (field_name)
```

Функции сводной таблицы

Эти функции могут использоваться только в выражениях диаграмм.

After

Функция **After()** возвращает значение выражения, оцененного со значениями измерения сводной таблицы по мере их отображения в столбце после текущего столбца в сегменте строки сводной таблицы.

```
After — функция диаграммы([TOTAL] expression [ , offset [,n]])
```

Before

Функция **Before()** возвращает значение выражения, оцененного со значениями измерения сводной таблицы по мере их отображения в столбце перед текущим столбцом в сегменте строки сводной таблицы.

```
Before — функция диаграммы([TOTAL] expression [ , offset [,n]])
```

First

Функция **First()** возвращает значение выражения, оцененного со значениями измерения сводной таблицы по мере их отображения в первом столбце текущего сегмента строки сводной таблицы. Данная функция возвращает значение NULL во всех типах диаграмм, кроме сводных таблиц.

```
First — функция диаграммы([TOTAL] expression [ , offset [,n]])
```

Last

Функция **Last()** возвращает значение выражения, оцененного со значениями измерения сводной таблицы по мере их отображения в последнем столбце текущего сегмента строки сводной таблицы. Данная функция возвращает значение NULL во всех типах диаграмм, кроме сводных таблиц.

```
Last — функция диаграммы([TOTAL] expression [ , offset [,n]])
```

ColumnNo

Функция **ColumnNo()** возвращает количество текущих столбцов в текущем сегменте строки сводной таблицы. Первый столбец имеет номер 1.

```
ColumnNo — функция диаграммы([TOTAL])
```

NoOfColumns
Функция **NoOfColumns()** возвращает количество столбцов в текущем сегменте строки сводной таблицы.

```
NoOfColumns — функция диаграммы([TOTAL])
```

Функции между записями в скрипте загрузки данных

Exists

Функция **Exists()** определяет, загружено ли определенное значение поля в поле в скрипте загрузки данных. Функция возвращает значение TRUE или FALSE, таким образом, ее можно использовать в предложении **where** оператора **LOAD** или **IF**.

```
Exists (field_name [, expr])
```

LookUp

Функция **Lookup()** просматривает загруженную таблицу и возвращает значение поля **field_name**, соответствующее первому вхождению значения **match_field_value** в поле **match_field_name**. Таблица может быть текущей таблицей или другой ранее загруженной таблицей.

```
LookUp (field_name, match_field_name, match_field_value [, table_name])
```

Peek

Функция **Peek()** возвращает значение поля в таблице для строки, которая уже загружена. Можно указать номер строки или таблицу. Если номер строки не указан, будет использована последняя запись, загруженная ранее.

```
Peek (field_name[, row_no[, table_name ] ])
```

Previous

Функция **Previous()** находит значение выражения **expr** с помощью данных из ранее введенной записи, которая не была сброшена из-за предложения **where**. В первой записи внутренней таблицы функция возвратит значение NULL.

```
Previous (page 722)(expr)
```

См. также:

Функции над выборкой (page 743)

Above – функция диаграммы

Функция **Above()** оценивает выражение в строке над текущей строкой в сегменте столбца в таблице. Строка, для которой выполняется вычисление, зависит от значения элемента **offset**, если таковой имеет место, по умолчанию принимается строка непосредственно над текущей строкой. Для диаграмм, за исключением таблиц, функция **Above()** используется для оценки строки над текущей строкой в эквиваленте прямой таблицы диаграммы.

Синтаксис:

```
Above([TOTAL] expr [ , offset [,count]])
```

Возвращаемые типы данных: двойное значение

Аргументы:

Аргументы

Аргумент	Описание
expr	Выражение или поле, содержащее данные для измерения.
offset	Если задать значение offsetn больше 0, можно будет переместить оценку выражения n по строкам выше текущей строки. Если задать смещение равным 0, оценка выражения будет выполнена в текущей строке. Если задать отрицательное число смещения, функция Above будет работать как функция Below с соответствующим положительным числом смещения.
count	Если задать для третьего аргумента **count** значение больше 1, функция вернет диапазон значений элемента **count**: по одному для каждой строки таблицы элемента **count**, считая вверх от исходной ячейки. В данной форме функция может использоваться в качестве аргумента для любой специальной функции интервала. *Функции над выборкой (page 743)*
TOTAL	Если таблица имеет одно измерение, или если в качестве аргумента используется префикс **TOTAL**, текущий сегмент столбца всегда равен всему столбцу.

В первой строке сегмента столбца возвращено значение NULL, так как над этой строкой нет других строк.

Сегмент столбца определяется как последовательное подмножество ячеек с теми же значениями для измерений в текущем порядке сортировки. Межзаписные функции диаграмм выполняют вычисления в сегменте столбца за исключением крайнего правого измерения в эквивалентной прямой таблице. Если в диаграмме есть только одно измерение, или если указан квалификатор TOTAL, выражение оценивается по всей таблице.

Если таблица или эквивалент таблицы имеют несколько вертикальных измерений, текущий сегмент столбца будет включать только строки с теми же значениями, что и текущая строка во всех столбцах измерений, кроме столбца с последним измерением в межполевом порядке сортировки.

Ограничения:

Рекурсивные вызовы возвращают значение NULL.

Примеры и результаты:

Example 1:

Визуализация таблицы для примера 1

Customer	Sum([Sales])	Above(Sum(Sales))	Sum(Sales)+Above(Sum(Sales))	Above offset 3	Higher?
	2566	-	-	-	-
Astrida	587	-	-	-	-
Betacab	539	587	1126	-	-
Canutility	683	539	1222	-	Higher
Divadip	757	683	1440	1344	Higher

На снимке таблицы, показанной в этом примере, визуализация таблицы создана из измерения **Customer** и мер: `Sum(Sales)` и `Above(Sum(Sales))`.

Столбец Above(Sum(Sales)) возвращает значение NULL для строки **Customer**, содержащей элемент **Astrida**, так как над этой строкой нет других строк. В результате для строки **Betacab** показано значение элемента Sum(Sales) для элемента **Astrida**, в результате для строки **Canutility** показано значение для элемента **Sum(Sales)** для строки **Betacab** и так далее.

Для столбца, помеченного как Sum(Sales)+Above(Sum(Sales)), в строке для элемента **Betacab** показан результат добавления значений **Sum(Sales)** в строки **Betacab** + **Astrida** (539+587). В результате для строки **Betacab** будет показан результат добавления значений **Sum(Sales)** в строки **Canutility** + **Canutility** (683+539).

Меры, помеченные как Above offset 3, созданные с помощью выражения `Sum(Sales)+Above(Sum(Sales), 3)`, имеют аргумент **offset**, установленный на 3, и эффект выбора значения в строке на три строки выше текущей строки. Таким образом, добавляется значение **Sum(Sales)** для текущего элемента **Customer** к значению для элемента **Customer** на три строки выше. Значения, возвращенные для первых трех строк **Customer**, являются нулевыми.

В таблице также показаны более сложные меры: одна, созданная из элемента `Sum(Sales)+Above(Sum(Sales))`, а другая, помеченная как **Higher?**, созданная из элемента `IF(Sum(Sales)>Above(Sum(Sales)), 'Higher')`.

Эту функцию можно также использовать в диаграммах, кроме таблиц, например, в линейчатых диаграммах.

Для других типов диаграмм преобразуйте диаграмму в эквивалент прямой таблицы, чтобы можно было легко интерпретировать соотношение строк и функций.

Example 2:

На снимках таблиц, показанных в этом примере, к визуализациям добавлено больше измерений: **Month** и **Product**. Для диаграмм с несколькими измерениями результаты выражений, содержащих функции **Above**, **Below**, **Top** и **Bottom**, зависят от порядка, в котором измерения столбцов сортируются Qlik Sense. Программа Qlik Sense оценивает функции на основе сегментов столбца, полученных из измерения, отсортированного последним. Контроль за порядком сортировки столбцов осуществляется на панели свойств под элементом **Сортировка**. Этот порядок не обязательно соответствует порядку отображения столбцов в таблице.

На следующем снимке визуализации таблицы для примера 2 последним отсортированным измерением является **Month**, поэтому функция **Above** выполняет оценку на основе месяцев. Существует серия результатов для каждого значения **Product** для каждого месяца (от **Jan** до **Aug**) – сегмент столбца. За этим сегментом следует серия для другого сегмента столбца: для каждого элемента **Month** для следующего элемента **Product**. Будет указан сегмент столбца для каждого значения **Customer** для каждого элемента **Product**.

Визуализация таблицы для примера 2

Customer	Product	Month	Sum([Sales])	Above(Sum(Sales))
			2566	-
Astrida	AA	Jan	46	-
Astrida	AA	Feb	60	46
Astrida	AA	Mar	70	60
Astrida	AA	Apr	13	70
Astrida	AA	May	78	13
Astrida	AA	Jun	20	78
Astrida	AA	Jul	45	20
Astrida	AA	Aug	65	45

Example 3:

На снимке визуализации таблицы для примера 3 последним отсортированным измерением является **Product**. Это выполняется путем перемещения измерения Product в позицию 3 на вкладке «Сортировка» на панели свойств. Функция **Above** оценивается для каждого элемента **Product**, и поскольку существует только два продукта, **AA** и **BB**, в каждой серии будет выдан только один результат, не являющийся нулевым. В строке **BB** для месяца **Jan** значение для элемента **Above (Sum(Sales))** равно 46. Для строки **AA** значение нулевое. Значение в каждой строке **AA** для любого месяца всегда будет нулевым, поскольку отсутствует значение элемента **Product** над строкой AA. Вторая серия оценивается в строках **AA** и **BB** для месяца **Feb** для значения **Customer**, **Astrida**. Если все месяцы для значения **Astrida** оценены, эта последовательность повторяется для второго значения **Customer**Betacab и так далее.

Визуализация таблицы для примера 3

Customer	Product	Month	Sum([Sales])	Above(Sum(Sales))
			2566	-
Astrida	AA	Jan	46	-
Astrida	BB	Jan	46	46
Astrida	AA	Feb	60	-
Astrida	BB	Feb	60	60
Astrida	AA	Mar	70	-
Astrida	BB	Mar	70	70
Astrida	AA	Apr	13	-
Astrida	BB	Apr	13	13

Пример 4

Example 4:	**Результат**
Функцию Above можно использовать как ввод в функции над выборкой. Например, элемент `RangeAvg (Above(Sum (Sales),1,3))`.	В аргументах для функции Above() для элемента offset задано значение 1, а для элемента count задано значение 3. Функция находит результаты выражения Sum(Sales) в трех строках непосредственно над текущей строкой в сегменте столбца (если есть строка). Эти три значения используются как ввод в функцию RangeAvg(), которая находит среднее значение в предоставленном диапазоне чисел. Таблица с элементом Customer в виде измерения выдает следующие результаты для выражения RangeAvg(). Astrida - Betacab 587 Canutility 563 Divadip: 603

Данные, используемые в примерах:

```
Monthnames:
LOAD *, Dual(MonthText,MonthNumber) as Month INLINE [
MonthText, MonthNumber
Jan, 1
Feb, 2
Mar, 3
Apr, 4
May, 5
Jun, 6
Jul, 7
```

```
Aug, 8
Sep, 9
Oct, 10
Nov, 11
Dec, 12
];

Sales2013:
Crosstable (MonthText, Sales) LOAD * inline [
Customer|Jan|Feb|Mar|Apr|May|Jun|Jul|Aug|Sep|Oct|Nov|Dec
Astrida|46|60|70|13|78|20|45|65|78|12|78|22
Betacab|65|56|22|79|12|56|45|24|32|78|55|15
Canutility|77|68|34|91|24|68|57|36|44|90|67|27
Divadip|57|36|44|90|67|27|57|68|47|90|80|94
] (delimiter is '|');
```

См. также:

- *Below – функция диаграммы (page 694)*
- *Bottom – функция диаграммы (page 698)*
- *Top – функция диаграммы (page 723)*
- *RangeAvg (page 746)*

Below – функция диаграммы

Функция **Below()** оценивает выражение в строке под текущей строкой в сегменте столбца в таблице. Строка, для которой выполняется вычисление, зависит от значения элемента **offset**, если таковой имеет место, по умолчанию принимается строка непосредственно под текущей строкой. Для диаграмм, за исключением таблиц, функция **Below()** используется для оценки строки под текущим столбцом в эквиваленте прямой таблицы диаграммы.

Синтаксис:

```
Below([TOTAL] expr [ , offset [,count ]])
```

Возвращаемые типы данных: двойное значение

Аргументы:

Аргументы

Аргумент	Описание
expr	Выражение или поле, содержащее данные для измерения.
offset	Если задать значение **offset**n больше 1, можно будет переместить оценку выражения n по строкам ниже текущей строки. Если задать смещение равным 0, оценка выражения будет выполнена в текущей строке. Если задать отрицательное число смещения, функция **Below** будет работать как функция **Above** с соответствующим положительным числом смещения.

Аргумент	Описание
count	Если задать для третьего параметра **count** значение больше 1, функция вернет диапазон значений элемента **count**: по одному для каждой строки таблицы элемента **count**, считая вниз от исходной ячейки. В данной форме функция может использоваться в качестве аргумента для любой специальной функции интервала. *Функции над выборкой (page 743)*
TOTAL	Если таблица имеет одно измерение, или если в качестве аргумента используется префикс **TOTAL**, текущий сегмент столбца всегда равен всему столбцу.

В последней строке сегмента столбца возвращено значение NULL, так как под этой строкой нет других строк.

Сегмент столбца определяется как последовательное подмножество ячеек с теми же значениями для измерений в текущем порядке сортировки. Межзаписные функции диаграмм выполняют вычисления в сегменте столбца за исключением крайнего правого измерения в эквивалентной прямой таблице. Если в диаграмме есть только одно измерение, или если указан квалификатор TOTAL, выражение оценивается по всей таблице.

Если таблица или эквивалент таблицы имеют несколько вертикальных измерений, текущий сегмент столбца будет включать только строки с теми же значениями, что и текущая строка во всех столбцах измерений, кроме столбца с последним измерением в межполевом порядке сортировки.

Ограничения:

Рекурсивные вызовы возвращают значение NULL.

Примеры и результаты:

Example 1:

Визуализация таблицы для примера 1

Customer	Sum([Sales])	Below(Sum(Sales))	Sum(Sales)+Below(Sum(Sales))	Below + Offset 3	Higher
	2566	-	-	-	-
Astrida	587	539	1126	1344	Higher
Betacab	539	683	1222	-	-
Canutility	683	757	1440	-	-
Divadip	757	-	-	-	-

В таблице, показанной на снимке для примера 1, визуализация таблицы создана из измерения **Customer** и мер: `Sum(Sales)` и `Below(Sum(Sales))`.

Столбец **Below(Sum(Sales))** возвращает значение NULL для строки **Customer**, содержащей элемент **Divadip**, так как под этой строкой нет других строк. В результате для строки **Canutility** показано значение элемента Sum(Sales) для элемента **Divadip**, в результате для строки **Betacab** показано значение для элемента **Sum(Sales)** для строки **Canutility** и так далее.

В таблице также показаны более сложные меры, которые можно увидеть в столбцах, помеченных как: `Sum(Sales)+Below(Sum(Sales))`, **Below +Offset 3** и **Higher?**. Эти выражения работают как описано в следующих абзацах.

Для столбца, помеченного как **Sum(Sales)+Below(Sum(Sales))**, в строке для элемента **Astrida** показан результат добавления значений **Sum(Sales)** в строки **Betacab** + **Astrida** (539+587). В результате для строки **Betacab** будет показан результат добавления значений **Sum(Sales)** в строки **Canutility** + **Betacab** (539+683).

Для мер, помеченных как **Below +Offset 3**, созданных с помощью выражения `Sum(Sales)+Below(Sum(Sales), 3)`, аргумент **offset** установлен на 3 и опускает значение в строке на три строки ниже текущей. Таким образом, добавляется значение **Sum(Sales)** для текущего элемента **Customer** к значению из элемента **Customer** на три строки ниже. Значения для нижних трех строк **Customer** являются нулевыми.

Мера, помеченная как **Higher?**, создается из выражения: `IF(Sum(Sales)>Below(Sum(Sales)), 'Higher')`. Таким образом сравниваются значения текущей строки в мере **Sum(Sales)** со значениями строки под этой строкой. Если текущая строка представляет большее значение, выходными данными является текст «Higher».

Эту функцию можно также использовать в диаграммах, кроме таблиц, например, в линейчатых диаграммах.

Для других типов диаграмм преобразуйте диаграмму в эквивалент прямой таблицы, чтобы можно было легко интерпретировать соотношение строк и функций.

Для диаграмм с несколькими измерениями результаты выражений, содержащих функции **Above**, **Below**, **Top** и **Bottom**, зависят от порядка, в котором измерения столбцов сортируются Qlik Sense. Программа Qlik Sense оценивает функции на основе сегментов столбца, полученных из измерения, отсортированного последним. Контроль за порядком сортировки столбцов осуществляется на панели свойств под элементом **Сортировка**. Этот порядок не обязательно соответствует порядку отображения столбцов в таблице.Дополнительную информацию см. в примере 2 для функции **Above**.

Пример 2.

Example 2:	Результат
Функцию **Below** можно использовать как ввод в функции над выборкой. Например, элемент `RangeAvg (Below(Sum(Sales),1,3))`.	В аргументах для функции **Below()** для элемента offset задано значение 1, а для элемента count задано значение 3. Функция находит результаты выражения **Sum(Sales)** в трех строках непосредственно под текущей строкой в сегменте столбца (если есть строка). Эти три значения используются как ввод в функцию RangeAvg(), которая находит среднее значение в предоставленном диапазоне чисел. Таблица с элементом **Customer** в виде измерения выдает следующие результаты для выражения RangeAvg().
	Astrida 659,67 Betacab 720 Canutility 757 Divadip: -

Данные, используемые в примерах:

```
Monthnames:
LOAD *, Dual(MonthText,MonthNumber) as Month INLINE [
MonthText, MonthNumber
Jan, 1
Feb, 2
Mar, 3
Apr, 4
May, 5
Jun, 6
Jul, 7
Aug, 8
Sep, 9
Oct, 10
Nov, 11
Dec, 12
];

Sales2013:
Crosstable (MonthText, Sales) LOAD * inline [
Customer|Jan|Feb|Mar|Apr|May|Jun|Jul|Aug|Sep|Oct|Nov|Dec
Astrida|46|60|70|13|78|20|45|65|78|12|78|22
Betacab|65|56|22|79|12|56|45|24|32|78|55|15
Canutility|77|68|34|91|24|68|57|36|44|90|67|27
Divadip|57|36|44|90|67|27|57|68|47|90|80|94
] (delimiter is '|');
```

См. также:

- *Above – функция диаграммы (page 689)*
- *Bottom – функция диаграммы (page 698)*
- *Top – функция диаграммы (page 723)*
- *RangeAvg (page 746)*

Bottom – функция диаграммы

Функция **Bottom()** оценивает выражение в последней (нижней) строке сегмента столбца в таблице. Строка, для которой выполняется вычисление, зависит от значения элемента **offset**, если таковой имеет место, по умолчанию принимается нижняя строка. Для диаграмм, за исключением таблиц, оценка выполняется в последней строке текущего столбца в эквиваленте прямой таблицы диаграммы.

Синтаксис:

```
Bottom([TOTAL] expr [ , offset [,count ]])
```

Возвращаемые типы данных: двойное значение

Аргументы:

Аргументы

Аргумент	Описание
expr	Выражение или поле, содержащее данные для измерения.
offset	Если задать значение **offset**n больше 1, можно будет переместить оценку выражения по n строкам выше нижней строки. Если задать отрицательное число смещения, функция **Bottom** будет работать как функция **Top** с соответствующим положительным числом смещения.
count	Если задать для третьего параметра **count** значение больше 1, функция вернет не одно, а ряд значений элемента **count**: по одному для каждой последней строки элемента **count** текущего сегмента столбца. В данной форме функция может использоваться в качестве аргумента для любой специальной функции интервала. *Функции над выборкой (page 743)*
TOTAL	Если таблица имеет одно измерение, или если в качестве аргумента используется префикс **TOTAL**, текущий сегмент столбца всегда равен всему столбцу.

Сегмент столбца определяется как последовательное подмножество ячеек с теми же значениями для измерений в текущем порядке сортировки. Межзаписные функции диаграмм выполняют вычисления в сегменте столбца за исключением крайнего правого измерения в эквивалентной прямой таблице. Если в диаграмме есть только одно измерение, или если указан квалификатор TOTAL, выражение оценивается по всей таблице.

Если таблица или эквивалент таблицы имеют несколько вертикальных измерений, текущий сегмент столбца будет включать только строки с теми же значениями, что и текущая строка во всех столбцах измерений, кроме столбца с последним измерением в межполевом порядке сортировки.

Ограничения:

Рекурсивные вызовы возвращают значение NULL.

Примеры и результаты:

Визуализация таблицы для примера 1

Customer	Sum([Sales])	Bottom(Sum(Sales))	Sum(Sales)+Bottom(Sum(Sales))	Bottom offset 3
	2566	**757**	**3323**	**3105**
Astrida	587	757	1344	1126
Betacab	539	757	1296	1078
Canutility	683	757	1440	1222
Divadip	757	757	1514	1296

На снимке таблицы, показанной в этом примере, визуализация таблицы создана из измерения **Customer** и мер: `Sum(Sales)` и `Bottom(Sum(Sales))`.

Столбец **Bottom(Sum(Sales))** возвращает значение 757 для всех строк, поскольку это значение нижней строки: **Divadip**.

В таблице также показаны более сложные меры: одна, созданная из элемента `Sum(Sales)+Bottom(Sum(Sales))`, а другая, помеченная как **Bottom offset 3**, созданная с помощью выражения `Sum(Sales)+Bottom(Sum(Sales), 3)`, и имеющая аргумент **offset**, установленный на 3. Таким образом добавляется значение **Sum(Sales)** для текущей строки к значению из третьей строки от нижней строки, т. е. текущая строка плюс значение для элемента**Betacab**.

Пример: 2

На снимках таблиц, показанных в этом примере, к визуализациям добавлено больше измерений: **Month** и **Product**. Для диаграмм с несколькими измерениями результаты выражений, содержащих функции **Above**, **Below**, **Top** и **Bottom**, зависят от порядка, в котором измерения столбцов

сортируются Qlik Sense. Программа Qlik Sense оценивает функции на основе сегментов столбца, полученных из измерения, отсортированного последним. Контроль за порядком сортировки столбцов осуществляется на панели свойств под элементом **Сортировка**. Этот порядок не обязательно соответствует порядку отображения столбцов в таблице.

В первой таблице выражение оценивается на основе элемента **Month**, а во второй таблице оно основывается на элементе **Product**. Мера **End value** содержит выражение `Bottom(Sum(Sales))`. Нижней строкой для измерения **Month** является Dec, а значением для Dec, как и для обоих значений элемента **Product** показанных на снимке, является 22. (Некоторые строки были исключены из снимков при редактировании, чтобы сэкономить место.)

Первая таблица для примера 2. Значение элемента Bottom для меры End value основано на элементе Month (Dec).

Customer	Product	Month	Sum(Sales)	End value
			2566	-
Astrida	AA	Jan	46	22
Astrida	AA	Feb	60	22
Astrida	AA	Mar	70	22
Astrida	AA	Sep	78	22
Astrida	AA	Oct	12	22
Astrida	AA	Nov	78	22
Astrida	AA	Dec	22	22
Astrida	BB	Jan	46	22

Вторая таблица для примера 2. Значение элемента Bottom для меры End value основано на элементе Product (BB для Astrida).

Customer	Product	Month	Sum(Sales)	End value
			2566	-
Astrida	AA	Jan	46	46
Astrida	BB	Jan	46	46
Astrida	AA	Feb	60	60
Astrida	BB	Feb	60	60
Astrida	AA	Mar	70	70
Astrida	BB	Mar	70	70
Astrida	AA	Apr	13	13
Astrida	BB	Apr	13	13

Дополнительную информацию см. в примере 2 для функции **Above**.

Пример 3

Пример: 3	Результат
Функцию **Bottom** можно использовать как ввод в функции над выборкой. Например, элемент `RangeAvg (Bottom(Sum(Sales),1,3))`.	В аргументах для функции **Bottom()** для элемента offset задано значение 1, а для элемента count задано значение 3. Функция находит результаты выражения **Sum(Sales)** в трех строках, начиная со строки над нижней строкой в сегменте столбца (поскольку offset=1) и в двух строках над ней (если есть строка). Эти три значения используются как ввод в функцию RangeAvg(), которая находит среднее значение в предоставленном диапазоне чисел. Таблица с элементом **Customer** в виде измерения выдает следующие результаты для выражения RangeAvg().
	Astrida 659,67 Betacab 659,67 Canutility 659,67 Divadip: 659,67

```
Monthnames:
LOAD *, Dual(MonthText,MonthNumber) as Month INLINE [
MonthText, MonthNumber
Jan, 1
Feb, 2
Mar, 3
Apr, 4
May, 5
Jun, 6
Jul, 7
Aug, 8
Sep, 9
Oct, 10
Nov, 11
Dec, 12
];

Sales2013:
Crosstable (MonthText, Sales) LOAD * inline [
Customer|Jan|Feb|Mar|Apr|May|Jun|Jul|Aug|Sep|Oct|Nov|Dec
Astrida|46|60|70|13|78|20|45|65|78|12|78|22
Betacab|65|56|22|79|12|56|45|24|32|78|55|15
Canutility|77|68|34|91|24|68|57|36|44|90|67|27
Divadip|57|36|44|90|67|27|57|68|47|90|80|94
] (delimiter is '|');
```

См. также:

- *Top – функция диаграммы (page 723)*

Column – функция диаграммы

Функция **Column()** возвращает значение, обнаруженное в столбце, соответствующем элементу **ColumnNo**, в прямую таблицу без учета измерений. Например, элемент **Column(2)** возвращает значение второго столбца мер.

Синтаксис:

```
Column(ColumnNo)
```

Возвращаемые типы данных: двойное значение

Аргументы:

Аргументы

Аргумент	Описание
ColumnNo	Номер столбца в таблице, содержащей меру. *Функция Column() игнорирует столбцы измерений.*

Ограничения:

Если элемент **ColumnNo** ссылается на столбец, для которого нет мер, возвращается значение NULL.

Рекурсивные вызовы возвращают значение NULL.

Примеры и результаты:

Пример: Процентное соотношение итоговых продаж

Customer	Product	UnitPrice	UnitSales	Order Value	Total Sales Value	% Sales
A	AA	15	10	150	505	29.70
A	AA	16	4	64	505	12.67
A	BB	9	9	81	505	16.04
B	BB	10	5	50	505	9.90
B	CC	20	2	40	505	7.92

Customer	Product	UnitPrice	UnitSales	Order Value	Total Sales Value	% Sales
B	DD	25	-	0	505	0.00
C	AA	15	8	120	505	23.76
C	CC	19	-	0	505	0.00

Пример: Процентное соотношение продаж для выбранного клиента

Customer	Product	UnitPrice	UnitSales	Order Value	Total Sales Value	% Sales
A	AA	15	10	150	295	50.85
A	AA	16	4	64	295	21.69
A	BB	9	9	81	295	27.46

Примеры и результаты

Примеры	Результаты
Элемент Order Value добавляется к таблице в качестве меры с выражением: `Sum (UnitPrice*UnitSales)`. Элемент Total Sales Value добавляется как мера с выражением: `Sum(TOTAL UnitPrice*UnitSales)` Элемент % Sales добавляется как мера с выражением `100*Column (1)/Column(2)`	Результат элемента Column(1) взят из столбца Order Value, поскольку это первый столбец с мерами. Результат элемента Column(2) взят из столбца Total Sales Value, поскольку это второй столбец с мерами. См. результат в столбце % Sales в примере *Процентное соотношение итоговых продаж (page 702)*.
Выполните выборку Customer A.	Выборка изменяет элемент Total Sales Value и, следовательно, элемент %Sales. См. пример *Процентное соотношение продаж для выбранного клиента (page 703)*.

Данные, используемые в примерах:

```
ProductData:
LOAD * inline [
Customer|Product|UnitSales|UnitPrice
Astrida|AA|4|16
Astrida|AA|10|15
Astrida|BB|9|9
Betacab|BB|5|10
Betacab|CC|2|20
Betacab|DD||25
Canutility|AA|8|15
Canutility|CC||19
```

```
] (delimiter is '|');
```

Dimensionality – функция диаграммы

Функция **Dimensionality()** возвращает измерения для текущей строки. В случае со сводными таблицами эта функция возвращает итоговое число столбцов измерений, имеющих неагрегированное содержимое, т. е. не содержащих частичных сумм или свернутых агрегированных показателей.

Синтаксис:

```
Dimensionality ( )
```

Возвращаемые типы данных: целое число

Ограничения:

Данная функция доступна только в диаграммах. Для всех типов диаграмм, кроме сводной таблицы, будет возвращено количество измерений во всех строках, за исключением общей, которая будет равна 0.

Пример: Выражение диаграммы с использованием Dimensionality

Пример: Выражение диаграммы
Функцию Dimensionality() можно использовать со сводной таблицей в качестве выражения диаграммы, когда требуется применять разное форматирование ячеек в зависимости от количества измерений в строке, содержащей неагрегированные данные. В этом примере функция Dimensionality() используется для применения черного фона к ячейкам таблицы, отвечающим заданному условию.

Скрипт загрузки

Загрузите следующие данные через встроенную загрузку в редакторе загрузки данных, чтобы создать пример с выражениями диаграммы, показанный ниже.

```
ProductSales:  Load * inline [ Country,Product,Sales,Budget Sweden,AA,100000,50000
Germany,AA,125000,175000 Canada,AA,105000,98000 Norway,AA,74850,68500 Ireland,AA,49000,48000
Sweden,BB,98000,99000 Germany,BB,115000,175000 Norway,BB,71850,68500 Ireland,BB,31000,48000 ]
(delimiter is ',');
```

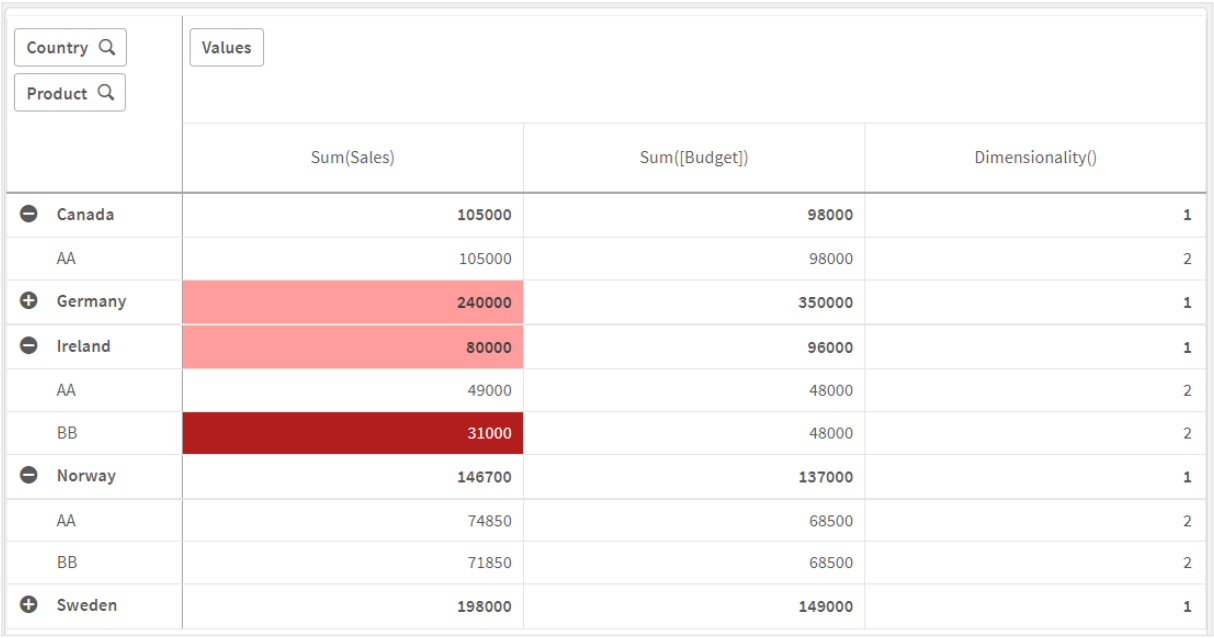
Выражение диаграммы

Создайте на листе Qlik Sense визуализацию сводной таблицы с измерениями **Country** и **Product**. Добавьте меры **Sum(Sales)**, **Sum(Budget)** и **Dimensionality()**.

На панели **Свойства** введите следующее выражение в поле **Выражение для цвета фона** для меры **Sum(Sales)**:

```
If(Dimensionality()=1 and Sum(Sales)<Sum(Budget),RGB(255,156,156), If(Dimensionality()=2 and
Sum(Sales)<Sum(Budget),RGB(178,29,29) ))
```

Результат:



Country / Product	Sum(Sales)	Sum([Budget])	Dimensionality()
Canada	105000	98000	1
AA	105000	98000	2
Germany	240000	350000	1
Ireland	80000	96000	1
AA	49000	48000	2
BB	31000	48000	2
Norway	146700	137000	1
AA	74850	68500	2
BB	71850	68500	2
Sweden	198000	149000	1

Объяснение

Выражение `If(Dimensionality()=1 and Sum(Sales)<Sum(Budget),RGB(255,156,156), If (Dimensionality()=2 and Sum(Sales)<Sum(Budget),RGB(178,29,29)))` содержит условные операторы, которые проверяют значение размерности, а также Sum(Sales) и Sum(Budget) для каждого продукта. Если условия соблюдаются, фоновый цвет применяется к значению Sum(Sales).

Exists

Функция **Exists()** определяет, загружено ли определенное значение поля в поле в скрипте загрузки данных. Функция возвращает значение TRUE или FALSE, таким образом, ее можно использовать в предложении **where** оператора **LOAD** или **IF**.

> *Также для определения случаев, когда значение поля загружено не было, можно использовать функцию **Not Exists()**. Однако **Not Exists()** в предложении Where следует использовать с осторожностью. Функция **Exists()** проверяет ранее загруженные таблицы и ранее загруженные значения текущей таблицы. Таким образом, загружается только первое вхождение. На момент обнаружения второго вхождения значение уже загружено. Для получения дополнительной информации см. примеры.*

Синтаксис:

```
Exists(field_name [, expr])
```

Возвращаемые типы данных: Булево значение

Аргументы:

Аргументы

Аргумент	Описание
field_name	Имя поля, в котором необходимо выполнить поиск значения. Можно использовать явное имя поля без кавычек. Поле должно быть уже загружено при помощи скрипта. Это означает, что нельзя ссылаться на поле, загруженное предложением далее по скрипту.
expr	Значение, которое необходимо проверить, при условии его существования. Можно использовать явное значение или выражение, которое ссылается на одно или несколько полей текущего оператора load. *Нельзя ссылаться на поля, не включенные в текущий оператор load.* Данный аргумент является дополнительным. Если его пропустить, функция будет проверять, существует ли значение **field_name** в текущей записи.

Примеры и результаты:

Пример 1.

```
Exists (Employee)
```

Возвращает -1 (True), если значение поля **Employee** в текущей записи уже существует в любой ранее прочитанной записи, содержащей это поле.

Операторы `Exists (Employee, Employee)` и `Exists (Employee)` эквивалентны.

Пример 2.

```
Exists(Employee, 'Bill')
```

Возвращает -1 (True), если значение поля **'Bill'** найдено в текущем содержимом поля **Employee**.

Пример 3

```
Employees: LOAD * inline [ Employee|ID|Salary Bill|001|20000 John|002|30000 Steve|003|35000 ]
(delimiter is '|');  Citizens: Load * inline [ Employee|Address Bill|New York Mary|London
Steve|Chicago Lucy|Madrid Lucy|Paris John|Miami ] (delimiter is '|') where Exists (Employee);
Drop Tables Employees;
```

В результате будет получена таблица, которую можно использовать в визуализации таблицы с помощью измерений Employee и Address.

Предложение where, `where Exists (Employee)`, означает только имена из таблицы Citizens, загруженные в новую таблицу, которые также находятся в таблице Employees. Оператор Drop удаляет таблицу Employees во избежание неопределенности.

Результаты

Employee	Address
Bill	New York
John	Miami
Steve	Chicago

Пример 4

```
Employees: Load * inline [  Employee|ID|Salary Bill|001|20000 John|002|30000 Steve|003|35000 ] (delimiter is '|');   Citizens: Load * inline [ Employee|Address Bill|New York Mary|London Steve|Chicago Lucy|Madrid Lucy|Paris John|Miami ] (delimiter is '|') where not Exists (Employee);   Drop Tables Employees;
```

Предложение where, включая not: `where not Exists (Employee)`.

Это означает, что в новую таблицу загружаются только имена из таблицы Citizens, отсутствующие в таблице Employees.

Обратите внимание, что для Lucy в таблице Citizens имеются два значения, однако в результирующую таблицу включается только одно из них. При загрузке первой строки значение Lucy включается в поле Employee. Таким образом, в ходе проверки второй строки значение уже существует.

Результаты

Employee	Address
Mary	London
Lucy	Madrid

Пример 5

Этот пример демонстрирует порядок загрузки всех значений.

```
Employees: Load Employee As Name; LOAD * inline [ Employee|ID|Salary Bill|001|20000 John|002|30000 Steve|003|35000 ] (delimiter is '|');   Citizens: Load * inline [ Employee|Address Bill|New York Mary|London Steve|Chicago Lucy|Madrid Lucy|Paris John|Miami ] (delimiter is '|') where not Exists (Name, Employee);   Drop Tables Employees;
```

Чтобы получить все значения для Lucy, внесены два изменения:

- В таблицу Employees была вставлена предыдущая загрузка, где Employee переименовано в Name.
  `Load Employee As Name;`
- Условие Where в Citizens изменено на:
  `not Exists (Name, Employee).`

Это создает поля для Name и Employee. Когда проверяется вторая строка с Lucy, она еще не существует в Name.

Результаты

Employee	Address
Mary	London
Lucy	Madrid
Lucy	Paris

FieldIndex

Функция **FieldIndex()**возвращает позицию значения поля **value** в поле **field_name** (в порядке загрузки).

Синтаксис:

```
FieldIndex(field_name , value)
```

Возвращаемые типы данных: целое

Аргументы:

Аргументы

Аргумент	Описание
field_name	Имя поля, для которого требуется индекс. Например, столбец в таблице. Это значение должно быть дано строковым. Это означает, что имя поля должно быть заключено в одинарные кавычки.
value	Значение поля **field_name**.

Ограничения:

Если элемент **value** не может быть найден среди значений поля **field_name**, 0 возвращается.

Примеры и результаты:

В следующих примерах используется поле: **First name** из таблицы **Names**.

Примеры и результаты

Примеры	Результаты
Добавьте образец данных в свое приложение и запустите.	Таблица **Names** загружается как в данных для образца.
Функция диаграммы. В таблице, содержащей измерение First name, добавьте следующую меру:	
`FieldIndex ('First name','John')`	1, поскольку элемент «John» появляется сначала в порядке загрузки поля **First name** Обратите внимание, что в фильтре элемент **John** появится как число 2 сверху, поскольку он отсортирован в алфавитном порядке, а не в порядке загрузки.
`FieldIndex ('First name','Peter')`	4, поскольку элемент **FieldIndex()** возвращает только одно значение, которое встречается сначала в порядке загрузки.
Функция скрипта. При условии, что таблица **Names** загружается как в данных для образца:	

`John1:`
 	
`Load FieldIndex('First name','John') as MyJohnPos`
 	
`Resident Names;`
	`MyJohnPos=1`, поскольку элемент «John» появляется сначала в порядке загрузки поля **First name**. Обратите внимание, что в фильтре элемент **John** появится как число 2 сверху, поскольку он отсортирован в алфавитном порядке, а не в порядке загрузки.

`Peter1:`
 	
`Load FieldIndex('First name','Peter') as MyPeterPos`
 	
`Resident Names;`
	`MyPeterPos=4`, поскольку элемент **FieldIndex()** возвращает только одно значение, которое встречается сначала в порядке загрузки.

Данные, используемые в примере:

```
Names: LOAD * inline [ First name|Last name|Initials|Has cellphone John|Anderson|JA|Yes
Sue|Brown|SB|Yes Mark|Carr|MC|No Peter|Devonshire|PD|No Jane|Elliot|JE|Yes Peter|Franc|PF|Yes
] (delimiter is '|');                    John1: Load FieldIndex('First name','John') as MyJohnPos Resident
Names;   Peter1: Load FieldIndex('First name','Peter') as MyPeterPos Resident Names;
```

FieldValue

Функция **FieldValue()** возвращает значение, находящееся в позиции **elem_no** поля **field_name** (в порядке загрузки).

Синтаксис:

```
FieldValue(field_name , elem_no)
```

Возвращаемые типы данных: двойное значение

Аргументы:

Аргументы

Аргумент	Описание
field_name	Имя поля, для которого требуется значение. Например, столбец в таблице. Это значение должно быть дано строковым. Это означает, что имя поля должно быть заключено в одинарные кавычки.
elem_no	Номер позиции (элемента) поля, следующего в порядке загрузки, для которого возвращено значение. Значение может соответствовать строке в таблице, но это зависит от порядка, в котором загружаются элементы (строки).

Ограничения:

Если элемент **elem_no** больше, чем число значений поля, возвращается значение NULL.

Пример

Скрипт загрузки

Загрузите следующие данные через встроенную загрузку в редакторе загрузки данных, чтобы создать пример, показанный ниже.

```
Names:                                  LOAD * inline [ First name|Last name|Initials|Has cellphone John|And
Sue|Brown|SB|Yes Mark|Carr|MC |No Peter|Devonshire|PD|No Jane|Elliot|JE|Yes Peter|Franc|PF|Yes
] (delimiter is '|');                                   John1:                                                  Load
Names;  Peter1:  Load FieldValue('First name',5) as MyPos2 Resident Names;
```

Создать визуализацию

Создайте визуализацию таблицы на листе Qlik Sense. Добавьте поля **First name**, **MyPos1** и **MyPos2** в таблицу.

Результат

First name	MyPos1	MyPos2
Jane	John	Jane
John	John	Jane
Mark	John	Jane
Peter	John	Jane
Sue	John	Jane

Объяснение

FieldValue('First name','1') возвращает John в качестве значения **MyPos1** для всех имен, так как имя John стоит первым в порядке загрузки поля **First name** (Имя). Обратите внимание, что в фильтре элемент John появится как число 2 сверху после элемента Jane, поскольку он отсортирован в алфавитном порядке, а не в порядке загрузки.

FieldValue('First name','5') возвращает Jane в качестве значения **MyPos2** для всех имен, так как имя Jane стоит пятым в порядке загрузки поля **First name**.

FieldValueCount

Функция **FieldValueCount()** – это функция **целого числа**, которая возвращает уникальные значения в поле.

Частичная перезагрузка может привести к удалению значений из данных, которые не будут отражены в возвращенном числе. Возвращенное число будет включать все уникальные значения, которые были загружены либо при первоначальной перезагрузке или при последующей частичной перезагрузке.

Синтаксис:

```
FieldValueCount(field_name)
```

Возвращаемые типы данных: целое

Аргументы:

Аргументы

Аргумент	Описание
field_name	Имя поля, для которого требуется значение. Например, столбец в таблице. Это значение должно быть дано строковым. Это означает, что имя поля должно быть заключено в одинарные кавычки.

Примеры и результаты:

В следующих примерах используется поле **First name** из таблицы **Names**.

Примеры и результаты

Примеры	Результаты
Добавьте образец данных в свое приложение и запустите.	Таблица **Names** загружается как в данных для образца.
Функция диаграммы. В таблице, содержащей измерение First name, добавьте следующую меру:	
`FieldValueCount('First name')`	Значение 5, поскольку элемент **Peter** появляется дважды.

Примеры	Результаты
`FieldValueCount('Initials')`	Значение 6, поскольку элемент **Initials** имеет только уникальные значения.
Функция скрипта. При условии, что таблица **Names** загружается как в данных для образца:	

`FieldCount1:`
 	
`Load FieldValueCount('First name') as MyFieldCount1`
 	
`Resident Names;`
	`MyFieldCount1`=5, поскольку элемент 'Peter' появляется дважды.

`FieldCount2:`
 	
`Load FieldValueCount('Initials') as MyInitialsCount1`
 	
`Resident Names;`
	`MyFieldCount1`=6, поскольку элемент 'Initials' имеет только уникальные значения.

Данные, используемые в примерах:

```
Names: LOAD * inline [ First name|Last name|Initials|Has cellphone John|Anderson|JA|Yes
Sue|Brown|SB|Yes Mark|Carr|MC|No Peter|Devonshire|PD|No Jane|Elliot|JE|Yes Peter|Franc|PF|Yes
] (delimiter is '|');                    FieldCount1: Load FieldValueCount('First name') as MyFieldCount1
Resident Names;   FieldCount2: Load FieldValueCount('Initials') as MyInitialsCount1 Resident
Names;
```

LookUp

Функция **Lookup()** просматривает загруженную таблицу и возвращает значение поля **field_name**, соответствующее первому вхождению значения **match_field_value** в поле **match_field_name**. Таблица может быть текущей таблицей или другой ранее загруженной таблицей.

Синтаксис:

```
lookup(field_name, match_field_name, match_field_value [, table_name])
```

Возвращаемые типы данных: двойное значение

Аргументы:

Аргументы

Аргумент	Описание
field_name	Имя поля, для которого требуется возвращаемое значение. Вводимое значение необходимо задать в виде строки (например литералы ссылочного типа).
match_ field_name	Имя поля, в котором необходимо искать элемент **match_field_value**. Вводимое значение необходимо задать в виде строки (например литералы ссылочного типа).

Аргумент	Описание
match_ field_value	Значение, которое необходимо искать в поле **match_field_name**.
table_name	Имя таблицы, в которой необходимо искать значение. Вводимое значение необходимо задать в виде строки (например, литералы ссылочного типа). Если элемент **table_name** отсутствует, принимается текущая таблица.

Аргументы без кавычек относятся к текущей таблице. Чтобы отнести аргументы к другой таблице, заключите их в одинарные кавычки.

Ограничения:

Порядком поиска является порядок загрузки, если таблица не является результатом таких сложных операций, как операции объединения, в случае которых порядок недостаточно определен. Поля **field_name** и **match_field_name** должны быть полями в одной таблице, указанной с помощью элемента **table_name**.

Если совпадений не найдено, возвращается значение NULL.

Пример

Скрипт загрузки

Загрузите следующие данные через встроенную загрузку в редакторе загрузки данных, чтобы создать пример, показанный ниже.

```
ProductList: Load * Inline [ ProductID|Product|Category|Price 1|AA|1|1 2|BB|1|3 3|CC|2|8
4|DD|3|2 ] (delimiter is '|');   OrderData: Load *, Lookup('Category', 'ProductID', ProductID,
'ProductList') as CategoryID Inline [ InvoiceID|CustomerID|ProductID|Units 1|Astrida|1|8
1|Astrida|2|6 2|Betacab|3|10 3|Divadip|3|5 4|Divadip|4|10 ] (delimiter is '|');   Drop Table
ProductList;
```

Создать визуализацию

Создайте визуализацию таблицы на листе Qlik Sense. Добавьте в таблицу поля **ProductID**, **InvoiceID**, **CustomerID**, **Units** и **CategoryID**.

Результат

Результирующая таблица

ProductID	InvoiceID	CustomerID	Units	CategoryID
1	1	Astrida	8	1
2	1	Astrida	6	1
3	2	Betacab	10	2

ProductID	InvoiceID	CustomerID	Units	CategoryID
3	3	Divadip	5	2
4	4	Divadip	10	3

Объяснение

Данные образца используют функцию **Lookup()** в следующем виде:

```
Lookup('Category', 'ProductID', ProductID, 'ProductList')
```

Сначала загружается таблица **ProductList**.

Функция **Lookup()** используется для построения таблицы **OrderData**. Она указывает третий аргумент как **ProductID**. Это поле, для которого будет выполняться поиск значения во втором аргументе **'ProductID'** в таблице **ProductList**, как определено завершающими одинарными кавычками.

Функция возвращает значение для '**Category**' (в таблице **ProductList**), загруженной как **CategoryID**.

Оператор **drop** удаляет таблицу **ProductList** из модели данных, поскольку она не требуется. В результате остается конечная таблица **OrderData**.

> *Функция Lookup() гибкая, она может получить доступ к любой ранее загруженной таблице. Тем не менее, она медленно сравнивается с функцией Applymap().*

См. также:

ApplyMap (page 735)

NoOfRows – функция диаграммы

Функция **NoOfRows()** возвращает строки в текущий сегмент столбца в таблице. Для растровых диаграмм функция **NoOfRows()** возвращает строки в эквивалент прямой таблицы диаграммы.

Если таблица или эквивалент таблицы имеют несколько вертикальных измерений, текущий сегмент столбца будет включать только строки с теми же значениями, что и текущая строка во всех столбцах измерений, кроме столбца с последним измерением в межполевом порядке сортировки.

Синтаксис:

```
NoOfRows([TOTAL])
```

Возвращаемые типы данных: целое

Аргументы:

Аргументы

Аргумент	Описание
TOTAL	Если таблица имеет одно измерение, или если в качестве аргумента используется префикс **TOTAL**, текущий сегмент столбца всегда равен всему столбцу.

Пример: Выражение диаграммы с использованием NoOfRows

Пример: выражение диаграммы

Скрипт загрузки

Загрузите следующие данные через встроенную загрузку в редакторе загрузки данных, чтобы создать примеры с выражениями диаграммы, показанные ниже.

```
Temp: LOAD * inline [ Region|SubRegion|RowNo()|NoOfRows() Africa|Eastern Africa|Western Americas|Central Americas|Northern Asia|Eastern Europe|Eastern Europe|Northern Europe|Western Oceania|Australia ] (delimiter is '|');
```

Выражение диаграммы

Создайте на листе Qlik Sense визуализацию таблицы с измерениями **Region** и **SubRegion**. Добавьте меры `RowNo( )`, `NoOfRows()` и `NoOfRows(Total)`.

Результат

Region	SubRegion	RowNo()	NoOfRows()	NoOfRows (Total)
Africa	Eastern	1	2	9
Africa	Western	2	2	9
Americas	Central	1	2	9
Americas	Northern	2	2	9
Asia	Eastern	1	1	9
Europe	Eastern	1	3	9
Europe	Northern	2	3	9
Eurrope	Western	3	3	9
Oceania	Australia	1	1	9

Объяснение

В этом примере сортировка выполняется по первому измерению, Region (Регион). В результате, каждый сегмент столбца состоит из группы регионов с одинаковым значением, например Africa (Африка).

В столбце **RowNo()** отображаются номера строк для каждого сегмента столбца, например имеется две строки для Африки. Нумерация строк для следующего сегмента столбца, который является Americas, начинается в таком случае снова с 1.

Столбец **NoOfRows()** подсчитывает количество строк в каждом сегменте столбца, например для Европы каждый сегмент столбца содержит по три строки.

Столбец **NoOfRows(Total)** игнорирует измерения из-за аргумента `TOTAL` для `NoOfRows()` и подсчитывает строки в таблице.

Если бы таблица была отсортирована по второму измерению, SubRegion (Субрегион), сегменты столбца были бы основаны на этом измерении, поэтому нумерация строк изменилась бы для каждого субрегиона.

См. также:

RowNo – функция диаграммы (page 478)

Peek

Функция **Peek()** возвращает значение поля в таблице для строки, которая уже загружена. Можно указать номер строки или таблицу. Если номер строки не указан, будет использована последняя запись, загруженная ранее.

Функция peek() наиболее часто используется для поиска релевантных границ в загруженной ранее таблице, то есть первое или последнее значение в конкретном поле. В большинстве случаев это значение сохраняется для дальнейшего использования, например в качестве условия в цикле do-while.

Синтаксис:

```
Peek(
field_name
[, row_no[, table_name ] ])
```

Возвращаемые типы данных: двойное значение

Аргументы:

Аргументы

Аргумент	Описание
field_name	Имя поля, для которого требуется возвращаемое значение.Вводимое значение необходимо задать в виде строки (например литералы ссылочного типа).
row_no	Необходима строка в таблице, которая указывает поле. Может быть выражением, но оно должно определяться по целому числу. 0 обозначает первую запись, 1 обозначает вторую и т. д. Отрицательные числа указывают порядок с конца таблицы. -1 обозначает последнюю прочитанную запись. Если элемент **row_no** не задан, используется -1.
table_name	Метка таблицы без двоеточия на конце. Если элемент **table_name** не указан, принимается текущая таблица. При использовании вне оператора **LOAD** или относительно другой таблицы должен быть включен элемент **table_name**.

Ограничения:

Функция может возвращать только значения из уже загруженных записей. Это означает, что в первой записи в таблицы вызов, в котором row_no имеет значение -1, будет возвращать NULL (0).

Примеры и результаты:

Пример 1.

Добавьте образец скрипта в свое приложение и запустите. Чтобы увидеть результаты, добавьте поля, перечисленные в столбце результатов, в лист приложения.

```
EmployeeDates: Load * Inline [ EmployeeCode|StartDate|EndDate 101|02/11/2010|23/06/2012
102|01/11/2011|30/11/2013 103|02/01/2012| 104|02/01/2012|31/03/2012 105|01/04/2012|31/01/2013
106|02/11/2013| ] (delimiter is '|');       First_last_Employee: Load  EmployeeCode,  Peek
('EmployeeCode',0,'EmployeeDates') As FirstCode,  Peek('EmployeeCode',-1,'EmployeeDates') As
LastCode Resident EmployeeDates;
```

Результирующая таблица

Код сотрудника	StartDate	EndDate	FirstCode	LastCode
101	02/11/2010	23/06/2012	101	106
102	01/11/2011	30/11/2013	101	106
103	02/01/2012		101	106
104	02/01/2012	31/03/2012	101	106
105	01/04/2012	31/01/2013	101	106
106	02/11/2013		101	106

FirstCode = 101, поскольку `Peek('EmployeeCode',0, 'EmployeeDates')` возвращает первое значение элемента EmployeeCode в таблице EmployeeDates.

LastCode = 106, поскольку `Peek('EmployeeCode',-1, 'EmployeeDates')` возвращает последнее значение EmployeeCode в таблице EmployeeDates.

Замена значения аргумента **row_no** возвращает значения других строк в таблице следующим образом:

`Peek('EmployeeCode',2, 'EmployeeDates')` возвращает третье значение, 103, в таблице в качестве FirstCode:

Тем не менее, обратите внимание, что без указания таблицы в качестве третьего аргумента **table_name** функция ссылается на текущую (в данном случае внутреннюю) таблицу.

Пример 2.

Если требуется доступ к данным в более низких строчках таблицы, необходимо выполнить два действия: во-первых, загрузить всю таблицу во временную таблицу, а затем выполнить повторную сортировку с помощью **Peek()**.

Добавьте образец скрипта в свое приложение и запустите. Чтобы увидеть результаты, добавьте поля, перечисленные в столбце результатов, в лист приложения.

```
T1: LOAD * inline [ ID|Value 1|3 1|4 1|6 3|7 3|8 2|1 2|11 5|2 5|78 5|13 ] (delimiter is '|');
T2:  LOAD *, IF(ID=Peek('ID'), Peek('List')&','&Value,Value) AS List RESIDENT T1 ORDER BY ID ASC; DROP TABLE T1;
```

Create a table in a sheet in your app with **ID**, **List**, and **Value** as the dimensions.

Результирующая таблица

ID	Список	Значение
1	3,4	4
1	3,4,6	6
1	3	3
2	1,11	11
2	1	1
3	7,8	8
3	7	7
5	2,78	78
5	2,78,13	13
5	2	2

Оператор **IF()** строится на основе временной таблицы T1.
`Peek('ID')` ссылается на поле ID в предыдущей строке в текущей таблице T2.
`Peek('List')` ссылается на поле List в предыдущей строке в таблице T2, которая строится в настоящее время как оценивающееся выражение.

Оператор оценивается следующим образом:
если текущее значение элемента ID такое же, как предыдущее значение элемента ID, то значение элемента Peek('List') записывается как объединенное с текущим значением элемента Value. В противном случае записывается только текущее значение элемента Value.

Если функция Peek('List') уже содержит объединенный результат, новый результат элемента Peek ('List') будет объединен с ним.

*Обратите внимание на предложение **Order by**. Оно указывает порядок организации таблицы (по ID по возрастанию). Без этого функция Peek() будет использовать тот обязательный порядок, который указан во внутренней таблице, что может привести к непредсказуемым результатам.*

Пример 3

Добавьте образец скрипта в свое приложение и запустите. Чтобы увидеть результаты, добавьте поля, перечисленные в столбце результатов, в лист приложения.

```
Amounts: Load  Date#(Month,'YYYY-MM') as Month, Amount, Peek(Amount) as AmountMonthBefore
Inline [Month,Amount 2022-01,2 2022-02,3 2022-03,7 2022-04,9 2022-05,4 2022-06,1];
```

Результирующая таблица

Amount	AmountMonthBefore	Месяц
1	4	2022-06
2	-	2022-01
3	2	2022-02
4	9	2022-05
7	3	2022-03
9	7	2022-04

Поле AmountMonthBefore не будет содержать сумму за предыдущий месяц.

Здесь параметры row_no и table_name отброшены, поэтому используются значения по умолчанию. В этом примере следующие три вызова функции являются эквивалентными:

- Peek(Amount)
- Peek(Amount,-1)
- Peek(Amount,-1,'Amounts')

Значение -1 для параметра row_no указывает, что будет использоваться значение из предыдущей строки. В результате замены этого значения можно извлечь значения других строк таблицы:

Peek(Amount,2) возвращает третье значение в таблице: 7.

Пример 4

Данные необходимо правильно сортировать для получения правильных результатов, но, к сожалению, это не всегда выполняется. Функцию Peek() нельзя использовать для указания ссылки на данные, которые еще не загружены. Такие проблемы можно предотвратить, используя временные таблицы и выполняя несколько обходов данных.

Добавьте образец скрипта в свое приложение и запустите. Чтобы увидеть результаты, добавьте поля, перечисленные в столбце результатов, в лист приложения.

```
tmp1Amounts: Load * Inline [Month,Product,Amount 2022-01,B,3 2022-01,A,8 2022-02,B,4 2022-02,A,6 2022-03,B,1 2022-03,A,6 2022-04,A,5 2022-04,B,5 2022-05,B,6 2022-05,A,7 2022-06,A,4 2022-06,B,8];  tmp2Amounts: Load *, If(Product=Peek(Product),Peek(Amount)) as AmountMonthBefore Resident tmp1Amounts Order By Product, Month Asc;  Drop Table tmp1Amounts; Amounts: Load *, If(Product=Peek(Product),Peek(Amount)) as AmountMonthAfter Resident tmp2Amounts Order By Product, Month Desc;  Drop Table tmp2Amounts;
```

Объяснение

Первоначальная таблица сортируется по месяцу, таким образом функция peek() во многих случаях будет возвращать сумму для неправильного продукта. Таким образом, данную таблицу потребуется сортировать повторно. Это осуществляется путем выполнения второго обхода данных, в рамках которого создается новая таблица tmp2Amounts. Обратите внимание на предложение Order by. Оно упорядочивает записи сначала по продукту, затем по месяцу в восходящем порядке.

Необходимо использовать функцию If(), так как AmountMonthBefore следует рассчитывать, только если предыдущая строка содержит данные для того же продукта, но за предыдущий месяц. Это условие можно проверить путем сравнения продукта в текущей строке с продуктом в предыдущей строке.

Когда создается вторая таблица, первая таблица tmp1Amounts отбрасывается с использованием оператора Drop Table.

В заключение, выполняется третий проход по данным, но теперь с сортировкой месяцев в обратном порядке. Таким образом также можно рассчитать AmountMonthAfter.

Предложения Order by указывают, как упорядочивается таблица. Без этого функция Peek() будет использовать любой произвольный порядок, который указан во внутренней таблице, что может привести к непредсказуемым результатам.

Результат

Результирующая таблица

Месяц	Продукт	Amount	AmountMonthBefore	AmountMonthAfter
2022-01	A	8	-	6
2022-02	B	3	-	4
2022-03	A	6	8	6
2022-04	B	4	3	1
2022-05	A	6	6	5
2022-06	B	1	4	5
2022-01	A	5	6	7
2022-02	B	5	1	6
2022-03	A	7	5	4
2022-04	B	6	5	8
2022-05	A	4	7	-
2022-06	B	8	6	-

Пример 5

Добавьте образец скрипта в свое приложение и запустите. Чтобы увидеть результаты, добавьте поля, перечисленные в столбце результатов, в лист приложения.

```
T1: Load * inline [ Quarter, Value 2003q1, 10000 2003q1, 25000 2003q1, 30000 2003q2, 1250 2003q2, 55000 2003q2, 76200 2003q3, 9240 2003q3, 33150 2003q3, 89450 2003q4, 1000 2003q4, 3000 2003q4, 5000 2004q1, 1000 2004q1, 1250 2004q1, 3000 2004q2, 5000 2004q2, 9240 2004q2, 10000 2004q3, 25000 2004q3, 30000 2004q3, 33150 2004q4, 55000 2004q4, 76200 2004q4, 89450 ];  T2: Load *, rangesum(SumVal,peek('AccSumVal')) as AccSumVal; Load Quarter, sum(Value) as SumVal resident T1 group by Quarter;
```

Результат

Результирующая таблица

Квартал	SumVal	AccSumVal
2003q1	65000	65000
2003q2	132450	197450
2003q3	131840	329290
2003q4	9000	338290
2004q1	5250	343540

Квартал	SumVal	AccSumVal
2004q2	24240	367780
2004q3	88150	455930
2004q4	220650	676580

Объяснение

Оператор загрузки **Load *, rangesum(SumVal,peek('AccSumVal')) as AccSumVal** включает рекурсивный вызов, где предыдущие значения добавляются к текущему значению. Эта операция служит для расчета аккумуляции значений в скрипте.

См. также:

Previous

Функция **Previous()** находит значение выражения **expr** с помощью данных из ранее введенной записи, которая не была сброшена из-за предложения **where**. В первой записи внутренней таблицы функция возвратит значение NULL.

Синтаксис:

```
Previous(expr)
```

Возвращаемые типы данных: двойное значение

Аргументы:

Аргументы

Аргумент	Описание
expr	Выражение или поле, содержащее данные для измерения. Выражение может содержать вложенные функции **previous()**, чтобы получить доступ к более ранним записям. Данные выбираются из входного источника напрямую, что также позволяет ссылаться на поля, которые не были загружены в программу Qlik Sense, то есть даже если они не были сохранены в ассоциативной базе данных.

Ограничения:

В первой записи внутренней таблицы функция возвращает значение NULL.

Пример:

Введите следующее в скрипт загрузки

```
Sales2013:
Load *, (Sales - Previous(Sales) )as Increase Inline [
```

```
Month|Sales
1|12
2|13
3|15
4|17
5|21
6|21
7|22
8|23
9|32
10|35
11|40
12|41
] (delimiter is '|');
```

При использовании функции **Previous()** в операторе **Load** можно сравнить текущее значение элемента Sales с предшествующим значением и использовать его в третьем поле Increase.

Результирующая таблица

Месяц	Sales	Increase
1	12	-
2	13	1
3	15	2
4	17	2
5	21	4
6	21	0
7	22	1
8	23	1
9	32	9
10	35	3
11	40	5
12	41	1

Top — функция диаграммы

Функция **Top()** оценивает выражение в первой (верхней) строке сегмента столбца в таблице. Строка, для которой выполняется вычисление, зависит от значения элемента **offset**, если таковой имеет место, по умолчанию принимается верхняя строка. Для диаграмм, за исключением таблиц, функция **Top()** используется для оценки в первой строке текущего столбца в эквиваленте прямой таблицы диаграммы.

Синтаксис:

```
Top([TOTAL] expr [ , offset [,count ]])
```

Возвращаемые типы данных: двойное значение

Аргументы:

Аргументы

Аргумент	Описание
expr	Выражение или поле, содержащее данные для измерения.
offset	Если задать значение **offset** элемента n больше 1, можно будет переместить оценку выражения n по строкам ниже верхней строки. Если задать отрицательное число смещения, функция **Top** будет работать как функция **Bottom** с соответствующим положительным числом смещения.
count	Если задать для третьего параметра **count** значение больше 1, функция вернет ряд значений элемента **count**: по одному для каждой последней строки элемента **count** текущего сегмента столбца. В данной форме функция может использоваться в качестве аргумента для любой специальной функции интервала. *Функции над выборкой (page 743)*
TOTAL	Если таблица имеет одно измерение, или если в качестве аргумента используется префикс **TOTAL**, текущий сегмент столбца всегда равен всему столбцу.

Сегмент столбца определяется как последовательное подмножество ячеек с теми же значениями для измерений в текущем порядке сортировки. Межзаписные функции диаграмм выполняют вычисления в сегменте столбца за исключением крайнего правого измерения в эквивалентной прямой таблице. Если в диаграмме есть только одно измерение, или если указан квалификатор TOTAL, выражение оценивается по всей таблице.

Если таблица или эквивалент таблицы имеют несколько вертикальных измерений, текущий сегмент столбца будет включать только строки с теми же значениями, что и текущая строка во всех столбцах измерений, кроме столбца с последним измерением в межполевом порядке сортировки.

Ограничения:

Рекурсивные вызовы возвращают значение NULL.

Примеры и результаты:

Пример: 1

На снимке таблицы, показанной в этом примере, визуализация таблицы создана из измерения **Customer** и мер: `Sum(Sales)` и `Top(Sum(Sales))`.

Столбец **Top(Sum(Sales))** возвращает значение 587 для всех строк, поскольку это значение верхней строки: **Astrida**.

В таблице также показаны более сложные меры: одна, созданная из элемента `Sum(Sales)+Top(Sum(Sales))`, а другая, помеченная как **Top offset 3**, созданная с помощью выражения `Sum(Sales)+Top(Sum(Sales), 3)`, и имеющая аргумент **offset**, установленный на 3. Таким образом добавляется значение **Sum(Sales)** для текущей строки к значению из третьей строки от верхней строки, т. е. текущая строка плюс значение для элемента**Canutility**.

Пример 1.

Top and Bottom

Customer	Sum(Sales)	Top(Sum(Sales))	Sum(Sales)+Top(Sum(Sales))	Top offset 3
Totals	2566	587	3153	3249
Astrida	587	587	1174	1270
Betacab	539	587	1126	1222
Canutility	683	587	1270	1366
Divadip	757	587	1344	1440

Пример: 2

На снимках таблиц, показанных в этом примере, к визуализациям добавлено больше измерений: **Month** и **Product**. Для диаграмм с несколькими измерениями результаты выражений, содержащих функции **Above**, **Below**, **Top** и **Bottom**, зависят от порядка, в котором измерения столбцов сортируются Qlik Sense. Программа Qlik Sense оценивает функции на основе сегментов столбца, полученных из измерения, отсортированного последним. Контроль за порядком сортировки столбцов осуществляется на панели свойств под элементом **Сортировка**. Этот порядок не обязательно соответствует порядку отображения столбцов в таблице.

Первая таблица для примера 2. Значение элемента Top для меры First value основано на элементе Month (Jan).

Customer	Product	Month	Sum(Sales)	First value
			2566	-
Astrida	AA	Jan	46	46
Astrida	AA	Feb	60	46
Astrida	AA	Mar	70	46
Astrida	AA	Apr	13	46
Astrida	AA	May	78	46
Astrida	AA	Jun	20	46
Astrida	AA	Jul	45	46
Astrida	AA	Aug	65	46
Astrida	AA	Sep	78	46
Astrida	AA	Oct	12	46
Astrida	AA	Nov	78	46
Astrida	AA	Dec	22	46

Вторая таблица для примера 2. Значение элемента Top для меры First value основано на элементе Product (AA для Astrida).

Customer	Product	Month	Sum(Sales)	First value
			2566	-
Astrida	AA	Jan	46	46
Astrida	BB	Jan	46	46
Astrida	AA	Feb	60	60
Astrida	BB	Feb	60	60
Astrida	AA	Mar	70	70
Astrida	BB	Mar	70	70
Astrida	AA	Apr	13	13
Astrida	BB	Apr	13	13

Дополнительную информацию см. в примере 2 для функции **Above**.

Пример 3

Пример: 3	Результат
Функцию **Top** можно использовать как ввод в функции над выборкой. Например, элемент `RangeAvg (Top(Sum(Sales),1,3))`.	В аргументах для функции **Top()** для элемента offset задано значение 1, а для элемента count задано значение 3. Функция находит результаты выражения **Sum(Sales)** в трех строках, начиная со строки под нижней строкой в сегменте столбца (поскольку offset=1), и в двух строках под ней (если есть строка). Эти три значения используются как ввод в функцию RangeAvg(), которая находит среднее значение в предоставленном диапазоне чисел. Таблица с элементом **Customer** в виде измерения выдает следующие результаты для выражения RangeAvg(). Astrida 603 Betacab 603 Canutility 603 Divadip: 603

```
Monthnames:
LOAD *, Dual(MonthText,MonthNumber) as Month INLINE [
MonthText, MonthNumber
Jan, 1
Feb, 2
Mar, 3
Apr, 4
May, 5
Jun, 6
Jul, 7
Aug, 8
Sep, 9
Oct, 10
Nov, 11
Dec, 12
];

Sales2013:
Crosstable (MonthText, Sales) LOAD * inline [
Customer|Jan|Feb|Mar|Apr|May|Jun|Jul|Aug|Sep|Oct|Nov|Dec
Astrida|46|60|70|13|78|20|45|65|78|12|78|22
Betacab|65|56|22|79|12|56|45|24|32|78|55|15
Canutility|77|68|34|91|24|68|57|36|44|90|67|27
Divadip|57|36|44|90|67|27|57|68|47|90|80|94
] (delimiter is '|');
```

См. также:

- *Bottom – функция диаграммы (page 698)*
- *Above – функция диаграммы (page 689)*
- *Sum – функция диаграммы (page 248)*
- *RangeAvg (page 746)*
- *Функции над выборкой (page 743)*

SecondaryDimensionality – функция диаграммы

Функция **SecondaryDimensionality()** возвращает количество строк измерений сводной таблицы, имеющих неагрегированное содержимое, т. е. не содержащих частичных сумм или свернутых агрегированных показателей. Данная функция является эквивалентом функции **dimensionality()** для горизонтальных измерений сводной таблицы.

Синтаксис:

```
SecondaryDimensionality( )
```

Возвращаемые типы данных: целое

Ограничения:

Функция **SecondaryDimensionality** всегда возвращает 0, за исключением случаев использования в сводных таблицах.

After – функция диаграммы

Функция **After()** возвращает значение выражения, оцененного со значениями измерения сводной таблицы по мере их отображения в столбце после текущего столбца в сегменте строки сводной таблицы.

Синтаксис:

```
after([TOTAL] expr [, offset [, count ]])
```

Данная функция возвращает значение NULL во всех типах диаграмм, кроме сводных таблиц.

Аргументы:

Аргумент	Описание
expr	Выражение или поле, содержащее данные для измерения.

Аргумент	Описание
offset	Если задать значение **offset** n больше 1, можно переместить оценку выражения на n строк вправо от текущей строки. Если задать смещение равным 0, оценка выражения будет выполнена в текущей строке. Если задать отрицательное число смещения, функция **After** будет работать как функция **Before** с соответствующим положительным числом смещения.
count	Если задать для третьего параметра **count** значение больше 1, функция вернет диапазон значений элемента **count**, по одному для каждой строки таблицы элемента, считая вправо от исходной ячейки.
TOTAL	Если таблица имеет одно измерение, или если в качестве аргумента используется префикс **TOTAL**, текущий сегмент столбца всегда равен всему столбцу.

В последнем столбце сегмента строки будет возвращено значение NULL, так как после этого столбца нет других столбцов.

Если сводная таблица имеет несколько горизонтальных измерений, текущий сегмент строки будет включать только столбцы с теми же значениями, что и текущий столбец во всех строках с измерениями, кроме строки с последним горизонтальным измерением в межполевом порядке сортировки. Межполевой порядок сортировки для горизонтальных измерений в сводных таблицах определяется просто по порядку измерений сверху вниз.

Пример:

```
after( sum( Sales ))
after( sum( Sales ), 2 )
after( total sum( Sales ))
```

`rangeavg (after(sum(x),1,3))` возвращает средний из трех результатов функции **sum(x)**, оцененной в трех столбцах непосредственно справа от текущего столбца.

Before – функция диаграммы

Функция **Before()** возвращает значение выражения, оцененного со значениями измерения сводной таблицы по мере их отображения в столбце перед текущим столбцом в сегменте строки сводной таблицы.

Синтаксис:

```
before([TOTAL] expr [, offset [, count]])
```

Данная функция возвращает значение NULL во всех типах диаграмм, кроме сводных таблиц.

Аргументы:

Аргументы

Аргумент	Описание
expr	Выражение или поле, содержащее данные для измерения.
offset	Если задать значение **offset** n больше 1, можно переместить оценку выражения на n строк влево от текущей строки. Если задать смещение равным 0, оценка выражения будет выполнена в текущей строке. Если задать отрицательное число смещения, функция **Before** будет работать как функция **After** с соответствующим положительным числом смещения.
count	Если задать для третьего параметра **count** значение больше 1, функция вернет диапазон значений элемента **count**, по одному для каждой строки таблицы элемента, считая влево от исходной ячейки.
TOTAL	Если таблица имеет одно измерение, или если в качестве аргумента используется префикс **TOTAL**, текущий сегмент столбца всегда равен всему столбцу.

В первом столбце сегмента строки будет возвращено значение NULL, так как перед этим столбцом нет других столбцов.

Если сводная таблица имеет несколько горизонтальных измерений, текущий сегмент строки будет включать только столбцы с теми же значениями, что и текущий столбец во всех строках с измерениями, кроме строки с последним горизонтальным измерением в межполевом порядке сортировки.Межполевой порядок сортировки для горизонтальных измерений в сводных таблицах определяется просто по порядку измерений сверху вниз.

Примеры:

```
before( sum( Sales ))
before( sum( Sales ), 2 )
before( total sum( Sales ))
```

`rangeavg (before(sum(x),1,3))` возвращает средний из трех результатов функции **sum(x)**, оцененной в трех столбцах непосредственно слева от текущего столбца.

First – функция диаграммы

Функция **First()** возвращает значение выражения, оцененного со значениями измерения сводной таблицы по мере их отображения в первом столбце текущего сегмента строки сводной таблицы. Данная функция возвращает значение NULL во всех типах диаграмм, кроме сводных таблиц.

Синтаксис:

```
first([TOTAL] expr [, offset [, count]])
```

Аргументы:

Аргументы

Аргумент	Описание
expression	Выражение или поле, содержащее данные для измерения.
offset	Если задать значение **offset** n больше 1, можно переместить оценку выражения на n строк вправо от текущей строки. Если задать смещение равным 0, оценка выражения будет выполнена в текущей строке. Если задать отрицательное число смещения, функция **First** будет работать как функция **Last** с соответствующим положительным числом смещения.
count	Если задать для третьего параметра **count** значение больше 1, функция вернет диапазон значений элемента **count**, по одному для каждой строки таблицы элемента, считая вправо от исходной ячейки.
TOTAL	Если таблица имеет одно измерение, или если в качестве аргумента используется префикс **TOTAL**, текущий сегмент столбца всегда равен всему столбцу.

Если сводная таблица имеет несколько горизонтальных измерений, текущий сегмент строки будет включать только столбцы с теми же значениями, что и текущий столбец во всех строках с измерениями, кроме строки с последним горизонтальным измерением в межполевом порядке сортировки.Межполевой порядок сортировки для горизонтальных измерений в сводных таблицах определяется просто по порядку измерений сверху вниз.

Примеры:

```
first( sum( Sales ))
first( sum( Sales ), 2 )
first( total sum( Sales )
```

`rangeavg (first(sum(x),1,5))` возвращает средний из результатов функции **sum(x)**, оцененной в пяти самых левых столбцах текущего сегмента строки.

Last – функция диаграммы

Функция **Last()** возвращает значение выражения, оцененного со значениями измерения сводной таблицы по мере их отображения в последнем столбце текущего сегмента строки сводной таблицы. Данная функция возвращает значение NULL во всех типах диаграмм, кроме сводных таблиц.

Синтаксис:

```
last([TOTAL] expr [, offset [, count]])
```

Аргументы:

Аргументы

Аргумент	Описание
expr	Выражение или поле, содержащее данные для измерения.
offset	Если задать значение **offset** n больше 1, можно переместить оценку выражения на n строк влево от текущей строки. Если задать смещение равным 0, оценка выражения будет выполнена в текущей строке. Если задать отрицательное число смещения, функция **First** будет работать как функция **Last** с соответствующим положительным числом смещения.
count	Если задать для третьего параметра **count** значение больше 1, функция вернет диапазон значений элемента **count**, по одному для каждой строки таблицы элемента, считая влево от исходной ячейки.
TOTAL	Если таблица имеет одно измерение, или если в качестве аргумента используется префикс **TOTAL**, текущий сегмент столбца всегда равен всему столбцу.

Если сводная таблица имеет несколько горизонтальных измерений, текущий сегмент строки будет включать только столбцы с теми же значениями, что и текущий столбец во всех строках с измерениями, кроме строки с последним горизонтальным измерением в межполевом порядке сортировки.Межполевой порядок сортировки для горизонтальных измерений в сводных таблицах определяется просто по порядку измерений сверху вниз.

Пример:

```
last( sum( Sales ))
last( sum( Sales ), 2 )
last( total sum( Sales )
```

`rangeavg (last(sum(x),1,5))` возвращает средний из результатов функции **sum(x)**, оцененной в пяти самых правых столбцах текущего сегмента строки.

ColumnNo – функция диаграммы

Функция **ColumnNo()** возвращает количество текущих столбцов в текущем сегменте строки сводной таблицы. Первый столбец имеет номер 1.

Синтаксис:

```
ColumnNo([total])
```

Аргументы:

Аргументы

Аргумент	Описание
TOTAL	Если таблица имеет одно измерение, или если в качестве аргумента используется префикс **TOTAL**, текущий сегмент столбца всегда равен всему столбцу.

Если сводная таблица имеет несколько горизонтальных измерений, текущий сегмент строки будет включать только столбцы с теми же значениями, что и текущий столбец во всех строках с измерениями, кроме строки с последним горизонтальным измерением в межполевом порядке сортировки.Межполевой порядок сортировки для горизонтальных измерений в сводных таблицах определяется просто по порядку измерений сверху вниз.

Пример:

```
if( ColumnNo( )=1, 0, sum( Sales ) / before( sum( Sales )))
```

NoOfColumns – функция диаграммы

Функция **NoOfColumns()** возвращает количество столбцов в текущем сегменте строки сводной таблицы.

Синтаксис:

```
NoOfColumns([total])
```

Аргументы:

Аргументы

Аргумент	Описание
TOTAL	Если таблица имеет одно измерение, или если в качестве аргумента используется префикс **TOTAL**, текущий сегмент столбца всегда равен всему столбцу.

Если сводная таблица имеет несколько горизонтальных измерений, текущий сегмент строки будет включать только столбцы с теми же значениями, что и текущий столбец во всех строках с измерениями, кроме строки с последним измерением в межполевом порядке сортировки.Межполевой порядок сортировки для горизонтальных измерений в сводных таблицах определяется просто по порядку измерений сверху вниз.

Пример:

```
if( ColumnNo( )=NoOfColumns( ), 0, after( sum( Sales )))
```

5.17 Логические функции

В этом разделе описаны функции, относящиеся к логическим операциям. Все функции можно использовать как в скрипте загрузки данных, так и в выражениях диаграмм.

IsNum
Возвращает -1 (True), если выражение можно интерпретировать как число, в противном случае возвращает 0 (False).

```
IsNum( expr )
```

IsText
Возвращает -1 (True), если выражение предусматривает представление текста, в противном случае возвращает 0 (False).

```
IsText( expr )
```

***IsNum** и **IsText** возвращают 0, если выражение равно NULL.*

Пример:

В следующем примере загружается встроенная таблица с текстовыми и числовыми значениями и добавляются два поля для проверки, является ли значение числовым или текстовым.

```
Load *, IsNum(Value), IsText(Value)
Inline [
Value
23
Green
Blue
12
33Red];
```

Полученная таблица выглядит следующим образом:

Resulting table

Value	IsNum(Value)	IsText(Value)
23	-1	0
Green	0	-1
Blue	0	-1
12	-1	0
33Red	0	-1

5.18 Функции сопоставления

В этом разделе описаны функции, относящиеся к таблицам сопоставления. Таблица сопоставления может быть использована для замены значений полей или имен полей во время выполнения скрипта.

Функции сопоставления можно использовать только в скрипте загрузки данных.

Обзор функций сопоставления

Каждая функция подробно описана после обзора. Также можно щелкнуть имя функции в синтаксисе, чтобы получить немедленный доступ к подробной информации об этой конкретной функции.

ApplyMap

Функция скрипта **ApplyMap** используется для сопоставления результата выражения с ранее загруженной таблицей сопоставления.

```
ApplyMap ('mapname', expr [ , defaultexpr ] )
```

MapSubstring

Функция скрипта **MapSubstring** используется для сопоставления частей выражения с загруженной таблицей сопоставления. Сопоставление выполняется с учетом регистра и не является итеративным, причем подстроки сопоставляются слева направо.

```
MapSubstring ('mapname', expr)
```

ApplyMap

Функция скрипта **ApplyMap** используется для сопоставления результата выражения с ранее загруженной таблицей сопоставления.

Синтаксис:

```
ApplyMap('map_name', expression [ , default_mapping ] )
```

Возвращаемые типы данных: двойное значение

Аргументы:

Аргументы

Аргумент	Описание
map_name	Имя таблицы сопоставления, созданной ранее с помощью операторов **mapping load** или **mapping select**. Имя таблицы должно быть заключено в одинарные прямые кавычки. *Если функция используется в развернутой переменной макроса со ссылкой на несуществующую таблицу сопоставления, происходит сбой вызова функции и поле не создается.*
expression	Выражение, результат которого должен быть сопоставлен.
default_ mapping	Если это значение задано, оно будет использовано как значение по умолчанию, если таблица сопоставления не содержит совпадающего значения для параметра expression. Если значение не задано, то значение параметра expression выводится как есть.

Название поля вывода ApplyMap не должно совпадать с названием поля ввода. Это может привести к непредвиденным результатам. Пример недопустимого использования: `ApplyMap('Map', A) as A`.

Пример:

В этом примере мы загружаем список продавцов с кодом страны, представляющим их страну проживания. Мы используем таблицу, соответствующую коду страны, для той страны, код которой будет заменен ее названием. В таблице сопоставления указаны только три страны, коды других стран указаны в параметре 'Rest of the world'.

```
// Load mapping table of country codes:
map1:
mapping LOAD *
Inline [
CCode, Country
Sw, Sweden
Dk, Denmark
No, Norway
] ;

// Load list of salesmen, mapping country code to country
// If the country code is not in the mapping table, put Rest of the world
Salespersons:
LOAD *,
```

```
ApplyMap('map1', CCode,'Rest of the world') As Country
Inline [
CCode, Salesperson
Sw, John
Sw, Mary
Sw, Per
Dk, Preben
Dk, Olle
No, Ole
Sf, Risttu
] ;

// We don't need the CCode anymore
Drop Field 'CCode';
```

Полученная таблица («Менеджеры по продажам») выглядит следующим образом:

Resulting table

Salesperson	Country
John	Sweden
Mary	Sweden
Per	Sweden
Preben	Denmark
Olle	Denmark
Ole	Norway
Risttu	Rest of the world

MapSubstring

Функция скрипта **MapSubstring** используется для сопоставления частей выражения с загруженной таблицей сопоставления. Сопоставление выполняется с учетом регистра и не является итеративным, причем подстроки сопоставляются слева направо.

Синтаксис:

```
MapSubstring('map_name', expression)
```

Возвращаемые типы данных: строка

Аргументы:

Аргументы

Аргумент	Описание
map_name	Имя таблицы сопоставления, считанной ранее оператором **mapping load** или **mapping select**. Имя должно быть заключено в одинарные прямые кавычки. *Если функция используется в развернутой переменной макроса со ссылкой на несуществующую таблицу сопоставления, происходит сбой вызова функции и поле не создается.*
expression	Выражение, результат которого должен быть сопоставлен по подстрокам.

Пример:

В этом примере мы загружаем список моделей продукта. Каждая модель имеет набор атрибутов, которые описываются составным кодом. С помощью таблицы сопоставления с MapSubstring мы можем расширить коды атрибутов до описания.

```
map2:
mapping LOAD *
Inline [
AttCode, Attribute
R, Red
Y, Yellow
B, Blue
C, Cotton
P, Polyester
S, Small
M, Medium
L, Large
] ;

Productmodels:
LOAD *,
MapSubString('map2', AttCode) as Description
Inline [
Model, AttCode
Twixie, R C S
Boomer, B P L
Raven, Y P M
Seedling, R C L
SeedlingPlus, R C L with hood
Younger, B C with patch
MultiStripe, R Y B C S/M/L
] ;
```

```
// We don't need the AttCode anymore
Drop Field 'AttCode';
```

Полученная таблица выглядит следующим образом:

Resulting table

Model	Description
Twixie	Red Cotton Small
Boomer	Blue Polyester Large
Raven	Yellow Polyester Medium
Seedling	Red Cotton Large
SeedlingPlus	Red Cotton Large with hood
Younger	Blue Cotton with patch
MultiStripe	Red Yellow Blue Cotton Small/Medium/Large

5.19 Математические функции

В этом разделе описаны функции для математических констант и булевых значений. Эти функции не имеют никаких параметров, но завершающие скобки тем не менее требуются.

Все функции можно использовать как в скрипте загрузки данных, так и в выражениях диаграмм.

e

Функция возвращает основание натуральных логарифмов **e** (2,71828...).

```
e( )
```

false

Функция возвращает двойное значение, включающее текстовое значение 'False' и числовое значение 0, которое может использоваться в выражении как логическое значение false.

```
false( )
```

pi

Функция возвращает значение π (3,14159...)

```
pi( )
```

rand

Функция возвращает случайное число в пределах от 0 до 1. Это можно использовать для создания данных образца.

```
rand( )
```

Пример:

В этом примере скрипт создает таблицу из 1000 записей с произвольно выбранными символами в верхнем регистре, т. е. символами в диапазоне от 65 до 91 (65+26).

```
Load
     Chr( Floor(rand() * 26) + 65) as UCaseChar,
     RecNo() as ID
     Autogenerate 1000;
```

true

Функция возвращает двойное значение, включающее текстовое значение 'True' и числовое значение -1, которое может использоваться в выражении как логическое значение true.

```
true( )
```

5.20 Функции NULL

В этом разделе описаны функции для возврата или обнаружения значений NULL.

Все функции можно использовать как в скрипте загрузки данных, так и в выражениях диаграмм.

Обзор функций NULL

Каждая функция подробно описана после обзора. Также можно щелкнуть имя функции в синтаксисе, чтобы получить немедленный доступ к подробной информации об этой конкретной функции.

EmptyIsNull

Функция **EmptyIsNull** преобразует пустые строки в NULL. Следовательно, если параметр – пустая строка, она возвращает NULL, иначе она возвращает параметр.

```
EmptyIsNull (expr )
```

IsNull

Функция **IsNull** проверяет, является ли значение выражения NULL. Если да, то функция возвращает -1 (True), в противном случае – 0 (False).

```
IsNull (expr )
```

Null

Функция **Null** возвращает значение NULL.

```
NULL( )
```

EmptyIsNull

Функция **EmptyIsNull** преобразует пустые строки в NULL. Следовательно, если параметр – пустая строка, она возвращает NULL, иначе она возвращает параметр.

Синтаксис:

```
EmptyIsNull(exp )
```

Примеры и результаты:

Примеры написания скриптов

Пример	Результат
`EmptyIsNull(AdditionalComments)`	Это выражение вернет как нуль любые значения пустой строки поля *AdditionalComments* вместо пустых строк. Возвращаются непустые строки и числа.
`EmptyIsNull(PurgeChar (PhoneNumber, ' -()'))`	Это выражение удалит любые тире, пробелы и круглые скобки из поля *PhoneNumber*. Если других символов не осталось, функция EmptyIsNull возвращает пустую строку как нуль; пустой номер телефона совпадает с отсутствующим номером телефона.

IsNull

Функция **IsNull** проверяет, является ли значение выражения NULL. Если да, то функция возвращает -1 (True), в противном случае – 0 (False).

Синтаксис:

```
IsNull(expr )
```

*Строка с нулевой длиной не считается значением NULL и приведет к возврату значения False функцией **IsNull**.*

Пример: Скрипт загрузки данных

В этом примере загружена встроенная таблица с четырьмя строками, где первые три строки не содержат ничего, - или содержат значение 'NULL' в столбце Value. Мы преобразуем эти значения в представления значения true NULL с предшествующим в середине оператором **LOAD** с помощью функции **Null**.

Предшествующий в начале оператор **LOAD** добавляет поле для проверки, является ли значение NULL, с помощью функции **IsNull**.

```
NullsDetectedAndConverted:

LOAD *,
If(IsNull(ValueNullConv), 'T', 'F') as IsItNull;

LOAD *,
If(len(trim(Value))= 0 or Value='NULL' or Value='-', Null(), Value ) as ValueNullConv;

LOAD * Inline
[ID, Value
```

```
0,
1,NULL
2,-
3,Value];
```

Это результирующая таблица. В столбце ValueNullConv значения NULL представлены элементом -.

Resulting table

ID	Value	ValueNullConv	IsItNull
0		-	T
1	NULL	-	T
2	-	-	T
3	Value	Value	F

NULL

Функция **Null** возвращает значение NULL.

Синтаксис:

```
Null( )
```

Пример: Скрипт загрузки данных

В этом примере загружена встроенная таблица с четырьмя строками, где первые три строки не содержат ничего, - или содержат значение 'NULL' в столбце Value. Мы хотим преобразовать эти значения в представления значений true NULL.

Предшествующий в середине оператор **LOAD** выполняет преобразование с помощью функции **Null**.

Предшествующий в начале оператор **LOAD** добавляет поле, чтобы проверить, представлено ли значение NULL в данном примере только в целях иллюстрации.

```
NullsDetectedAndConverted:

LOAD *,
If(IsNull(ValueNullConv), 'T', 'F') as IsItNull;

LOAD *,
If(len(trim(Value))= 0 or Value='NULL' or Value='-', Null(), Value ) as ValueNullConv;

LOAD * Inline
[ID, Value
0,
1,NULL
2,-
3,Value];
```

Это результирующая таблица. В столбце ValueNullConv значения NULL представлены элементом -.

Resulting table

ID	Value	ValueNullConv	IsItNull
0		-	T
1	NULL	-	T
2	-	-	T
3	Value	Value	F

5.21 Функции над выборкой

Функции над выборкой – это функции, которые принимают диапазон значений и выдают в результате одиночное значение. Все функции над выборкой можно использовать как в скрипте загрузки данных, так и в выражениях диаграмм.

Например в визуализации функция над выборкой может вычислить одиночное значение из диапазона между записями. В скрипте загрузки данных функция над выборкой может вычислить одиночное значение из диапазона значений во внутренней таблице.

Функция над выборкой заменяет следующие общие числовые функции: ***numsum****,* ***numavg****,* ***numcount****,* ***nummin*** *и* ***nummax****, которые теперь должны считаться устаревшими.*

Базовые функции над выборкой

RangeMax

RangeMax() возвращает самые высокие числовые значения, обнаруженные в выражении или поле.

```
RangeMax (first_expr[, Expression])
```

RangeMaxString

RangeMaxString() возвращает последнее значение в порядке сортировки текста, обнаруженное в выражении или поле.

```
RangeMaxString (first_expr[, Expression])
```

RangeMin

RangeMin() возвращает самые низкие числовые значения, обнаруженные в выражении или поле.

```
RangeMin (first_expr[, Expression])
```

RangeMinString

RangeMinString() возвращает первое значение в порядке сортировки текста, обнаруженное в выражении или поле.

```
RangeMinString (first_expr[, Expression])
```

RangeMode

RangeMode() находит наиболее часто встречающееся значение (значение режима) в выражении или поле.

```
RangeMode (first_expr[, Expression])
```

RangeOnly

RangeOnly() – это функция dual, которая возвращает значение, если выражение оценивает до одного уникального значения. Если это другой случай, тогда возвращается значение **NULL**.

```
RangeOnly (first_expr[, Expression])
```

RangeSum

RangeSum() возвращает сумму диапазона значений. Все нечисловые значения обрабатываются как 0.

```
RangeSum (first_expr[, Expression])
```

Функции над выборкой счетчика

RangeCount

RangeCount() возвращает количество текстовых и числовых значений в выражении или поле.

```
RangeCount (first_expr[, Expression])
```

RangeMissingCount

RangeMissingCount() возвращает количество нечисловых значений (включая NULL) в выражении или поле.

```
RangeMissingCount (first_expr[, Expression])
```

RangeNullCount

RangeNullCount() находит значения NULL в выражении или поле.

```
RangeNullCount (first_expr[, Expression])
```

RangeNumericCount

RangeNumericCount() находит числовые значения в выражении или поле.

```
RangeNumericCount (first_expr[, Expression])
```

RangeTextCount

RangeTextCount() возвращает текстовые значения в выражении или поле.

```
RangeTextCount (first_expr[, Expression])
```

Статистические функции над выборкой

RangeAvg

RangeAvg() возвращает среднее значение диапазона. Для ввода в функцию может использоваться диапазон значений или выражение.

```
RangeAvg (first_expr[, Expression])
```

RangeCorrel

RangeCorrel() возвращает коэффициент корреляции для двух наборов данных. Коэффициент корреляции – это мера отношений между наборами данных.

```
RangeCorrel (x_values , y_values[, Expression])
```

RangeFractile

RangeFractile() возвращает значение, соответствующее n-ному **fractile** (квантилю) диапазона чисел.

```
RangeFractile (fractile, first_expr[ ,Expression])
```

RangeKurtosis

RangeKurtosis() возвращает значение, соответствующее эксцессу диапазона чисел.

```
RangeKurtosis (first_expr[, Expression])
```

RangeSkew

RangeSkew() возвращает значение, соответствующее асимметрии диапазона чисел.

```
RangeSkew (first_expr[, Expression])
```

RangeStdev

RangeStdev() находит стандартное отклонение диапазона чисел.

```
RangeStdev (expr1[, Expression])
```

Финансовые функции над выборкой

RangeIRR

RangeIRR() возвращает внутреннюю ставку доходов для серии потоков денежных средств, представленных вводимыми значениями.

```
RangeIRR (value[, value][, Expression])
```

RangeNPV

RangeNPV() возвращает чистую стоимость инвестиций на основе льготного тарифа, серии будущих периодических платежей (отрицательные значения) и дохода (положительные значения). Результат имеет формат числа **money** по умолчанию.

```
RangeNPV (discount_rate, value[, value][, Expression])
```

RangeXIRR

RangeXIRR() возвращает внутреннюю ставку доходов для расписания потоков денежных средств, которые не обязательно периодические. Для вычисления внутренней ставки доходов для серии периодических потоков денежных средств необходимо использовать функцию **RangeIRR**.

```
RangeXIRR (values, dates[, Expression])
```

RangeXNPV

RangeXNPV() возвращает чистую стоимость для графика потоков денежных средств (необязательно периодических). Результат имеет числовой денежный формат по умолчанию. Для вычисления чистой стоимости для серии периодических потоков денежных средств необходимо использовать функцию **RangeNPV**.

```
RangeXNPV (discount_rate, values, dates[, Expression])
```

См. также:

Функции между записями (page 686)

RangeAvg

RangeAvg() возвращает среднее значение диапазона. Для ввода в функцию может использоваться диапазон значений или выражение.

Синтаксис:

```
RangeAvg(first_expr[, Expression])
```

Возвращаемые типы данных: числовое значение

Аргументы:

Аргументы этой функции могут содержать межзаписные функции, которые в свою очередь возвращают список значений.

Аргументы

Аргумент	Описание
first_expr	Выражение или поле, содержащее данные для измерения.
Expression	Дополнительные выражения или поля, содержащие диапазон значений для измерения.

Ограничения:

Если числовые значения не найдены, возвращается значение NULL.

Примеры и результаты:

Примеры написания скриптов

Примеры	Результаты
`RangeAvg (1,2,4)`	Возвращает 2,33333333
`RangeAvg (1,'xyz')`	Возвращает 1
`RangeAvg (null( ), 'abc')`	Возвращает NULL

Пример:

Добавьте образец скрипта в свое приложение и запустите. Чтобы увидеть результаты, добавьте поля, перечисленные в столбце результатов, в лист приложения.

```
RangeTab3:
LOAD recno() as RangeID, RangeAvg(Field1,Field2,Field3) as MyRangeAvg INLINE [
Field1, Field2, Field3
10,5,6
2,3,7
8,2,8
18,11,9
5,5,9
9,4,2
];
```

Результирующая таблица показывает возвращенные значения функции MyRangeAvg для каждой записи в таблице.

Результирующая таблица

RangeID	MyRangeAvg
1	7
2	4
3	6
4	12.666
5	6.333
6	5

Пример с выражением:

```
RangeAvg (Above(MyField),0,3))
```

Возвращает скользящее среднее результата диапазона из трех значений поля **MyField**, вычисленного в текущей строке и двух строках над ней. При указании третьего аргумента как 3 функция **Above()** возвращает три значения, над которыми достаточно строк и которые принимаются за вводимые значения в функцию **RangeAvg()**.

Данные, используемые в примерах:

*Отключите сортировку **MyField**, чтобы убедиться, что пример работает, как ожидается.*

Данные образца

MyField	RangeAvg (Above (MyField,0,3))	Comments
10	10	Поскольку данная строка является верхней, диапазон включает только одно значение.
2	6	Над этой строкой только одна строка, поэтому диапазон: 10,2.
8	6.6666666667	Эквивалент для RangeAvg(10,2,8)
18	9.333333333	-
5	10. 333333333	-
9	10.6666666667	-

```
RangeTab:
LOAD * INLINE [
MyField
10
2
8
18
5
9
] ;
```

См. также:

- *Avg – функция диаграммы (page 290)*
- *Count – функция диаграммы (page 253)*

RangeCorrel

RangeCorrel() возвращает коэффициент корреляции для двух наборов данных. Коэффициент корреляции – это мера отношений между наборами данных.

Синтаксис:

```
RangeCorrel(x_value , y_value[, Expression])
```

Возвращаемые типы данных: числовое значение

Серии данных необходимо добавлять в виде пар (x,y). Например, чтобы оценить две серии данных, диапазон 1 и диапазон 2, где диапазон 1 = 2, 6, 9, а диапазон 2 = 3, 8, 4, необходимо записать элемент `RangeCorrel (2,3,6,8,9,4)`, который возвращает значение 0,269.

Аргументы:

Аргументы

Аргумент	Описание
x-value, y-value	Каждое значение является одиночным значением или диапазоном значений, возвращаемых функциями между записями с третьим дополнительным параметром. Каждое значение или диапазон значений должны соответствовать значению **x-value** или диапазону значений **y-values**.
Expression	Дополнительные выражения или поля, содержащие диапазон значений для измерения.

Ограничения:

Для вычисления функции требуется хотя бы две пары координат.

Текстовые значения, значения NULL и отсутствующие значения возвращают NULL.

Примеры и результаты:

Примеры функции

Примеры	Результаты
`RangeCorrel (2,3,6,8,9,4,8,5)`	Возвращает 0,2492. Данную функцию можно загрузить в скрипт или добавить в визуализацию в редакторе выражения.

Пример:

Добавьте образец скрипта в свое приложение и запустите. Чтобы увидеть результаты, добавьте поля, перечисленные в столбце результатов, в лист приложения.

```
RangeList:
Load * Inline [
ID1|x1|y1|x2|y2|x3|y3|x4|y4|x5|y5|x6|y6
01|46|60|70|13|78|20|45|65|78|12|78|22
02|65|56|22|79|12|56|45|24|32|78|55|15
03|77|68|34|91|24|68|57|36|44|90|67|27
04|57|36|44|90|67|27|57|68|47|90|80|94
] (delimiter is '|');

XY:
LOAD recno() as RangeID, * Inline [
X|Y
2|3
6|8
9|4
8|5
](delimiter is '|');
```

В таблице с измерением ID1 и следующей мерой: RangeCorrel(x1,y1,x2,y2,x3,y3,x4,y4,x5,y5,x6,y6)) функция **RangeCorrel()** приобретает значение **Correl**, находящееся в диапазоне шести повторяемых значений x,y для каждого из значений ID1.

Результирующая таблица

ID1	MyRangeCorrel
01	-0.9517
02	-0.5209
03	-0.5209
04	-0.1599

Пример:

```
XY:
LOAD recno() as RangeID, * Inline [
X|Y
2|3
6|8
9|4
8|5
](delimiter is '|');
```

В таблице с измерением RangeID и следующей мерой: RangeCorrel(Below(X,0,4,BelowY,0,4)) функция **RangeCorrel()** использует результаты функций **Below()**, которые на основе третьего аргумента (count), установленного на значение 4, формируют диапазон из четырех значений x-y из загруженной таблицы XY.

Результирующая таблица

RangeID	MyRangeCorrel2
01	0.2492
02	-0.9959
03	-1.0000
04	-

Значение для параметра RangeID 01 совпадает с введенным вручную значением RangeCorrel (2,3,6,8,9,4,8,5). Для других значений параметра RangeID функция Below() приводит к созданию последовательности следующего вида: (6,8,9,4,8,5), (9,4,8,5) и (8,5), где последнее значение выводит нулевой результат.

См. также:

Correl – функция диаграммы (page 293)

RangeCount

RangeCount() возвращает количество текстовых и числовых значений в выражении или поле.

Синтаксис:

```
RangeCount(first_expr[, Expression])
```

Возвращаемые типы данных: целое

Аргументы:

Аргументы этой функции могут содержать межзаписные функции, которые в свою очередь возвращают список значений.

Аргументы

Аргумент	Описание
first_expr	Выражение или поле, содержащее данные для подсчета.
Expression	Дополнительные выражения или поля, содержащие диапазон значений для подсчета.

Ограничения:

Значения NULL не учитываются.

Примеры и результаты:

Примеры функции

Примеры	Результаты
`RangeCount (1,2,4)`	Возвращает 3
`RangeCount (2,'xyz')`	Возвращает 2
`RangeCount (null( ))`	Возвращает 0
`RangeCount (2,'xyz', null())`	Возвращает 2

Пример:

Добавьте образец скрипта в свое приложение и запустите. Чтобы увидеть результаты, добавьте поля, перечисленные в столбце результатов, в лист приложения.

```
RangeTab3:
LOAD recno() as RangeID, RangeCount(Field1,Field2,Field3) as MyRangeCount INLINE [
Field1, Field2, Field3
10,5,6
2,3,7
8,2,8
18,11,9
5,5,9
```

```
9,4,2
];
```

Результирующая таблица показывает возвращенные значения функции MyRangeCount для каждой записи в таблице.

Результирующая таблица

RangeID	MyRangeCount
1	3
2	3
3	3
4	3
5	3
6	3

Пример с выражением:

```
RangeCount (Above(MyField,1,3))
```

Возвращает значения, содержащиеся в трех результатах поля **MyField**. Если задать для первого аргумента функции **Above()** значение 1, а для второго аргумента – значение 3, будут возвращены значения из трех полей над текущей строкой, если имеется достаточно строк, которые принимаются как значения, вводимые в функцию **RangeCount()**.

Данные, используемые в примерах:

Данные образца

MyField	RangeCount(Above(MyField,1,3))
10	0
2	1
8	2
18	3
5	3
9	3

Данные, используемые в примерах:

```
RangeTab:
LOAD * INLINE [
MyField
10
2
8
18
5
9
] ;
```

См. также:

- *Count – функция диаграммы (page 253)*

RangeFractile

RangeFractile() возвращает значение, соответствующее n-ному **fractile** (квантилю) диапазона чисел.

> *RangeFractile() использует линейное интерполирование между ближайшими рядами при вычислении квантиля.*

Синтаксис:

```
RangeFractile(fractile, first_expr[, Expression])
```

Возвращаемые типы данных: числовое значение

Аргументы:

Аргументы этой функции могут содержать межзаписные функции, которые в свою очередь возвращают список значений.

Аргументы

Аргумент	Описание
fractile	Число от 0 до 1, соответствующее квантилю (выраженному в дробном виде), которое подлежит вычислению.
first_expr	Выражение или поле, содержащее данные для измерения.
Expression	Дополнительные выражения или поля, содержащие диапазон значений для измерения.

Примеры и результаты:

Примеры функции

Примеры	Результаты
`RangeFractile (0.24,1,2,4,6)`	Возвращает 1,72
`RangeFractile(0.5,1,2,3,4,6)`	Возвращает 3
`RangeFractile (0.5,1,2,5,6)`	Возвращает 3,5

Пример:

Добавьте образец скрипта в свое приложение и запустите. Чтобы увидеть результаты, добавьте поля, перечисленные в столбце результатов, в лист приложения.

```
RangeTab:
LOAD recno() as RangeID, RangeFractile(0.5,Field1,Field2,Field3) as MyRangeFrac INLINE [
Field1, Field2, Field3
10,5,6
2,3,7
8,2,8
18,11,9
5,5,9
9,4,2
];
```

Результирующая таблица показывает возвращенные значения функции MyRangeFrac для каждой записи в таблице.

Результирующая таблица

RangeID	MyRangeFrac
1	6
2	3
3	8
4	11
5	5
6	4

Пример с выражением:

```
RangeFractile (0.5, Above(Sum(MyField),0,3))
```

В этом примере функция между записями **Above()** содержит дополнительные аргументы offset и count. В результате формируется диапазон результатов, который можно использовать в качестве данных, вводимых в любую функцию над выборкой. В этом случае функция `Above(Sum(MyField),0,3)` возвращает значения поля `MyField` для текущей строки и двух строк над ней. Эти значения обеспечивают значения, вводимые в функцию **RangeFractile()**. Таким образом, для нижней строки в таблице ниже это является эквивалентом `RangeFractile(0.5, 3,4,6)`, т. е. вычисление квантиля 0,5 для серий 3, 4 и 6. Первые две строки в таблице ниже. Количество значений в диапазоне сокращается соответственно, если над текущей строкой нет других строк. Похожие результаты будут и для других межзаписных функций.

Данные образца

MyField	RangeFractile(0.5, Above(Sum(MyField),0,3))
1	1
2	1.5
3	2

MyField	RangeFractile(0.5, Above(Sum(MyField),0,3))
4	3
5	4
6	5

Данные, используемые в примерах:

```
RangeTab:
LOAD * INLINE [
MyField
1
2
3
4
5
6
] ;
```

См. также:

- *Above – функция диаграммы (page 689)*
- *Fractile – функция диаграммы (page 297)*

RangeIRR

RangeIRR() возвращает внутреннюю ставку доходов для серии потоков денежных средств, представленных вводимыми значениями.

Внутренняя ставка доходов – это процентная ставка для инвестиций, состоящих из платежей (отрицательные значения) и дохода (положительные значения), осуществляемых регулярно.

Синтаксис:

```
RangeIRR(value[, value][, Expression])
```

Возвращаемые типы данных: числовое значение

Аргументы:

Аргументы

Аргумент	Описание
value	Одиночное значение или диапазон значений, возвращаемые функциями между записями с третьим дополнительным параметром. Для вычисления этой функции необходимо по крайней мере одно положительное и одно отрицательное значение.
Expression	Дополнительные выражения или поля, содержащие диапазон значений для измерения.

Ограничения:

Текстовые значения, значения NULL и отсутствующие значения игнорируются.

Пример таблицы

Примеры	Результаты
`RangeIRR(-70000,12000,15000,18000,21000,26000)`	Возвращает 0,0866

Добавьте образец скрипта в свое приложение и запустите. Чтобы увидеть результаты, добавьте поля, перечисленные в столбце результатов, в лист приложения.
`RangeTab3:`
`LOAD *,`
`recno() as RangeID,`
`RangeIRR(Field1,Field2,Field3) as RangeIRR;`
`LOAD * INLINE [`
`Field1|Field2|Field3`
`-10000|5000|6000`
`-2000|NULL|7000`
`-8000|'abc'|8000`
`-1800|11000|9000`
`-5000|5000|9000`
`-9000|4000|2000`
`] (delimiter is '|');`
	Результирующая таблица показывает возвращенные значения функции RangeIRR для каждой записи в таблице.
	RangeID — RangeIRR
	1 — 0.0639
	2 — 0.8708
	3 — -
	4 — 5.8419
	5 — 0.9318
	6 — -0.2566

См. также:

- *Функции между записями (page 686)*

RangeKurtosis

RangeKurtosis() возвращает значение, соответствующее эксцессу диапазона чисел.

Синтаксис:

```
RangeKurtosis(first_expr[, Expression])
```

Возвращаемые типы данных: числовое значение

Аргументы:

Аргументы этой функции могут содержать межзаписные функции, которые в свою очередь возвращают список значений.

Аргументы

Аргумент	Описание
first_expr	Выражение или поле, содержащее данные для измерения.

Аргумент	Описание
Expression	Дополнительные выражения или поля, содержащие диапазон значений для измерения.

Ограничения:

Если числовые значения не найдены, возвращается значение NULL.

Примеры и результаты:

Примеры функции

Примеры	Результаты
`RangeKurtosis (1,2,4,7)`	Возвращает -0,28571428571429

См. также:

Kurtosis – функция диаграммы (page 305)

RangeMax

RangeMax() возвращает самые высокие числовые значения, обнаруженные в выражении или поле.

Синтаксис:

```
RangeMax(first_expr[, Expression])
```

Возвращаемые типы данных: числовое значение

Аргументы:

Аргументы

Аргумент	Описание
first_expr	Выражение или поле, содержащее данные для измерения.
Expression	Дополнительные выражения или поля, содержащие диапазон значений для измерения.

Ограничения:

Если числовые значения не найдены, возвращается значение NULL.

Примеры и результаты:

Примеры функции

Примеры	Результаты
`RangeMax (1,2,4)`	Возвращает 4
`RangeMax (1,'xyz')`	Возвращает 1
`RangeMax (null( ), 'abc')`	Возвращает NULL

Пример:

Добавьте образец скрипта в свое приложение и запустите. Чтобы увидеть результаты, добавьте поля, перечисленные в столбце результатов, в лист приложения.

```
RangeTab3:
LOAD recno() as RangeID, RangeMax(Field1,Field2,Field3) as MyRangeMax INLINE [
Field1, Field2, Field3
10,5,6
2,3,7
8,2,8
18,11,9
5,5,9
9,4,2
];
```

Результирующая таблица показывает возвращенные значения функции MyRangeMax для каждой записи в таблице.

Результирующая таблица

RangeID	MyRangeMax
1	10
2	7
3	8
4	18
5	9
6	9

Пример с выражением:

```
RangeMax (Above(MyField,0,3))
```

Возвращает максимальное значение в диапазоне из трех значений поля **MyField**, вычисленных в текущей строке и двух строках над ней. При указании третьего аргумента как 3 функция **Above()** возвращает три значения, над которыми достаточно строк и которые принимаются за вводимые значения в функцию **RangeMax()**.

Данные, используемые в примерах:

*Отключите сортировку **MyField**, чтобы убедиться, что пример работает, как ожидается.*

Данные образца

MyField	RangeMax (Above(Sum(MyField),1,3))
10	10
2	10
8	10
18	18
5	18
9	18

Данные, используемые в примерах:

```
RangeTab:
LOAD * INLINE [
MyField
10
2
8
18
5
9
] ;
```

RangeMaxString

RangeMaxString() возвращает последнее значение в порядке сортировки текста, обнаруженное в выражении или поле.

Синтаксис:

```
RangeMaxString(first_expr[, Expression])
```

Возвращаемые типы данных: строка

Аргументы:

Аргументы этой функции могут содержать межзаписные функции, которые в свою очередь возвращают список значений.

Аргументы

Аргумент	Описание
first_expr	Выражение или поле, содержащее данные для измерения.
Expression	Дополнительные выражения или поля, содержащие диапазон значений для измерения.

Примеры и результаты:

Примеры функции

Примеры	Результаты
`RangeMaxString (1,2,4)`	Возвращает 4
`RangeMaxString ('xyz','abc')`	Возвращает «xyz»
`RangeMaxString (5,'abc')`	Возвращает «abc»
`RangeMaxString (null( ))`	Возвращает NULL

Пример с выражением:

```
RangeMaxString (Above(MaxString(MyField),0,3))
```

Возвращает последний (в порядке сортировки текста) из трех результатов функции **MaxString (MyField)**, оцененной для текущей строки и двух строк над ней.

Данные, используемые в примерах:

*Отключите сортировку **MyField**, чтобы убедиться, что пример работает, как ожидается.*

Данные образца

MyField	RangeMaxString(Above(MaxString(MyField),0,3))
10	10
abc	abc
8	abc
def	def
xyz	xyz
9	xyz

Данные, используемые в примерах:

```
RangeTab:
LOAD * INLINE [
MyField
10
'abc'
8
'def'
'xyz'
9
] ;
```

См. также:

- *MaxString – функция диаграммы (page 429)*

RangeMin

RangeMin() возвращает самые низкие числовые значения, обнаруженные в выражении или поле.

Синтаксис:

```
RangeMin(first_expr[, Expression])
```

Возвращаемые типы данных: числовое значение

Аргументы:

Аргументы

Аргумент	Описание
first_expr	Выражение или поле, содержащее данные для измерения.
Expression	Дополнительные выражения или поля, содержащие диапазон значений для измерения.

Ограничения:

Если числовые значения не найдены, возвращается значение NULL.

Примеры и результаты:

Примеры функции

Примеры	Результаты
`RangeMin (1,2,4)`	Возвращает 1
`RangeMin (1,'xyz')`	Возвращает 1
`RangeMin (null( ), 'abc')`	Возвращает NULL

Пример:

Добавьте образец скрипта в свое приложение и запустите. Чтобы увидеть результаты, добавьте поля, перечисленные в столбце результатов, в лист приложения.

```
RangeTab3:
LOAD recno() as RangeID, RangeMin(Field1,Field2,Field3) as MyRangeMin INLINE [
Field1, Field2, Field3
10,5,6
2,3,7
8,2,8
18,11,9
5,5,9
```

```
9,4,2
];
```

Результирующая таблица показывает возвращенные значения функции MyRangeMin для каждой записи в таблице.

Результирующая таблица

RangeID	MyRangeMin
1	5
2	2
3	2
4	9
5	5
6	2

Пример с выражением:

```
RangeMin (Above(MyField,0,3)
```

Возвращает минимальное значение в диапазоне из трех значений поля **MyField**, вычисленных в текущей строке и двух строках над ней. При указании третьего аргумента как 3 функция **Above()** возвращает три значения, над которыми достаточно строк и которые принимаются за вводимые значения в функцию **RangeMin()**.

Данные, используемые в примерах:

Данные образца

MyField	RangeMin(Above(MyField,0,3))
10	10
2	2
8	2
18	2
5	5
9	5

Данные, используемые в примерах:

```
RangeTab:
LOAD * INLINE [
MyField
10
2
8
18
5
9
```

```
] ;
```

См. также:

Min – функция диаграммы (page 239)

RangeMinString

RangeMinString() возвращает первое значение в порядке сортировки текста, обнаруженное в выражении или поле.

Синтаксис:

```
RangeMinString(first_expr[, Expression])
```

Возвращаемые типы данных: строка

Аргументы:

Аргументы этой функции могут содержать межзаписные функции, которые в свою очередь возвращают список значений.

Аргументы

Аргумент	Описание
first_expr	Выражение или поле, содержащее данные для измерения.
Expression	Дополнительные выражения или поля, содержащие диапазон значений для измерения.

Примеры и результаты:

Примеры функции

Примеры	Результаты
`RangeMinString (1,2,4)`	Возвращает 1
`RangeMinString ('xyz','abc')`	Возвращает «abc»
`RangeMinString (5,'abc')`	Возвращает 5
`RangeMinString (null( ))`	Возвращает NULL

Пример с выражением:

```
RangeMinString (Above(MinString(MyField),0,3))
```

Возвращает первый (в порядке сортировки текста) из трех результатов функции **MinString (MyField)**, оцененной для текущей строки и двух строк над ней.

Данные, используемые в примерах:

*Отключите сортировку **MyField**, чтобы убедиться, что пример работает, как ожидается.*

Данные образца

MyField	RangeMinString(Above(MinString(MyField),0,3))
10	10
abc	10
8	8
def	8
xyz	8
9	9

Данные, используемые в примерах:

```
RangeTab:
LOAD * INLINE [
MyField
10
'abc'
8
'def'
'xyz'
9
] ;
```

См. также:

MinString – функция диаграммы (page 432)

RangeMissingCount

RangeMissingCount() возвращает количество нечисловых значений (включая NULL) в выражении или поле.

Синтаксис:

```
RangeMissingCount(first_expr[, Expression])
```

Возвращаемые типы данных: целое

Аргументы:

Аргументы этой функции могут содержать межзаписные функции, которые в свою очередь возвращают список значений.

Аргументы

Аргумент	Описание
first_expr	Выражение или поле, содержащее данные для подсчета.
Expression	Дополнительные выражения или поля, содержащие диапазон значений для подсчета.

Примеры и результаты:

Примеры функции

Примеры	Результаты
`RangeMissingCount (1,2,4)`	Возвращает 0
`RangeMissingCount (5,'abc')`	Возвращает 1
`RangeMissingCount (null( ))`	Возвращает 1

Пример с выражением:

```
RangeMissingCount (Above(MinString(MyField),0,3))
```

Возвращает число нечисловых значений из трех результатов функции **MinString(MyField)**, оцененной для текущей строки и двух строк над ней.

*Отключите сортировку **MyField**, чтобы убедиться, что пример работает, как ожидается.*

Данные образца

MyField	RangeMissingCount (Above(MinString (MyField),0,3))	Explanation
10	2	Возвращает 2, поскольку над этой строкой нет строк, поэтому 2 из 3 значений отсутствуют.
abc	2	Возвращает 2, поскольку над текущей строкой есть только 1 строка, а текущая строка не числовая ("abc").
8	1	Возвращает 1, поскольку 1 из 3 строк включает нечисловое ("abc").
def	2	Возвращает 2, поскольку 2 из 3 строк включают нечисловые значения ("def" и "abc").
xyz	2	Возвращает 2, поскольку 2 из 3 строк включают нечисловые значения (" xyz" и "def").
9	2	Возвращает 2, поскольку 2 из 3 строк включают нечисловые значения (" xyz" и "def").

Данные, используемые в примерах:

```
RangeTab:
LOAD * INLINE [
MyField
10
'abc'
8
'def'
'xyz'
9
] ;
```

См. также:

MissingCount – функция диаграммы (page 257)

RangeMode

RangeMode() находит наиболее часто встречающееся значение (значение режима) в выражении или поле.

Синтаксис:

```
RangeMode(first_expr {, Expression})
```

Возвращаемые типы данных: числовое значение

Аргументы:

Аргументы этой функции могут содержать межзаписные функции, которые в свою очередь возвращают список значений.

Аргументы

Аргумент	Описание
first_expr	Выражение или поле, содержащее данные для измерения.
Expression	Дополнительные выражения или поля, содержащие диапазон значений для измерения.

Ограничения:

Если одинаково часто встречаются несколько значений, возвращается значение NULL.

Примеры и результаты:

Примеры функции

Примеры	Результаты
`RangeMode (1,2,9,2,4)`	Возвращает 2

Примеры	Результаты
`RangeMode ('a',4,'a',4)`	Возвращает NULL
`RangeMode (null( ))`	Возвращает NULL

Пример:

Добавьте образец скрипта в свое приложение и запустите. Чтобы увидеть результаты, добавьте поля, перечисленные в столбце результатов, в лист приложения.

```
RangeTab3:
LOAD recno() as RangeID, RangeMode(Field1,Field2,Field3) as MyRangeMode INLINE [
Field1, Field2, Field3
10,5,6
2,3,7
8,2,8
18,11,9
5,5,9
9,4,2
];
```

Результирующая таблица показывает возвращенные значения функции **MyRangeMode** для каждой записи в таблице.

Результирующая таблица

RangeID	MyRangMode
1	-
2	-
3	8
4	-
5	5
6	-

Пример с выражением:

```
RangeMode (Above(MyField,0,3))
```

Возвращает наиболее часто встречающееся значение из трех результатов поля **MyField**, вычисленных в текущей строке и двух строках над ней. При указании третьего аргумента как 3 функция **Above()** возвращает три значения, над которыми достаточно строк и которые принимаются за вводимые значения в функцию **RangeMode()**.

Данные, используемые в примере:

```
RangeTab:
LOAD * INLINE [
MyField
10
2
```

```
8
18
5
9
] ;
```

*Отключите сортировку **MyField**, чтобы убедиться, что пример работает, как ожидается.*

Данные образца

MyField	RangeMode(Above(MyField,0,3))
10	Возвращает 10, поскольку выше нет строк, поэтому одно значение является наиболее часто встречающимся.
2	-
8	-
18	-
5	-
9	-

См. также:

- *Mode – функция диаграммы (page 243)*

RangeNPV

RangeNPV() возвращает чистую стоимость инвестиций на основе льготного тарифа, серии будущих периодических платежей (отрицательные значения) и дохода (положительные значения). Результат имеет формат числа **money** по умолчанию.

Сведения о потоках денежных средств, не обязательно являющихся периодическими, см. в *RangeXNPV (page 781)*.

Синтаксис:

```
RangeNPV(discount_rate, value[,value][, Expression])
```

Возвращаемые типы данных: числовое значение

Аргументы:

Аргументы

Аргумент	Описание
discount_rate	Процентная ставка за период.

Аргумент	Описание
value	Платеж или поступление в конце каждого периода. Каждое значение может быть одиночным значением или диапазоном значений, возвращаемым функцией между записями с третьим дополнительным параметром.
Expression	Дополнительные выражения или поля, содержащие диапазон значений для измерения.

Ограничения:

Текстовые значения, значения NULL и отсутствующие значения игнорируются.

Примеры	Результаты
`RangeNPV(0.1,-10000,3000,4200,6800)`	Возвращает 1188,44

Добавьте образец скрипта в свое приложение и запустите. Чтобы увидеть результаты, добавьте поля, перечисленные в столбце результатов, в лист приложения.
`RangeTab3:`
`LOAD *,`
`recno() as RangeID,`
`RangeNPV(Field1,Field2,Field3) as RangeNPV;`
`LOAD * INLINE [`
`Field1|Field2|Field3`
`10|5|-6000`
`2|NULL|7000`
`8|'abc'|8000`
`18|11|9000`
`5|5|9000`
`9|4|2000`
`] (delimiter is '|');`
	Результирующая таблица показывает возвращенные значения функции RangeNPV для каждой записи в таблице.
	RangeID RangeNPV
	1 $-49.13
	2 $777.78
	3 $98.77
	4 $25.51
	5 $250.83
	6 $20.40

См. также:

Функции между записями (page 686)

RangeNullCount

RangeNullCount() находит значения NULL в выражении или поле.

Синтаксис:

```
RangeNullCount(first_expr [, Expression])
```

Возвращаемые типы данных: целое

Аргументы:

Аргументы этой функции могут содержать межзаписные функции, которые в свою очередь возвращают список значений.

Аргументы

Аргумент	Описание
first_expr	Выражение или поле, содержащее данные для измерения.
Expression	Дополнительные выражения или поля, содержащие диапазон значений для измерения.

Примеры и результаты:

Примеры функции

Примеры	Результаты
`RangeNullCount (1,2,4)`	Возвращает 0
`RangeNullCount (5,'abc')`	Возвращает 0
`RangeNullCount (null( ), null( ))`	Возвращает 2

Пример с выражением:

```
RangeNullCount (Above(Sum(MyField),0,3))
```

Возвращает число значений NULL в трех результатах функции **Sum(MyField)**, оцененной для текущей строки и двух строк над ней.

*Копирование элемента **MyField** в примере ниже не приведет к получению значения NULL.*

Данные образца

MyField	RangeNullCount(Above(Sum(MyField),0,3))
10	Возвращает 2, поскольку над этой строкой нет строк, поэтому 2 из трех значений отсутствуют (=NULL).
'abc'	Возвращает 1, поскольку над текущей строкой есть только одна строка, поэтому одно из трех значений отсутствует (=NULL).
8	Возвращает 0, поскольку значение ни в одной из трех строк не является значением NULL.

Данные, используемые в примерах:

```
RangeTab:
```

```
LOAD * INLINE [
MyField
10
'abc'
8
] ;
```

См. также:

- *NullCount – функция диаграммы (page 260)*

RangeNumericCount

RangeNumericCount() находит числовые значения в выражении или поле.

Синтаксис:

```
RangeNumericCount(first_expr[, Expression])
```

Возвращаемые типы данных: целое

Аргументы:

Аргументы этой функции могут содержать межзаписные функции, которые в свою очередь возвращают список значений.

Аргументы

Аргумент	Описание
first_expr	Выражение или поле, содержащее данные для измерения.
Expression	Дополнительные выражения или поля, содержащие диапазон значений для измерения.

Примеры и результаты:

Примеры функции

Примеры	Результаты
`RangeNumericCount (1,2,4)`	Возвращает 3
`RangeNumericCount (5,'abc')`	Возвращает 1
`RangeNumericCount (null( ))`	Возвращает 0

Пример с выражением:

```
RangeNumericCount (Above(MaxString(MyField),0,3))
```

Возвращает число числовых значений в трех результатах функции **MaxString(MyField)**, оцененной в текущей строке и двух строках над ней.

*Отключите сортировку **MyField**, чтобы убедиться, что пример работает, как ожидается.*

Данные образца

MyField	RangeNumericCount(Above(MaxString(MyField),0,3))
10	1
abc	1
8	2
def	1
xyz	1
9	1

Данные, используемые в примерах:

```
RangeTab:
LOAD * INLINE [
MyField
10
'abc'
8
def
xyz
9
] ;
```

См. также:

NumericCount – функция диаграммы (page 263)

RangeOnly

RangeOnly() – это функция dual, которая возвращает значение, если выражение оценивает до одного уникального значения. Если это другой случай, тогда возвращается значение **NULL**.

Синтаксис:

```
RangeOnly(first_expr[, Expression])
```

Возвращаемые типы данных: двойное значение

Аргументы:

Аргументы этой функции могут содержать межзаписные функции, которые в свою очередь возвращают список значений.

Аргумент	Описание
first_expr	Выражение или поле, содержащее данные для измерения.
Expression	Дополнительные выражения или поля, содержащие диапазон значений для измерения.

Примеры и результаты:

Примеры	Результаты
`RangeOnly (1,2,4)`	Возвращает NULL
`RangeOnly (5,'abc')`	Возвращает NULL
`RangeOnly (null( ), 'abc')`	Возвращает «abc»
`RangeOnly(10,10,10)`	Возвращает 10

См. также:

Only – функция диаграммы (page 245)

RangeSkew

RangeSkew() возвращает значение, соответствующее асимметрии диапазона чисел.

Синтаксис:

```
RangeSkew(first_expr[, Expression])
```

Возвращаемые типы данных: числовое значение

Аргументы:

Аргументы этой функции могут содержать межзаписные функции, которые в свою очередь возвращают список значений.

Аргументы

Аргумент	Описание
first_expr	Выражение или поле, содержащее данные для измерения.
Expression	Дополнительные выражения или поля, содержащие диапазон значений для измерения.

Ограничения:

Если числовые значения не найдены, возвращается значение NULL.

Примеры и результаты:

Примеры функции

Примеры	Результаты
`rangeskew (1,2,4)`	Возвращает 0,93521952958283
`rangeskew (above (SalesValue,0,3))`	Возвращает скользящую асимметрию диапазона из трех значений, возвращенных функцией above(), вычисленной для текущей строки и двух строк над ней.

Данные, используемые в примере:

Данные образца

CustID	RangeSkew(Above(SalesValue,0,3))
1-20	-, -, 0.5676, 0.8455, 1.0127, -0.8741, 1.7243, -1.7186, 1.5518, 1.4332, 0, 1.1066, 1.3458, 1.5636, 1.5439, 0.6952, -0.3766

```
SalesTable:
LOAD recno() as CustID, * inline [
SalesValue
101
163
126
139
167
86
83
22
32
70
108
124
176
113
95
32
42
92
61
21
] ;
```

См. также:

Skew – функция диаграммы (page 339)

RangeStdev

RangeStdev() находит стандартное отклонение диапазона чисел.

Синтаксис:

```
RangeStdev(first_expr[, Expression])
```

Возвращаемые типы данных: числовое значение

Аргументы:

Аргументы этой функции могут содержать межзаписные функции, которые в свою очередь возвращают список значений.

Аргументы

Аргумент	Описание
first_expr	Выражение или поле, содержащее данные для измерения.
Expression	Дополнительные выражения или поля, содержащие диапазон значений для измерения.

Ограничения:

Если числовые значения не найдены, возвращается значение NULL.

Примеры и результаты:

Примеры функции

Примеры	Результаты
`RangeStdev (1,2,4)`	Возвращает 1,5275252316519
`RangeStdev (null( ))`	Возвращает NULL
`RangeStdev (above (SalesValue),0,3))`	Возвращает скользящее стандартное значение диапазона из трех значений, возвращенных функцией above(), вычисленное для текущей строки и двух строк над ней.

Данные, используемые в примере:

Данные образца

CustID	RangeStdev(SalesValue, 0,3))
1-20	-,43.841, 34.192, 18.771, 20.953, 41.138, 47.655, 36.116, 32.716, 25.325, 38,000, 27.737, 35.553, 33.650, 42.532, 33.858, 32.146, 25.239, 35.595

```
SalesTable:
LOAD recno() as CustID, * inline [
SalesValue
101
163
126
139
```

```
167
86
83
22
32
70
108
124
176
113
95
32
42
92
61
21
] ;
```

См. также:

- *Stdev – функция диаграммы (page 342)*

RangeSum

RangeSum() возвращает сумму диапазона значений. Все нечисловые значения обрабатываются как 0.

Синтаксис:

```
RangeSum(first_expr[, Expression])
```

Возвращаемые типы данных: числовое значение

Аргументы:

Аргументы этой функции могут содержать межзаписные функции, которые в свою очередь возвращают список значений.

Аргументы

Аргумент	Описание
first_expr	Выражение или поле, содержащее данные для измерения.
Expression	Дополнительные выражения или поля, содержащие диапазон значений для измерения.

Ограничения:

Функция **RangeSum** считает все нечисловые значения равными 0.

Примеры и результаты:

Примеры

Примеры	Результаты
`RangeSum (1,2,4)`	Возвращает 7
`RangeSum (5,'abc')`	Возвращает 5
`RangeSum (null( ))`	Возвращает 0

Пример:

Добавьте образец скрипта в свое приложение и запустите. Чтобы увидеть результаты, добавьте поля, перечисленные в столбце результатов, в лист приложения.

```
RangeTab3:

LOAD recno() as RangeID, Rangesum(Field1,Field2,Field3) as MyRangeSum INLINE [

Field1, Field2, Field3

10,5,6

2,3,7

8,2,8

18,11,9

5,5,9

9,4,2
];
```

Результирующая таблица показывает возвращенные значения функции MyRangeSum для каждой записи в таблице.

Результирующая таблица

RangeID	MyRangeSum
1	21
2	12
3	18
4	38
5	19
6	15

Пример с выражением:

```
RangeSum (Above(MyField,0,3))
```

Возвращает сумму трех значений поля **MyField)** из текущей строки и двух строк над ней. При указании третьего аргумента как 3 функция **Above()** возвращает три значения, над которыми достаточно строк и которые принимаются за вводимые значения в функцию **RangeSum()**.

Данные, используемые в примерах:

*Отключите сортировку **MyField**, чтобы убедиться, что пример работает, как ожидается.*

Данные образца

MyField	RangeSum(Above(MyField,0,3))
10	10
2	12
8	20
18	28
5	31
9	32

Данные, используемые в примерах:

```
RangeTab:
LOAD * INLINE [
MyField
10
2
8
18
5
9
] ;
```

См. также:

- *Sum – функция диаграммы (page 248)*
- *Above – функция диаграммы (page 689)*

RangeTextCount

RangeTextCount() возвращает текстовые значения в выражении или поле.

Синтаксис:

```
RangeTextCount(first_expr[, Expression])
```

Возвращаемые типы данных: целое

Аргументы:

Аргументы этой функции могут содержать межзаписные функции, которые в свою очередь возвращают список значений.

Аргумент

Аргумент	Описание
first_expr	Выражение или поле, содержащее данные для измерения.
Expression	Дополнительные выражения или поля, содержащие диапазон значений для измерения.

Примеры и результаты:

Примеры функции

Примеры	Результаты
`RangeTextCount (1,2,4)`	Возвращает 0
`RangeTextCount (5,'abc')`	Возвращает 1
`RangeTextCount (null( ))`	Возвращает 0

Пример с выражением:

```
RangeTextCount (Above(MaxString(MyField),0,3))
```

Возвращает число текстовых значений в трех результатах функции **MaxString(MyField)**, оцененной для текущей строки и двух строк над ней.

Данные, используемые в примерах:

*Отключите сортировку **MyField**, чтобы убедиться, что пример работает, как ожидается.*

Данные, используемые в примере

MyField	MaxString(MyField)	RangeTextCount(Above(Sum(MyField),0,3))
10	10	0
abc	abc	1
8	8	1
def	def	2
xyz	xyz	2
9	9	2

Данные, используемые в примерах:

```
RangeTab:
LOAD * INLINE [
MyField
10
'abc'
8
null()
'xyz'
9
] ;
```

См. также:

TextCount – функция диаграммы (page 267)

RangeXIRR

RangeXIRR() возвращает внутреннюю ставку доходов для расписания потоков денежных средств, которые не обязательно периодические. Для вычисления внутренней ставки доходов для серии периодических потоков денежных средств необходимо использовать функцию **RangeIRR**.

Синтаксис:

```
RangeXIRR(value, date{, value, date})
```

Возвращаемые типы данных: числовое значение

Аргументы:

Аргументы

Аргумент	Описание
value	Поток денежных средств или серия потоков, соответствующие расписанию платежей по датам. Серия значений должна содержать по крайней мере одно положительное и отрицательное значения.
date	Дата платежа или расписание дат платежей, соответствующие потоку денежных средств.

Ограничения:

Текстовые значения, значения NULL и отсутствующие значения игнорируются.

Все платежи учитываются на основе года с 365 днями.

Примеры	Результаты
`RangeXIRR(-2500,'2008-01-01',2750,'2008-09-01')`	Возвращает 0,1532

См. также:

- *RangeIRR (page 755)*

RangeXNPV

RangeXNPV() возвращает чистую стоимость для графика потоков денежных средств (необязательно периодических). Результат имеет числовой денежный формат по умолчанию. Для вычисления чистой стоимости для серии периодических потоков денежных средств необходимо использовать функцию **RangeNPV**.

Синтаксис:

```
RangeXNPV(discount_rate, values, dates[, Expression])
```

Возвращаемые типы данных: числовое значение

Аргументы:

Аргументы

Аргумент	Описание
discount_rate	Процентная ставка за период.
values	Поток денежных средств или серия потоков, соответствующие расписанию платежей по датам. Каждое значение может быть одиночным значением или диапазоном значений, возвращаемым функцией между записями с третьим дополнительным параметром. Серия значений должна содержать по крайней мере одно положительное и отрицательное значения.
dates	Дата платежа или расписание дат платежей, соответствующие потоку денежных средств.

Ограничения:

Текстовые значения, значения NULL и отсутствующие значения игнорируются.

Все платежи учитываются на основе года с 365 днями.

Пример таблицы

Примеры	Результаты
`RangeXNPV(0.1, -2500,'2008-01-01',2750,'2008-09-01')`	Возвращает 80,25

Примеры	Результаты

Добавьте образец скрипта в свое приложение и запустите. Чтобы увидеть результаты, добавьте поля, перечисленные в столбце результатов, в лист приложения.
`RangeTab3:`
`LOAD *,`
`recno() as RangeID,`
`RangeXNPV(Field1,Field2,Field3) as RangeNPV;`
`LOAD * INLINE [`
`Field1|Field2|Field3`
`10|5|-6000`
`2|NULL|7000`
`8|'abc'|8000`
`18|11|9000`
`5|5|9000`
`9|4|2000`
`] (delimiter is '|');`
	Результирующая таблица показывает возвращенные значения функции RangeXNPV для каждой записи в таблице.
	RangeID — RangeXNPV
	1 — $-49.13
	2 — $777.78
	3 — $98.77
	4 — $25.51
	5 — $250.83
	6 — $20.40

5.22 Функции ранжирования и кластеризации

Эти функции могут использоваться только в выражениях диаграмм.

Функции ранжирования в диаграммах

При использовании данных функций автоматически отключается запрещение нулевых значений. Значения NULL игнорируются.

Rank

Rank() оценивает строки диаграммы в выражении и для каждой строки отображает относительное положение значения измерения, оцененного в выражении. При оценке выражения эта функция сравнивает результат с результатом других строк, содержащих текущий сегмент столбца, и возвращает ранжирование текущей строки в сегменте.

```
Rank — функция диаграммы([TOTAL [<fld {, fld}>]] expr[, mode[, fmt]])
```

HRank

HRank() оценивает выражение и сравнивает результат с результатом других столбцов, содержащих сегмент текущей строки сводной таблицы. Затем функция возвращает ранжирование текущего столбца в сегменте.

```
HRank — функция диаграммы([TOTAL] expr[, mode[, fmt]])
```

Функции кластеризации в диаграммах

KMeans2D

Группа свойств **Лицензия сайта** содержит свойства, связанные с лицензией для системы Qlik Sense. Все поля являются обязательными и должны быть заполнены.

Свойства лицензии сайта

Имя свойства	Описание
Имя владельца	Имя пользователя, который является владельцем продукта Qlik Sense.
Организация владельца	Название организации, членом которой является владелец продукта Qlik Sense.
Серийный номер	Серийный номер, присвоенный программному обеспечению Qlik Sense.
Контрольный номер	Серийный номер, присвоенный программному обеспечению Qlik Sense.
Доступ к LEF	License Enabler File (LEF), назначенный присвоенный программному обеспечению Qlik Sense.

KMeans2D() вычисляет строки диаграммы путем применения кластеризации методом k-средних; для каждой строки диаграммы отображается идентификатор кластера, которому была назначена эта точка диаграммы. Столбцы, используемые алгоритмом кластеризации, определены соответственно параметрами coordinate_1 и coordinate_2. Они оба являются агрегированиями. Количество созданных кластеров определяется параметром num_clusters. Данные могут быть при необходимости нормализованы с помощью параметра нормы.

```
KMeans2D — функция диаграммы(num_clusters, coordinate_1, coordinate_2 [,
norm])
```

KMeansND

KMeansND() вычисляет строки диаграммы путем применения кластеризации методом k-средних; для каждой строки диаграммы отображается идентификатор кластера, которому была назначена эта точка диаграммы. Столбцы, используемые алгоритмом кластеризации, определяются соответственно параметрами coordinate_1 и coordinate_2 и т. д. до n столбцов. Все они являются агрегированиями. Количество созданных кластеров определяется параметром num_clusters.

```
KMeansND — функция диаграммы(num_clusters, num_iter, coordinate_1,
coordinate_2 [,coordinate_3 [, ...]])
```

KMeansCentroid2D

KMeansCentroid2D() вычисляет строки диаграммы путем применения кластеризации методом k-средних; для каждой строки диаграммы отображается желаемая координата кластера, которому была назначена эта точка диаграммы. Столбцы, используемые алгоритмом кластеризации, определяются соответственно параметрами coordinate_1 и coordinate_2. Они оба являются агрегированиями. Количество созданных кластеров определяется параметром num_clusters. Данные могут быть при необходимости нормализованы с помощью параметра нормы.

```
KMeansCentroid2D — функция диаграммы(num_clusters, coordinate_no, coordinate_
1, coordinate_2 [, norm])
```

KMeansCentroidND

KMeansCentroidND() вычисляет строки диаграммы путем применения кластеризации методом k-средних; для каждой строки диаграммы отображается желаемая координата кластера, которому была назначена эта точка диаграммы. Столбцы, используемые алгоритмом кластеризации, определяются соответственно параметрами coordinate_1 и coordinate_2 и т. д. до n столбцов. Все они являются агрегированиями. Количество созданных кластеров определяется параметром num_clusters.

```
KMeansCentroidND – функция диаграммы(num_clusters, num_iter, coordinate_no,
coordinate_1, coordinate_2 [,coordinate_3 [, ...]])
```

Rank – функция диаграммы

Rank() оценивает строки диаграммы в выражении и для каждой строки отображает относительное положение значения измерения, оцененного в выражении. При оценке выражения эта функция сравнивает результат с результатом других строк, содержащих текущий сегмент столбца, и возвращает ранжирование текущей строки в сегменте.

Сегменты столбцов

	Region	Country	Population	Rank(Population)
Column segment #1	Americas	Mexico	128,932,753	2
	Americas	Canada	37,742,154	3
	Americas	United States of America	331,002,651	1
Column segment #2	Europe	Sweden	10,099,265	4
	Europe	United Kingdom	67,886,011	2
	Europe	France	65,273,511	3
	Europe	Germany	83,783,942	1

Для диаграмм, за исключением таблиц, сегмент текущего столбца определяется так, как он отображается в эквиваленте прямой таблицы диаграммы.

Синтаксис:

```
Rank([TOTAL] expr[, mode[, fmt]])
```

Возвращаемые типы данных: двойное значение

Аргументы:

Аргументы

Аргумент	Описание
expr	Выражение или поле, содержащее данные для измерения.
mode	Указывает числовое представление результата функции.
fmt	Указывает текстовое представление результата функции.
TOTAL	Если диаграмма имеет одно измерение, или если выражению предшествует квалификатор **TOTAL**, функция выполняет оценку по всему столбцу. Если таблица или эквивалент таблицы имеют несколько вертикальных измерений, текущий сегмент столбца будет включать только строки с теми же значениями, что и текущая строка во всех столбцах измерений, кроме столбца с последним измерением в межполевом порядке сортировки.

Ранжирование возвращается в виде двойного значения, которое, в случае если каждая строка имеет уникальное ранжирование, будет представлять собой целое число от 1 до количества строк в текущем сегменте столбца.

В случае, если несколько строк имеют одно и то же ранжирование, текстовое и числовое представления могут управляться параметрами **mode** и **fmt**.

mode

Второй аргумент, **mode**, может принимать следующие значения:

Примеры **mode**

Значение	Описание
0 (по умолчанию)	Если все ряды в совместно используемой группе выпадают на нижнюю часть среднего значения всего ранжирования, все строки получают низший ряд в совместно используемой группе. Если все ряды в совместно используемой группе выпадают на верхнюю часть среднего значения всего ранжирования, все строки получают высший ряд в совместно используемой группе. Если ряды в совместно используемой группе охватывают среднее значение всего ранжирования, все строки получают значение, соответствующее среднему значению верхнего и нижнего ранжирования во всем сегменте столбца.
1	Нижний ряд на всех строках.
2	Средний ряд на всех строках.
3	Высший ряд на всех строках.
4	Самый нижний ряд на первой строке, увеличенный на один для каждой строки.

fmt

Третий аргумент, **fmt**, может принимать следующие значения:

Примеры **fmt**

Значение	Описание
0 (по умолчанию)	Низкое значение – высокое значение во всех строках (например, 3-4).
1	Нижнее значение на всех строках.
2	Нижнее значение на первой строке, пустое на следующих строках.

Порядок строк **mode** 4 и **fmt** 2 определяется порядком сортировки измерений диаграммы.

Примеры и результаты:

Создайте две визуализации. Одну с измерениями Product и Sales, а вторую с измерениями Product и UnitSales. Добавьте меры, как показано на следующей таблице.

Примеры Rank

Примеры	Результаты
Пример 1. Создайте таблицу с измерениями `Customer` и `Sales` и мерой `Rank(Sales)`	Результат зависит от порядка сортировки измерений. Если таблица сортируется по элементу Customer, в таблице перечисляются все значения элемента Sales для элемента Astrida, затем для элемента Betacab и т. д. В результатах для элемента Rank(Sales) будет отображено значение 10 для значения Sales, равного 12, 9 для значения Sales, равного 13 и т. д. со значением ранжирования 1, возвращенного для значения Sales, равного 78. Следующий сегмент столбца начинается с элемента Betacab, для которого первое значение элемента Sales в сегменте равно 12. Значение ранжирования элемента Rank(Sales) дано для этого как 11. Если таблица сортируется по элементу Sales, сегменты столбца состоят из значений элемента Sales и соответствующего элемента Customer. Поскольку существует два значения элемента Sales, равных 12, (для Astrida и Betacab), значение элемента Rank(Sales) для этого сегмента столбца составляет 1-2 для каждого значения элемента Customer. Это потому, что существуют два значения элемента Customer, где элемент Sales равен 12. Если бы было 4 значения, результат был бы 1-4 для всех строк. На этом примере видно, как выглядит результат для значения по умолчанию (0) аргумента fmt.
Пример 2. Замените измерение Customer измерением Product и добавьте меру `Rank(Sales,1,2)`	Будет возвращено значение 1 в первой строке в сегменте каждого столбца, а все остальные строки останутся пустыми, поскольку для аргументов **mode** и **fmt** установлены значения 1 и 2 соответственно.

Результаты для примера 1 – таблица отсортирована по значению Customer:

Результирующая таблица

Customer	Sales	Rank(Sales)
Astrida	12	10

Customer	Sales	Rank(Sales)
Astrida	13	9
Astrida	20	8
Astrida	22	7
Astrida	45	6
Astrida	46	5
Astrida	60	4
Astrida	65	3
Astrida	70	2
Astrida	78	1
Betcab	12	11

Результаты для примера 1 – таблица отсортирована по значению Sales:

Результирующая таблица

Customer	Sales	Rank(Sales)
Astrida	12	1-2
Betacab	12	1-2
Astrida	13	1
Betacab	15	1
Astrida	20	1
Astrida	22	1-2
Betacab	22	1-2
Betacab	24	1-2
Canutility	24	1-2

Данные, используемые в примерах:

```
ProductData:

Load * inline [

Customer|Product|UnitSales|UnitPrice

Astrida|AA|4|16

Astrida|AA|10|15
```

```
Astrida|BB|9|9

Betacab|BB|5|10

Betacab|CC|2|20

Betacab|DD|0|25

Canutility|AA|8|15

Canutility|CC|0|19

] (delimiter is '|');


Sales2013:
crosstable (Month, Sales) LOAD * inline [
Customer|Jan|Feb|Mar|Apr|May|Jun|Jul|Aug|Sep|Oct|Nov|Dec
Astrida|46|60|70|13|78|20|45|65|78|12|78|22
Betacab|65|56|22|79|12|56|45|24|32|78|55|15
Canutility|77|68|34|91|24|68|57|36|44|90|67|27
Divadip|57|36|44|90|67|27|57|68|47|90|80|94
] (delimiter is '|');
```

См. также:

- *Sum – функция диаграммы (page 248)*

HRank – функция диаграммы

HRank() оценивает выражение и сравнивает результат с результатом других столбцов, содержащих сегмент текущей строки сводной таблицы. Затем функция возвращает ранжирование текущего столбца в сегменте.

Синтаксис:

```
HRank([ TOTAL ] expr [ , mode [, fmt ] ])
```

Возвращаемые типы данных: двойное значение

> *Эта функция доступна только при работе со сводными таблицами. Во всех других типах диаграмм она возвращает значение NULL.*

Аргументы:

Аргументы

Аргумент	Описание
expr	Выражение или поле, содержащее данные для измерения.
mode	Указывает числовое представление результата функции.
fmt	Указывает текстовое представление результата функции.
TOTAL	Если диаграмма имеет одно измерение, или если выражению предшествует квалификатор **TOTAL**, функция выполняет оценку по всему столбцу. Если таблица или эквивалент таблицы имеют несколько вертикальных измерений, текущий сегмент столбца будет включать только строки с теми же значениями, что и текущая строка во всех столбцах измерений, кроме столбца с последним измерением в межполевом порядке сортировки.

Если сводная таблица имеет одно измерение, или если перед выражением находится классификатор **total**, сегмент текущей строки всегда равен всей строке. Если сводная таблица имеет несколько горизонтальных измерений, текущий сегмент строки будет включать только столбцы с теми же значениями, что и текущий столбец во всех строках с измерениями, кроме строки с последним горизонтальным измерением в межполевом порядке сортировки.

Ранжирование возвращается в виде двойного значения, что, в случае, если каждый столбец имеет уникальное ранжирование, будет в диапазоне от 1 до количества столбцов в сегменте текущей строки.

В случае, если несколько столбцов имеют одно и то же ранжирование, текстовое и числовое представления могут управляться аргументами **mode** и **format**.

Второй аргумент **mode** указывает числовое представление результата функции:

Примеры **mode**

Значение	Описание
0 (по умолчанию)	Если все ряды в совместно используемой группе выпадают на нижнюю часть среднего значения всего ранжирования, все столбцы получают низший ряд в совместно используемой группе. Если все ряды в совместно используемой группе выпадают на верхнюю часть среднего значения всего ранжирования, все столбцы получают высший ряд в совместно используемой группе. Если ряды в совместно используемой группе охватывают среднее значение всего ранжирования, все строки получают значение, соответствующее среднему значению верхнего и нижнего ранжирования во всем сегменте столбца.

Значение	Описание
1	Самый нижний ряд на всех столбцах в группе.
2	Средний ряд на всех столбцах в группе.
3	Самый высокий ряд на всех столбцах в группе.
4	Самый нижний ряд на первом столбце, увеличенный на один для каждой строки.

Третий аргумент **format** указывает текстовое представление результата функции:

Примеры **format**

Значение	Описание
0 (по умолчанию)	Низкое значение**&' - '&**высокое значение во всех столбцах в группе (напр., 3-4).
1	Нижнее значение на всех столбцах в группе.
2	Нижнее значение на первом столбце, пустое на следующих столбцах в группе.

Порядок столбцов для элементов **mode** 4 и **format** 2 определяется порядком сортировки измерений диаграммы.

Примеры:

```
HRank( sum( Sales ))
HRank( sum( Sales ), 2 )
HRank( sum( Sales ), 0, 1 )
```

Оптимизация методом k-средних: пример из реальной жизни

Следующий пример иллюстрирует реальный случай использования, когда функции кластеризации методом k-средних и центральных точек применяются к набору данных. Функция k-средних разделяет точки данных, имеющие сходства, на кластеры. Кластеры становятся более компактными и дифференцированными по мере применения алгоритма k-средних для настраиваемого числа итераций.

Метод k-средних используется во многих областях в самых разных случаях; некоторые примеры использования кластеризации включают сегментацию клиентов, обнаружение мошенничества, прогнозирование закрытия счетов, целевое стимулирование клиентов, идентификация киберпреступников и оптимизация маршрутов доставки. Алгоритм кластеризации методом k-средних все чаще используется в тех случаях, когда предприятия пытаются вывести закономерности и оптимизировать предложения услуг.

Функции k-средних и центральных точек Qlik Sense

Qlik Sense предоставляет две функции k-средних, которые группируют точки данных в кластеры на основе сходства. См. *KMeans2D – функция диаграммы (page 800)* и *KMeansND – функция диаграммы (page 811)*. Функция **KMeans2D** принимает два измерения и хорошо подходит для визуализации результатов с помощью **точечной диаграммы**. Функция **KMeansND** принимает более двух измерений. Поскольку легко представить себе двумерный результат на стандартных диаграммах, в следующей демонстрации метод k-средних применяется к **точечной диаграмме** с использованием двух измерений. Кластеризация методом k-средних может быть визуализирована с помощью раскрашивания по выражению или по измерению, как описано в данном примере.

Функции центральных точек Qlik Sense определяют среднее арифметическое положение всех точек данных в кластере и центральную точку, или центроид, для этого кластера. Для каждой строки (или записи) диаграммы функция центральных точек отображает координату кластера, к которому была отнесена данная точка данных. См. *KMeansCentroid2D – функция диаграммы (page 822)* и *KMeansCentroidND – функция диаграммы (page 824)*.

Пример использования и обзор примера

В следующем примере рассматривается смоделированный реальный сценарий. Текстильная компания в штате Нью-Йорк, США, должны сократить затраты путем минимизации затрат на доставку. Один из способов сделать это – переместить склады поближе к своим дистрибьюторам. В компании работают 118 дистрибьюторов по всему штату Нью-Йорк. В следующей демонстрации показано, как руководитель производства может сегментировать дистрибьюторов на пять географических кластеров с помощью функции k-средних, а затем определить пять оптимальных мест расположения складов в центре этих кластеров с помощью функции центральных точек. Задача состоит в том, чтобы найти картографические координаты, которые можно использовать для определения пяти мест расположения центральных складов.

Набор данных

Набор данных основан на случайно сгенерированных названиях и адресах в штате Нью-Йорк с реальными координатами широты и долготы. Набор данных содержит следующие десять столбцов: id, first_name, last_name, telephone, address, city, state, zip, latitude, longitude. Набор данных приводится ниже как файл, который можно загрузить на локальный диск, а затем в Qlik Sense, либо как встроенные данные для редактора загрузки данных. Создаваемое приложение называется *Дистрибьюторы – k-средние и центральные точки*, а первый лист в приложении – *Анализ кластеров дистрибуции*.

Щелкните следующую ссылку для загрузки файла с образцом данных: DistributorData.csv

Набор данных Distributor: встроенная загрузка для редактора загрузки данных в Qlik Sense (page 797)

Заголовок: DistributorData

Общее количество записей: 118

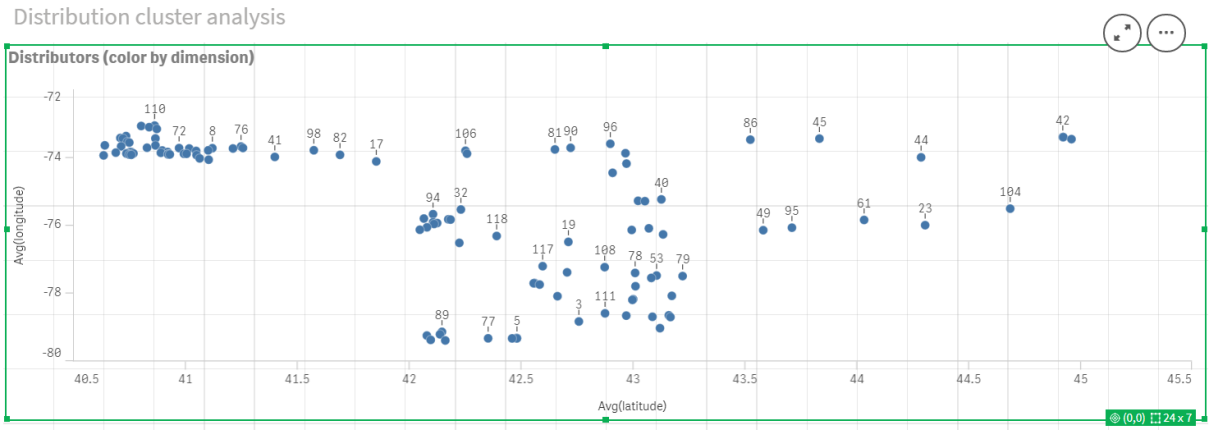
Применение функции KMeans2D

В этом примере демонстрируется настройка **точечной диаграммы** с использованием набора данных *DistributorData*, применяется функция **KMeans2D** и диаграмма раскрашивается по измерению.

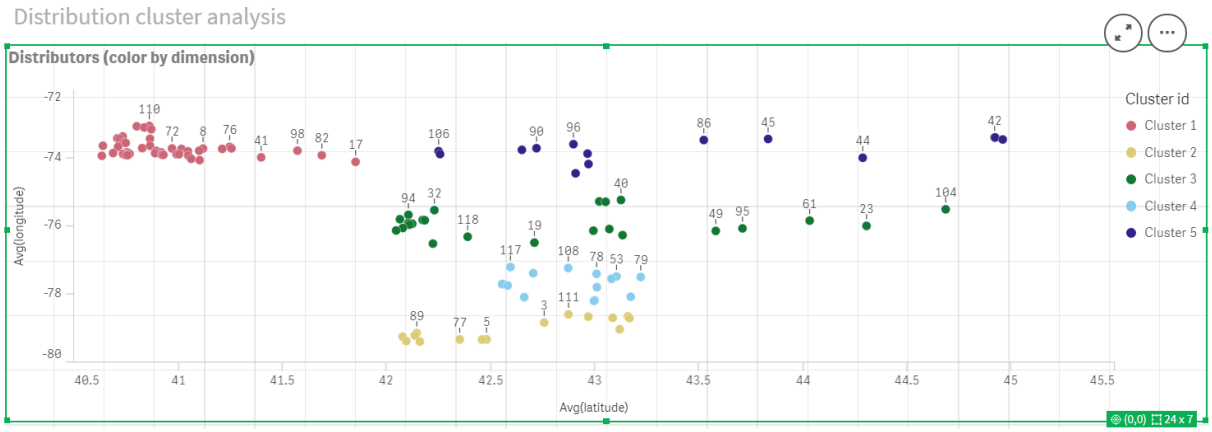
Обратите внимание, что функции k-средних Qlik Sense поддерживают автоматическую кластеризацию с помощью метода, называемого разницей глубины (DeD). Когда пользователь задает количество кластеров равным 0, оптимальное количество кластеров определяется для набора данных. Однако в данном примере переменная создается для аргумента **num_clusters** (синтаксис см. в разделе *KMeans2D – функция диаграммы (page 800)*). Поэтому желаемое количество кластеров (k=5) задается переменной.

1. Перетащите **точечную диаграмму** на лист и назовите ее *Дистрибьюторы (по измерению)*.
2. Создайте **переменную** для указания количества кластеров. **Переменная** называется *vDistClusters*. Для переменной **Определение** введите *5*.
3. Настройте **данные** для диаграммы.
    a. В разделе **Измерения** выберите поле *Идентификатор* для параметра **Пузырек**. Введите *идентификатор кластера* для **метки**.
    b. В разделе **Меры** выберите выражение *Avg([latitude])* для параметра **Ось X**.
    c. В разделе **Меры** выберите выражение *Avg([longitude])* для параметра **Ось Y**.
4. Настройка **вида**:
    a. В разделе **Цвета и легенда** выбран **Пользовательский** для параметра **Цвета**.
    b. Для раскрашивания диаграммы выбирается **По измерению**.
    c. Вводится следующее выражение: *=pick(aggr(KMeans2D(vDistClusters,only(latitude),only(longitude)),id)+1, 'Cluster 1', 'Cluster 2', 'Cluster 3', 'Cluster 4', 'Cluster 5')*
    d. Флажок **Устойчивые цвета** установлен.

Точечная диаграмма до применения раскрашивания k-средних по измерению



Точечная диаграмма после применения раскрашивания k-средних по измерению



Добавление **таблицы**: *Дистрибьюторы*

Полезно иметь такую таблицу под рукой для быстрого доступа к релевантным данным. В **точечной диаграмме** отображаются *ИД*, хотя для справки добавляется таблица с соответствующими именами дистрибьюторов.

1. Перетащите **таблицу** на лист и назовите ее *Дистрибьюторы*, при этом добавьте в нее следующие **столбцы** (измерения): *id*, *first_name* и *last_name*.

Таблица: Имена дистрибьюторов

Distributors

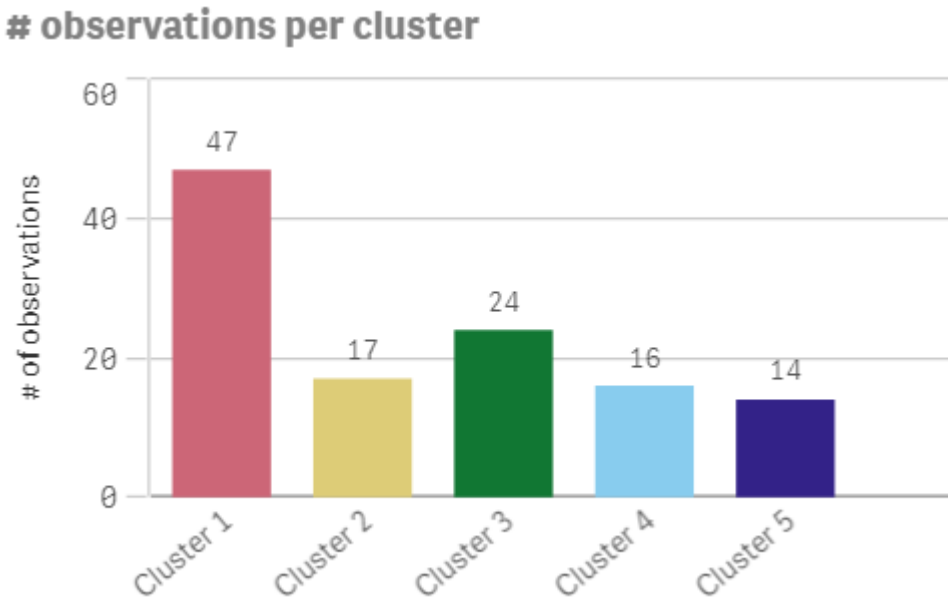
id	first_name	last_name
1	Kaiya	Snow
2	Dean	Roy
3	Eden	Paul
4	Bryanna	Higgins
5	Elisabeth	Lee
6	Skylar	Robinson
7	Cody	Bailey
8	Dario	Sims
9	Deacon	Hood

Добавление **линейчатой диаграммы**: *Кол-во наблюдений на кластер*

В сценарии распределения по складам полезно знать, сколько дистрибьюторов будет обслуживаться каждым складом. Поэтому создается **линейчатая диаграмма**, которая оценивает, сколько дистрибьюторов назначено каждому кластеру.

1. **Линейчатая диаграмма** перетаскивается на лист. Диаграмма называется: *Количество наблюдений на кластер*.
2. Конфигурация **данных** для **линейчатой диаграммы**:
    a. **Измерение** добавляется с меткой *Кластеры* (метку можно добавить после применения выражения). Вводится следующее выражение: *=pick(aggr(KMeans2D (vDistClusters,only(latitude),only(longitude)),id)+1, 'Cluster 1', 'Cluster 2', 'Cluster 3', 'Cluster 4', 'Cluster 5')*
    b. Добавляется **мера** с меткой *количество наблюдений*. Вводится следующее выражение: *=count(aggr(KMeans2D(vDistClusters,only(latitude),only(longitude)),id))*
3. Настройка **вида**:
    a. В разделе **Цвета и легенда** выбран **Пользовательский** для параметра **Цвета**.
    b. Для раскрашивания диаграммы выбирается **По измерению**.
    c. Вводится следующее выражение: *=pick(aggr(KMeans2D(vDistClusters,only(latitude),only(longitude)),id)+1, 'Cluster 1', 'Cluster 2', 'Cluster 3', 'Cluster 4', 'Cluster 5')*
    d. Флажок **Устойчивые цвета** установлен.
    e. Параметр **Показать легенду** выключен.
    f. В области **Представление** для параметра **Метки значений** выбрано значение **Авто**.
    g. В области **ось X**: **Кластеры** выбрано **Только метки**.

Линейчатая диаграмма: количество наблюдений на кластер



Применение функции Centroid2D

Вторая таблица добавлена для функции **Centroid2D**, которая будет идентифицировать координаты для потенциальных мест расположения складов. В этой таблице отображается центральное местоположение (значения центральных точек) для пяти идентифицированных групп дистрибьюторов.

1. **Таблица** перетаскивается на лист под именем *Центральные точки кластера*, в нее добавляются следующие столбцы:
    a. Добавляется **измерение** с меткой *Кластеры*. Вводится следующее выражение: *=pick (aggr(KMeans2D(vDistClusters,only(latitude),only(longitude)),id)+1,'Warehouse 1','Warehouse 2','Warehouse 3','Warehouse 4','Warehouse 5'*)
    b. Добавляется **мера** с меткой *широта (D1)*. Вводится следующее выражение: *=only(aggr (KMeansCentroid2D(vDistClusters,0,only(latitude),only(longitude)),id))*
    Примечание для параметра **coordinate_no** соответствует первому измерению(0). В этом случае измерение *latitude* соответствует оси x. Если бы мы работали с функцией **CentroidND**, имея максимум шесть измерений, эти записи параметров могли быть любым из шести значений: 0, 1, 2, 3, 4 или 5.
    c. Добавляется **мера** с меткой *longitude (D2)*. Вводится следующее выражение: *=only (aggr(KMeansCentroid2D(vDistClusters,1,only(latitude),only(longitude)),id))*
    Параметр **coordinate_no** в этом выражении соответствует второму измерению (1). Измерение *longitude* соответствует оси y.

Таблица: Расчеты центральных точек кластера

Cluster centroids

Clusters	latitude (D1)	longitude (D2)
Totals	-	-
Warehouse 1	40.945422240426	-73.719966482979
Warehouse 2	42.590538729412	-79.067889217647
Warehouse 3	42.805089516667	-75.901621883333
Warehouse 4	42.8581692625	-77.6800485875
Warehouse 5	43.436770771429	-73.734622635714

Сопоставление центральных точек

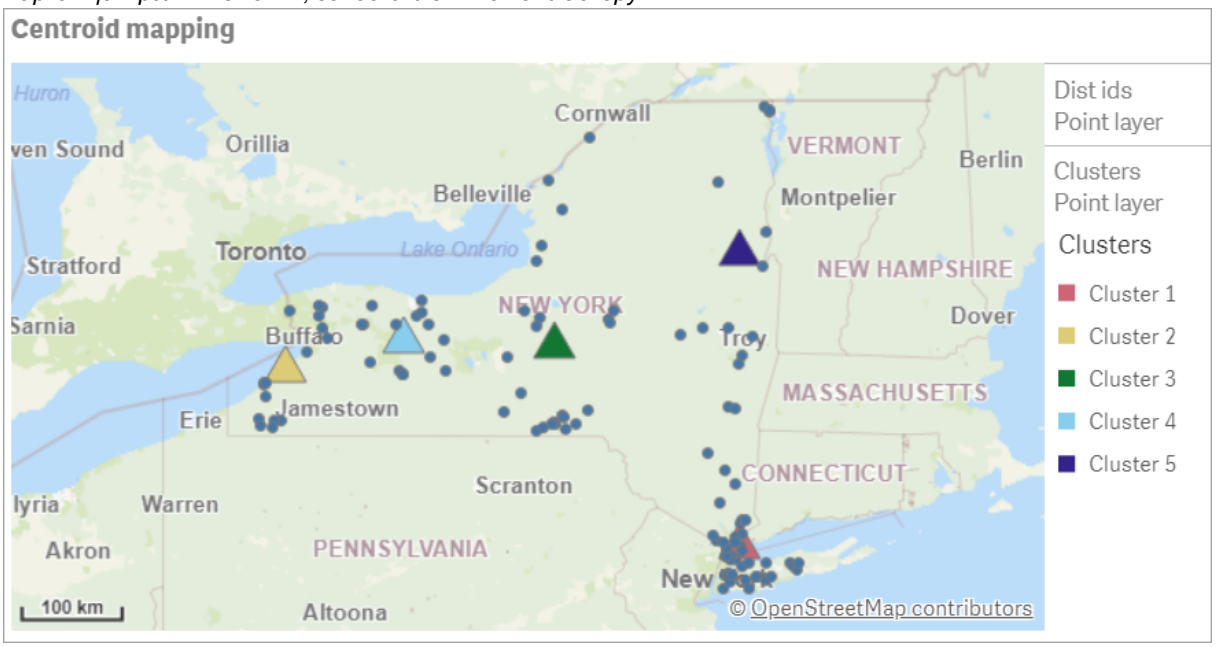
На следующем этапе выполняется сопоставление центральных точек. Разработчик приложения на свое усмотрение принимает решение о том, следует ли располагать визуализацию на отдельных листах.

1. **Карта** под именем *сопоставление центральных точек* перетаскивается на лист.
2. В разделе **Слои**. Выбрана команда **Добавить слой**, а затем **Слой точек**.
    a. Выбран **ID** *поля*, и добавлено *Dist ids* в качестве значения поля **Метка**.
    b. В разделе **Местоположение** установлен флажок **Поля широты и долготы**.
    c. В области **Широта** выбрано поле *latitude*.
    d. В области **Широта** выбрано поле *longitude*.
    e. В разделе **Размер и форма** выбрано значение **Пузырек** в поле **Форма**, и **размер** уменьшен до нужного значения с помощью ползунка.

f. В разделе **Цвета** выбран вариант **Основной**, в поле **Цвет** выбран синий, в поле **Цвет контура** (цвета можно выбрать на свое усмотрение).

3. В разделе **Слои** добавлен второй объект **Слой точек**: для этого сначала выбрана команда **Добавить слой**, а затем вариант **Слой точек**.
   a. Вводится следующее выражение: *=aggr(KMeans2D(vDistClusters,only(latitude),only(longitude)),id)*
   b. Добавлена **метка** *Кластеры*.
   c. В разделе **Местоположение** установлен флажок **Поля широты и долготы**.
   d. Для параметра **Широта**, который располагается на оси x, добавляется следующее выражение: *=aggr(KMeansCentroid2D(vDistClusters,0,only(latitude),only(longitude)),id)*
   e. Для параметра **Долгота**, который располагается на оси y, добавляется следующее выражение: *=aggr(KMeansCentroid2D(vDistClusters,1,only(latitude),only(longitude)),id)*
   f. В разделе **Размер и форма** выбрано значение **Треугольник** в поле **Форма**, и **размер** уменьшен до нужного значения с помощью ползунка.
   g. В разделе **Цвета и легенда** выбран **Пользовательский** для параметра **Цвета**.
   h. Для раскрашивания диаграммы выбирается **По измерению**. Вводится следующее выражение: *=pick(aggr(KMeans2D(vDistClusters,only(latitude),only(longitude)),id)+1,'Cluster 1','Cluster 2','Cluster 3','Cluster 4','Cluster 5')*
   i. Добавляется измерение с меткой *Кластеры*.
4. В области **Параметры карты** выбрано значение **Адаптивная** для параметра **Проекция**. Значение **Метрические** выбрано для параметра **Единицы измерения**.

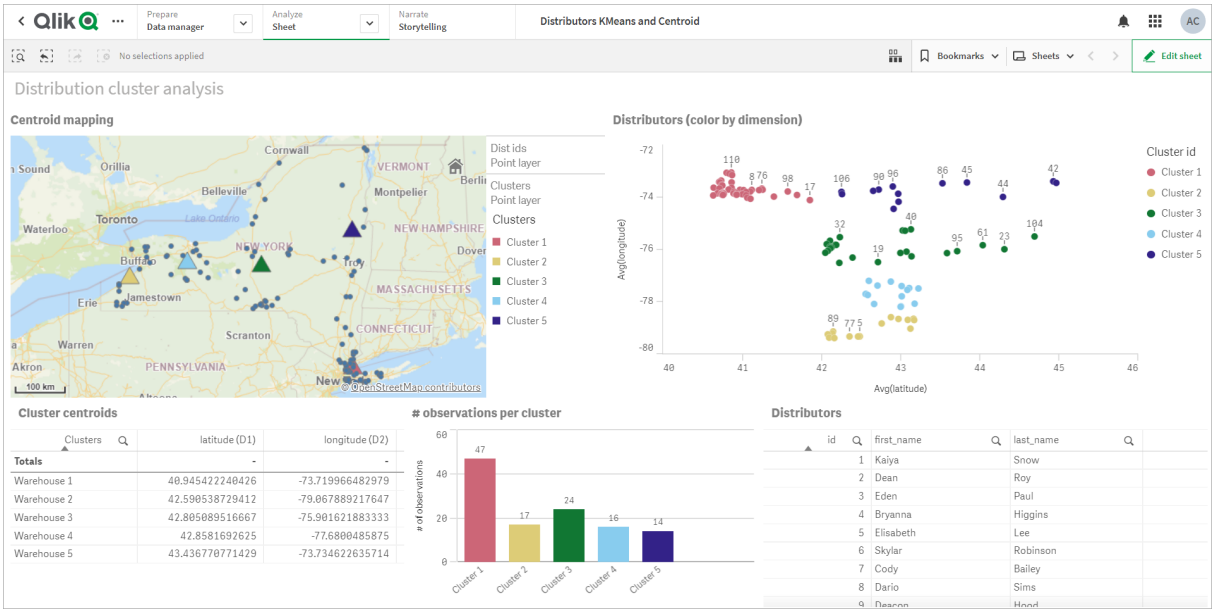
Карта: Центральные точки, сопоставленные по кластеру



Заключение

Благодаря использованию функции KMeans для данного примера из реальной жизни дистрибьюторы разделены на подобные группы или кластеры на основе сходства, в данном случае по близости друг к другу. К полученным кластерам применена функция Centroid для идентификации пяти координат сопоставления. Эти координаты определяют начальное центрально местоположение, в котором следует строить или размещать склады. Функция centroid применяется к диаграмме **Карта**, чтобы пользователи приложения могли визуализировать, где находятся центральные точки относительно окружающих точек данных. Полученные координаты представляют потенциальные места расположения складов, которые позволят сократить затраты на доставку дистрибьюторам в штате Нью-Йорк.

Приложение: Пример анализа k-средних и центральных точек



Набор данных Distributor: встроенная загрузка для редактора загрузки данных в Qlik Sense

```
DistributorData: Load * Inline [ id,first_name,last_
name,telephone,address,city,state,zip,latitude,longitude 1,Kaiya,Snow,(716) 201-1212,6231
Tonawanda Creek Rd #APT 308,Lockport,NY,14094,43.08926,-78.69313 2,Dean,Roy,(716) 201-
1588,6884 E High St,Lockport,NY,14094,43.16245,-78.65036 3,Eden,Paul,(716) 202-4596,4647
Southwestern Blvd #APT 350,Hamburg,NY,14075,42.76003,-78.83194 4,Bryanna,Higgins,(716) 203-
7041,418 Park Ave,Dunkirk,NY,14048,42.48279,-79.33088 5,Elisabeth,Lee,(716) 203-7043,36 E
Courtney St,Dunkirk,NY,14048,42.48299,-79.31928 6,Skylar,Robinson,(716) 203-7166,26 Greco
Ln,Dunkirk,NY,14048,42.4612095,-79.3317925 7,Cody,Bailey,(716) 203-7201,114 Lincoln
Ave,Dunkirk,NY,14048,42.4801269,-79.322232 8,Dario,Sims,(408) 927-1606,N Castle
Dr,Armonk,NY,10504,41.11979,-73.714864 9,Deacon,Hood,(410) 244-6221,4856 44th
St,Woodside,NY,11377,40.748372,-73.905445 10,Zackery,Levy,(410) 363-8874,61 Executive
Blvd,Farmingdale,NY,11735,40.7197457,-73.430239 11,Rey,Hawkins,(412) 344-8687,4585 Shimerville
Rd,Clarence,NY,14031,42.972075,-78.6592452 12,Phillip,Howard,(413) 269-4049,464 Main St
#101,Port Washington,NY,11050,40.8273756,-73.7009971 13,Shirley,Tyler,(434) 985-8943,114 Glann
Rd,Apalachin,NY,13732,42.0482515,-76.1229725 14,Aniyah,Jarvis,(440) 244-1808,87 N Middletown
```

Rd,Pearl River,NY,10965,41.0629,-74.0159 15,Alayna,Woodard,(478) 335-3704,70 W Red Oak Ln,West Harrison,NY,10604,41.0162722,-73.7234926 16,Jermaine,Lambert,(508) 561-9836,24 Kellogg Rd,New Hartford,NY,13413,43.0555739,-75.2793197 17,Harper,Gibbs,(239) 466-0238,Po Box 33,Cottekill,NY,12419,41.853392,-74.106082 18,Osvaldo,Graham,(252) 246-0816,6878 Sand Hill Rd,East Syracuse,NY,13057,43.073215,-76.081448 19,Roberto,Wade,(270) 469-1211,3936 Holley Rd,Moravia,NY,13118,42.713044,-76.481227 20,Kate,Mcguire,(270) 788-3080,6451 State 64 Rte #3,Naples,NY,14512,42.707366,-77.380489 21,Dale,Andersen,(281) 480-5690,205 W Service Rd,Champlain,NY,12919,44.9645392,-73.4470831 22,Lorelai,Burch,(302) 644-2133,1 Brewster St,Glen Cove,NY,11542,40.865177,-73.633019 23,Amiyah,Flowers,(303) 223-0055,46600 Us Interstate 81 Rte,Alexandria Bay,NY,13607,44.309626,-75.988365 24,Mckinley,Clements,(303) 918-3230,200 Summit Lake Dr,Valhalla,NY,10595,41.101145,-73.778298 25,Marc,Gibson,(607) 203-1233,25 Robinson St,Binghamton,NY,13901,42.107416,-75.901614 26,Kali,Norman,(607) 203-1400,1 Ely Park Blvd #APT 15,Binghamton,NY,13905,42.125866,-75.925026 27,Laci,Cain,(607) 203-1437,16 Zimmer Road,Kirkwood,NY,13795,42.066516,-75.792627 28,Mohammad,Perez,(607) 203-1652,71 Endicott Ave #APT 12,Johnson City,NY,13790,42.111894,-75.952187 29,Izabelle,Pham,(607) 204-0392,434 State 369 Rte,Port Crane,NY,13833,42.185838,-75.823074 30,Kiley,Mays,(607) 204-0870,244 Ballyhack Rd #14,Port Crane,NY,13833,42.175612,-75.814917 31,Peter,Trevino,(607) 205-1374,125 Melbourne St.,Vestal,NY,13850,42.080254,-76.051124 32,Ani,Francis,(607) 208-4067,48 Caswell St,Afton,NY,13730,42.232065,-75.525674 33,Jared,Sheppard,(716) 386-3002,4709 430th Rte,Bemus Point,NY,14712,42.162175,-79.39176 34,Dulce,Atkinson,(914) 576-2266,501 Pelham Rd,New Rochelle,NY,10805,40.895449,-73.782602 35,Jayla,Beasley,(716) 526-1054,5010 474th Rte,Ashville,NY,14710,42.096859,-79.375561 36,Dane,Donovan,(718) 545-3732,5014 31st Ave,Woodside,NY,11377,40.756967,-73.909506 37,Brendon,Clay,(585) 322-7780,133 Cummings Ave,Gainesville,NY,14066,42.664309,-78.085651 38,Asia,Nunez,(718) 426-1472,2407 Gilmore ,East Elmhurst,NY,11369,40.766662,-73.869185 39,Dawson,Odonnell,(718) 342-2179,5019 H Ave,Brooklyn,NY,11234,40.633245,-73.927591 40,Kyle,Collins,(315) 733-7078,502 Rockhaven Rd,Utica,NY,13502,43.129184,-75.226726 41,Eliza,Hardin,(315) 331-8072,502 Sladen Place,West Point,NY,10996,41.3993,-73.973003 42,Kasen,Klein,(518) 298-4581,2407 Lake Shore Rd,Chazy,NY,12921,44.925561,-73.387373 43,Reuben,Bradford,(518) 298-4581,33 Lake Flats Dr,Champlain,NY,12919,44.928092,-73.387884 44,Henry,Grimes,(518) 523-3990,2407 Main St,Lake Placid,NY,12946,44.291487,-73.98474 45,Kyan,Livingston,(518) 585-7364,241 Alexandria Ave,Ticonderoga,NY,12883,43.836553,-73.43155 46,Kaitlyn,Short,(516) 678-3189,241 Chance Dr,Oceanside,NY,11572,40.638534,-73.63079 47,Damaris,Jacobs,(914) 664-5331,241 Claremont Ave,Mount Vernon,NY,10552,40.919852,-73.827848 48,Alivia,Schroeder,(315) 469-4473,241 Lafayette Rd,Syracuse,NY,13205,42.996446,-76.12957 49,Bridget,Strong,(315) 298-4355,241 Maltby Rd,Pulaski,NY,13142,43.584966,-76.136317 50,Francis,Lee,(585) 201-7021,166 Ross St,Batavia,NY,14020,43.0031502,-78.17487 51,Makaila,Phelps,(585) 201-7422,58 S Main St,Batavia,NY,14020,42.99941,-78.1939285 52,Jazlynn,Stephens,(585) 203-1087,1 Sinclair Dr,Pittsford,NY,14534,43.084157,-77.545452 53,Ryann,Randolph,(585) 203-1519,331 Eaglehead Rd,East Rochester,NY,14445,43.10785,-77.475552 54,Rosa,Baker,(585) 204-4011,42 Ossian St,Dansville,NY,14437,42.560761,-77.70088 55,Marcel,Barry,(585) 204-4013,42 Jefferson St,Dansville,NY,14437,42.557735,-77.702983 56,Dennis,Schmitt,(585) 204-4061,750 Dansville Mount Morris Rd,Dansville,NY,14437,42.584458,-77.741648 57,Cassandra,Kim,(585) 204-4138,3 Perine Ave APT1,Dansville,NY,14437,42.562865,-77.69661 58,Kolton,Jacobson,(585) 206-5047,4925 Upper Holly Rd,Holley,NY,14470,43.175957,-78.074465 59,Nathanael,Donovan,(718) 393-3501,9604 57th Ave,Corona,NY,11373,40.736077,-73.864858 60,Robert,Frazier,(718) 271-3067,300 56th Ave,Corona,NY,11373,40.735304,-73.873997 61,Jessie,Mora,(315) 405-8991,9607 Forsyth Loop,Watertown,NY,13603,44.036466,-75.833437 62,Martha,Rollins,(347) 242-2642,22 Main St,Corona,NY,11373,40.757727,-73.829331 63,Emely,Townsend,(718) 699-0751,60 Sanford Ave,Corona,NY,11373,40.755466,-73.831029 64,Kylie,Cooley,(347) 561-7149,9608 95th Ave,Ozone Park,NY,11416,40.687564,-73.845715 65,Wendy,Cameron,(585) 571-4185,9608 Union St,Scottsville,NY,14546,43.013327,-77.7907839 66,Kayley,Peterson,(718) 654-5027,961 E 230th St,Bronx,NY,10466,40.889275,-73.850555 67,Camden,Ochoa,(718) 760-8699,59 Vark St,Yonkers,NY,10701,40.929322,-73.89957 68,Priscilla,Castillo,(910) 326-7233,9359 Elm St,Chadwicks,NY,13319,43.024902,-75.26886 69,Dana,Schultz,(913) 322-4580,99 Washington Ave,Hastings on Hudson,NY,10706,40.99265,-73.879748 70,Blaze,Medina,(914) 207-0015,60 Elliott

```
Ave,Yonkers,NY,10705,40.921498,-73.896682 71,Finnegan,Tucker,(914) 207-0015,90 Hillside
Drive,Yonkers,NY,10705,40.922514,-73.892911 72,Pranav,Palmer,(914) 214-8376,5 Bruce
Ave,Harrison,NY,10528,40.970916,-73.711493 73,Kolten,Wong,(914) 218-8268,70 Barker St,Mount
Kisco,NY,10549,41.211993,-73.723202 74,Jasiah,Vazquez,(914) 231-5199,30 Broadway,Dobbs
Ferry,NY,10522,41.004629,-73.879825 75,Lamar,Pierce,(914) 232-0380,68 Ridge
Rd,Katonah,NY,10536,41.256662,-73.707964 76,Carla,Coffey,(914) 232-0469,197 Beaver Dam
Rd,Katonah,NY,10536,41.247934,-73.664363 77,Brooklynn,Harmon,(716) 595-3227,8084 Glasgow
Rd,Cassadega,NY,14718,42.353861,-79.329558 78,Raquel,Hodges,(585) 398-8125,809 County Road
,Victor,NY,14564,43.011745,-77.398806 79,Jerimiah,Gardner,(585) 787-9127,809 Houston
Rd,Webster,NY,14580,43.224204,-77.491353 80,Clarence,Hammond,(720) 746-1619,809 Pierpont
Ave,Piermont,NY,10968,41.0491181,-73.918622 81,Rhys,Gill,(518) 427-7887,81 Columbia
St,Albany,NY,12210,42.652824,-73.752096 82,Edith,Parrish,(845) 452-7621,81 Glenwood
Ave,Poughkeepsie,NY,12603,41.691058,-73.910829 83,Kobe,Mcintosh,(845) 371-1101,81 Heitman
Dr,Spring Valley,NY,10977,41.103227,-74.054396 84,Ayden,Waters,(516) 796-2722,81 Kingfisher
Rd,Levittown,NY,11756,40.738939,-73.52826 85,Francis,Rogers,(631) 427-7728,81 Knollwood
Ave,Huntington,NY,11743,40.864905,-73.426107 86,Jaden,Landry,(716) 496-4038,12839 39th
Rte,Chaffee,NY,14030,43.527396,-73.462786 87,Giancarlo,Campos,(518) 885-5717,1284 Saratoga
Rd,Ballston Spa,NY,12020,42.968594,-73.862847 88,Eduardo,Contreras,(716) 285-8987,1285
Saunders Sett Rd,Niagara Falls,NY,14305,43.122963,-79.029274 89,Gabriela,Davidson,(716) 267-
3195,1286 Mee Rd,Falconer,NY,14733,42.147339,-79.137976 90,Evangeline,Case,(518) 272-9435,1287
2nd Ave,Watervliet,NY,12189,42.723132,-73.703818 91,Tyrone,Ellison,(518) 843-4691,1287 Midline
Rd,Amsterdam,NY,12010,42.9730876,-74.1700608 92,Bryce,Bass,(518) 943-9549,1288 Leeds Athens
Rd,Athens,NY,12015,42.259381,-73.876897 93,Londyn,Butler,(518) 922-7095,129 Argersinger
Rd,Fultonville,NY,12072,42.910969,-74.441917 94,Graham,Becker,(607) 655-1318,129 Baker
Rd,Windsor,NY,13865,42.107271,-75.66408 95,Rolando,Fitzgerald,(315) 465-4166,17164 County 90
Rte,Mannsville,NY,13661,43.713443,-76.06232 96,Grant,Hoover,(518) 692-8363,1718 County 113
Rte,Schaghticote,NY,12154,42.900648,-73.585036 97,Mark,Goodwin,(631) 584-6761,172 Cambon
Ave,Saint James,NY,11780,40.871152,-73.146032 98,Deacon,Cantu,(845) 221-7940,172 Carpenter
Rd,Hopewell Junction,NY,12533,41.57388,-73.77609 99,Tristian,Walsh,(516) 997-4750,172 E Cabot
Ln,Westbury,NY,11590,40.7480397,-73.54819 100,Abram,Alexander,(631) 588-3817,172 Lorenzo
Cir,Ronkonkoma,NY,11779,40.837123,-73.09367 101,Lesly,Bush,(516) 489-3791,172 Nassau
Blvd,Garden City,NY,11530,40.71147,-73.660753 102,Pamela,Espinoza,(716) 201-1520,172 Niagara
St ,Lockport,NY,14094,43.169871,-78.70093 103,Bryanna,Newton,(914) 328-4332,172 Warren
Ave,White Plains,NY,10603,41.047207,-73.79572 104,Marcelo,Schmitt,(315) 393-4432,319 Mansion
Ave,Ogdensburg,NY,13669,44.690246,-75.49992 105,Layton,Valenzuela,(631) 676-2113,319
Singingwood Dr,Holbrook,NY,11741,40.801391,-73.058993 106,Roderick,Rocha,(518) 671-6037,319
Warren St,Hudson,NY,12534,42.252527,-73.790629 107,Camryn,Terrell,(315) 635-1680,3192 Olive
Dr,Baldinsville,NY,13027,43.136843,-76.260303 108,Summer,Callahan,(585) 394-4195,3192 Smith
Road,Canandaigua,NY,14424,42.875457,-77.228039 109,Pierre,Novak,(716) 665-2524,3194 Falconer
Kimball Stand Rd,Falconer,NY,14733,42.138439,-79.211091 110,Kennedi,Fry,(315) 543-2301,32
College Rd,Selden,NY,11784,40.861624,-73.04757 111,Wyatt,Pruitt,(716) 681-4042,277 Ransom
Rd,Lancaster ,NY,14086,42.87702,-78.591302 112,Lilly,Jensen,(631) 841-0859,2772 Schliegel
Blvd,Amityville,NY,11701,40.708021,-73.413015 113,Tristin,Hardin,(631) 920-0927,278 Fulton
Street,West Babylon,NY,11704,40.733578,-73.357321 114,Tanya,Stafford,(716) 484-0771,278
Sampson St,Jamestown,NY,14701,42.0797,-79.247805 115,Paris,Cordova,(607) 589-4857,278 Washburn
Rd,Spencer,NY,14883,42.225046,-76.510257 116,Alfonso,Morse,(718) 359-5582,200 Colden
St,Flushing,NY,11355,40.750403,-73.822752 117,Maurice,Hooper,(315) 595-6694,4435 Italy Hill
Rd,Branchport,NY,14418,42.597957,-77.199267 118,Iris,Wolf,(607) 539-7288,444 Harford
Rd,Brooktondale,NY,14817,42.392164,-76.30756 ];
```

KMeans2D — функция диаграммы

KMeans2D() вычисляет строки диаграммы путем применения кластеризации методом k-средних; для каждой строки диаграммы отображается идентификатор кластера, которому была назначена эта точка диаграммы. Столбцы, используемые алгоритмом кластеризации, определены соответственно параметрами coordinate_1 и coordinate_2. Они оба являются агрегированиями. Количество созданных кластеров определяется параметром num_clusters. Данные могут быть при необходимости нормализованы с помощью параметра нормы.

KMeans2D возвращает одно значение на точку диаграммы. Возвращенное значение — двойное и является целочисленным значением, соответствующим кластеру, которому была назначена каждая точка диаграммы.

Синтаксис:

```
KMeans2D(num_clusters, coordinate_1, coordinate_2 [, norm])
```

Возвращаемые типы данных: двойное значение

Аргументы:

Аргументы

Аргумент	Описание
num_clusters	Целое число, которое указывает количество кластеров.
coordinate_1	Агрегирование, вычисляющее первую координату, обычно ось X точечной диаграммы, которая может быть сделана из диаграммы. Дополнительный параметр coordinate_2 вычисляет вторую координату.
norm	Дополнительный метод нормализации применяется к наборам данных перед кластеризацией методом k-средних. Возможные значения: 0 или 'нет' при отсутствии нормализации 1 или ‘zscore’ для нормализации с помощью z-оценки 2 или ‘minmax’ для нормализации с помощью мин./макс. Если параметры не предоставлены или предоставленный параметр неправильный, нормализация не применяется. Z-оценка нормализует данные на основе среднего и стандартного отклонения признака. Z-оценка не гарантирует, что у каждого признака будет одинаковый масштаб, но при выбросах этот подход лучше, чем мин./макс. Нормализация с помощью мин./макс. гарантирует, что признаки имеют одинаковый масштаб; для этого берутся минимальное и максимальное значения каждого признака и каждая точка данных вычисляется заново.

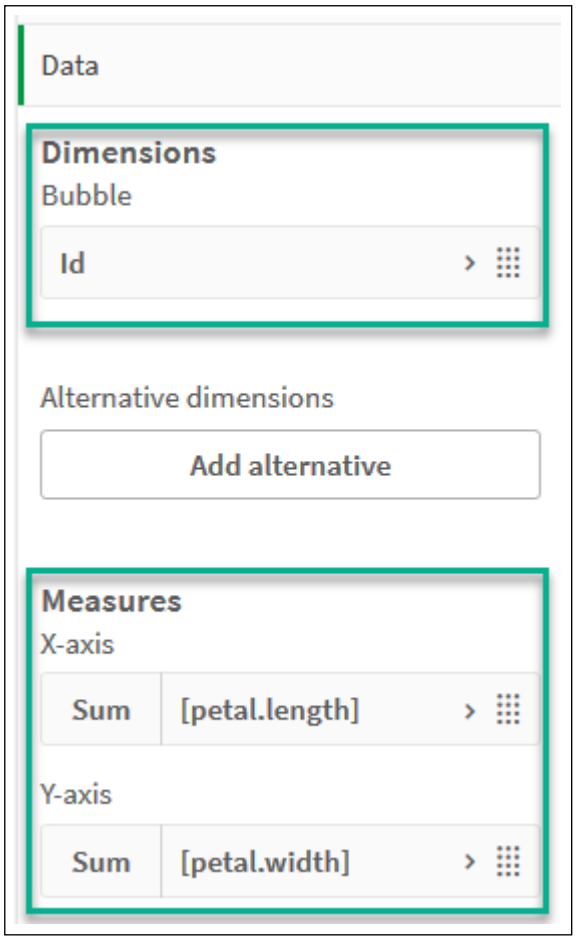
Пример: Выражение диаграммы
В этом примере создается точечная диаграмма с помощью набора данных *Iris*, и затем с помощью KMeans данные раскрашиваются по выражению.

Также мы создаем переменную для аргумента *num_clusters* и затем используем поле ввода переменной, чтобы изменить количество кластеров.

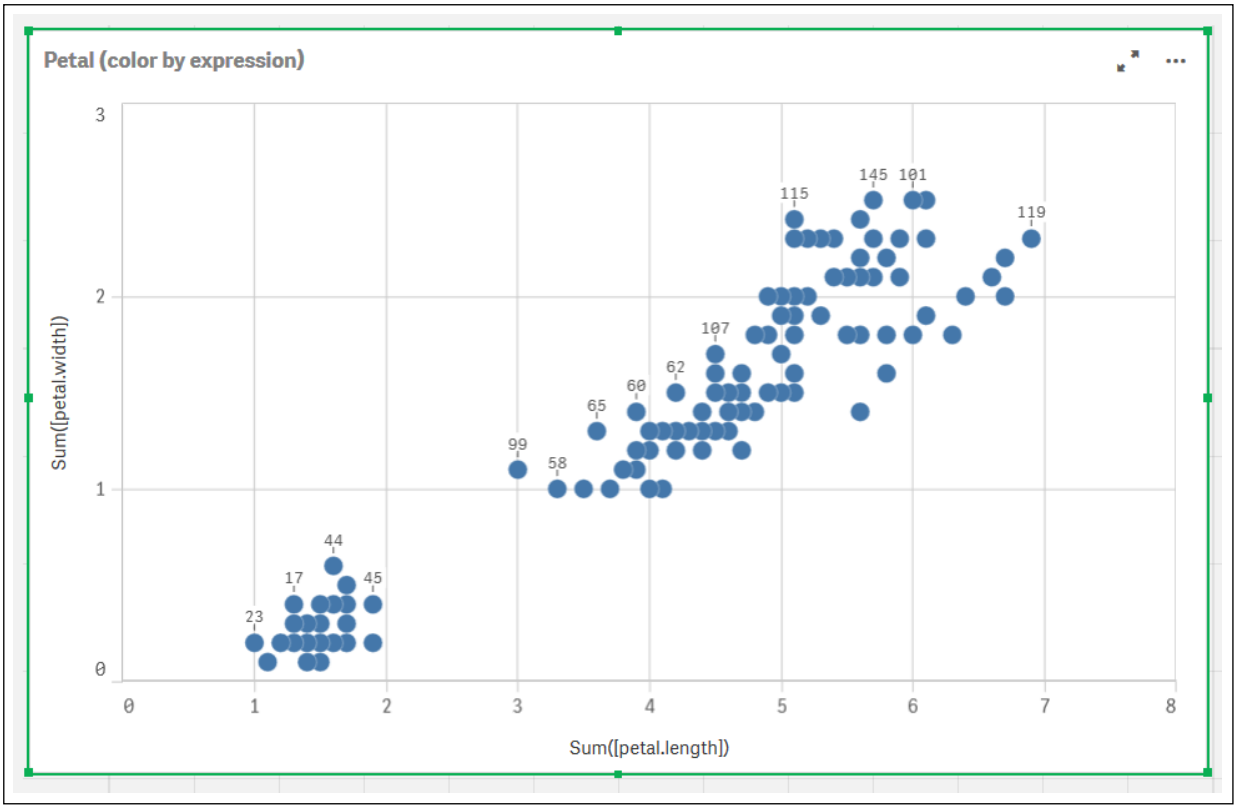
Набор данных *Iris* общедоступен в различных форматах. Данные предоставлены в виде встроенной таблицы для загрузки с помощью редактора загрузки данных в программе Qlik Sense. Обратите внимание, что к таблице данных для этого примера добавлен столбец *Идентификатор*.

После загрузки данных в Qlik Sense нужно выполнить следующие действия.

1. Перетащите **точечную диаграмму** на новый лист. Назовите диаграмму *Лепесток (раскрашивание по выражению)*.
2. Создайте переменную для указания количества кластеров. Для переменной **Имя** введите *KmeansPetalClusters*. Для переменной **Определение** введите *=2*.
3. Настройте **данные** для диаграммы.
    i. В разделе **Измерения** выберите *Идентификатор* поля для параметра **Пузырек**. Введите идентификатор кластера для метки.
    ii. В разделе **Меры** выберите *Sum([petal.length])* для выражения для параметра **Ось X**.
    iii. В разделе **Меры** выберите *Sum([petal.width])* для выражения для параметра **Ось Y**.
    Параметры данных для диаграммы Лепесток (раскрашивание по выражению)

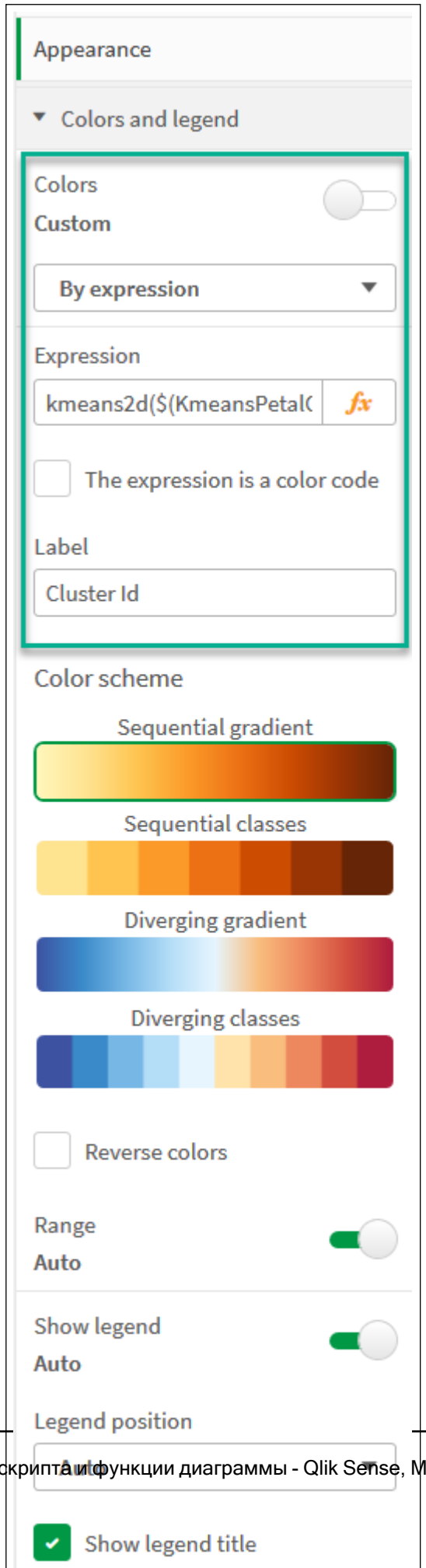


Точки диаграммы нанесены на диаграмму.
Точки диаграммы на диаграмме Лепесток (раскрашивание по выражению)

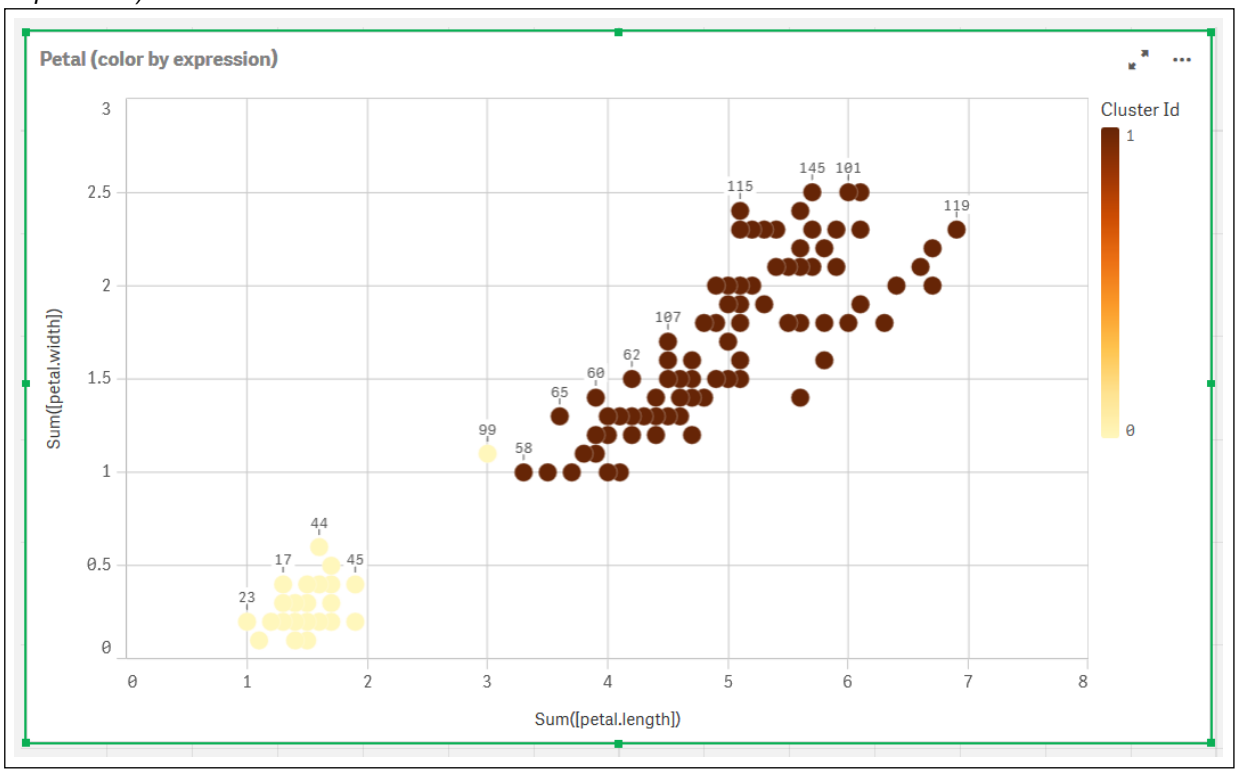


4. Настройте **Вид** диаграммы:
    i. В разделе **Цвета и легенда** выберите **Пользовательский** для параметра **Цвета**.
    ii. Выберите раскрашивание диаграммы **По выражению**.
    iii. Введите следующее для **выражения**: *kmeans2d($(KmeansPetalClusters), Sum ([petal.length]), Sum([petal.width]))*
    Обратите внимание, что *KmeansPetalClusters* – это переменная, для которой установлено *2*.
    Альтернативно введите следующее: *kmeans2d(2, Sum([petal.length]), Sum ([petal.width]))*
    iv. Снимите флажок **Выражение является цветовым кодом**.
    v. Введите следующее для **метки**: *Идентификатор кластера*

Параметры вида для диаграммы Лепесток (раскрашивание по выражению)

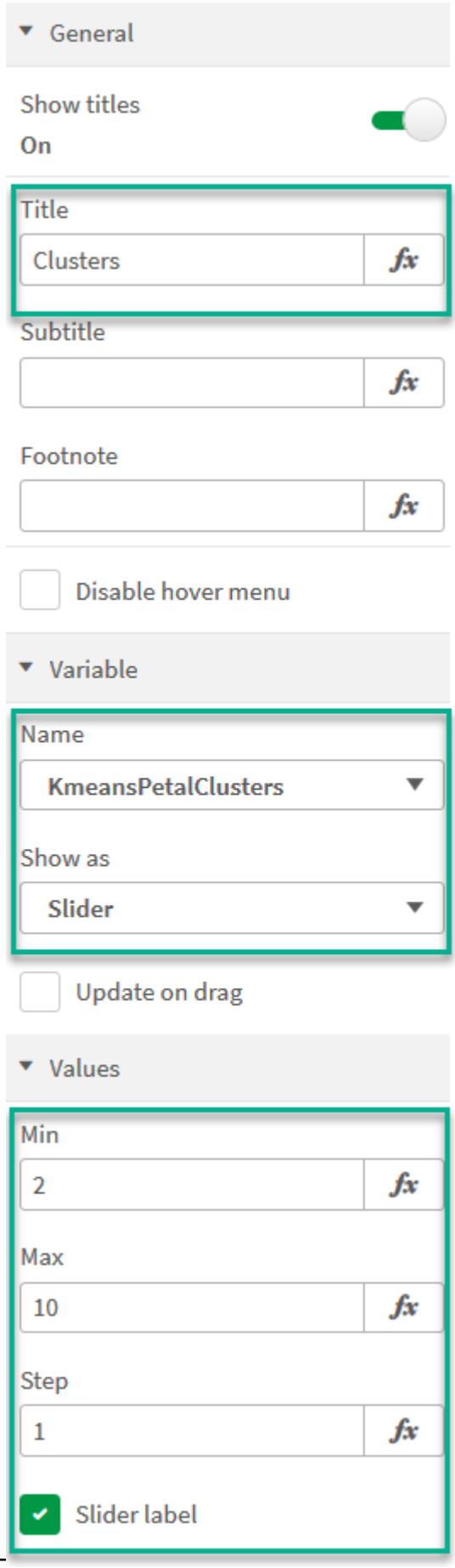

Appearance
Colors and legend
Colors
Custom
By expression
Expression
kmeans2d($(KmeansPetal(
The expression is a color code
Label
Cluster Id
Color scheme
Sequential gradient
Sequential classes
Diverging gradient
Diverging classes
Reverse colors
Range
Auto
Show legend
Auto
Legend position
Show legend title

Два кластера на диаграмме раскрашены по выражению KMeans.
Кластеры, раскрашенные по выражению на диаграммы Лепесток (раскрашивание по выражению)



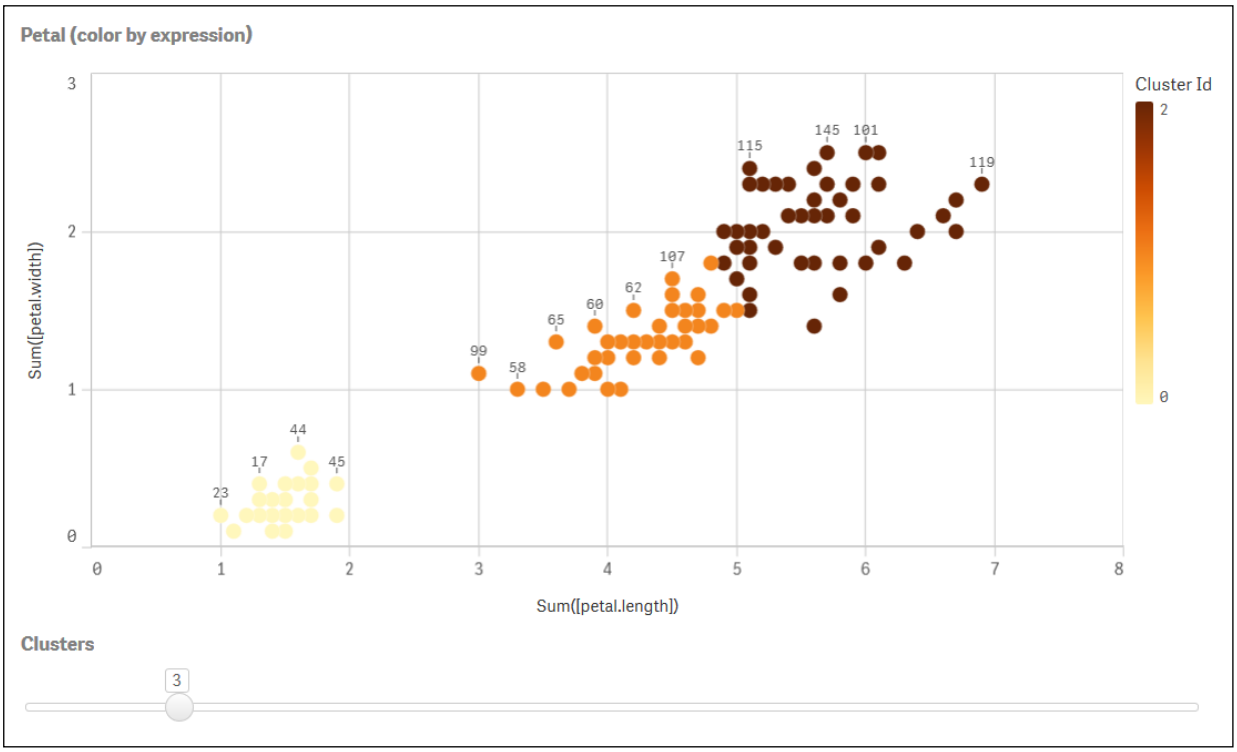
5. Добавьте поле **Ввод переменной** для количества кластеров.
    i. В разделе **Пользовательские объекты** на панели **Ресурсы** выберите **Qlik Dashboard Bundle**. Если бы к пакету информационной панели не было доступа, количество кластеров можно бы было изменить с помощью созданной переменной или непосредственно с помощью целого числа в выражении.
    ii. Перетащите поле **Ввод переменной** на лист.
    iii. В разделе **Вид** щелкните **Общее**.
    iv. Введите следующее для параметра **Заголовок**: *Кластеры*
    v. Щелкните **Переменная**.
    vi. Выберите следующую переменную для параметра **Имя**: *KmeansPetalClusters*.
    vii. Выберите **Ползунок** для параметра **Показать в виде**.
    viii. Выберите **Значения** и настройте параметры как требуется,

Вид поля ввода переменной Кластеры


General
Show titles
On
Title
Clusters
Subtitle
Footnote
Disable hover menu
Variable
Name
KmeansPetalClusters
Show as
Slider
Update on drag
Values
Min
2
Max
10
Step
1
Slider label

По завершении редактирования можно изменить количество кластеров с помощью ползунка в поле ввода переменной *Кластеры*.

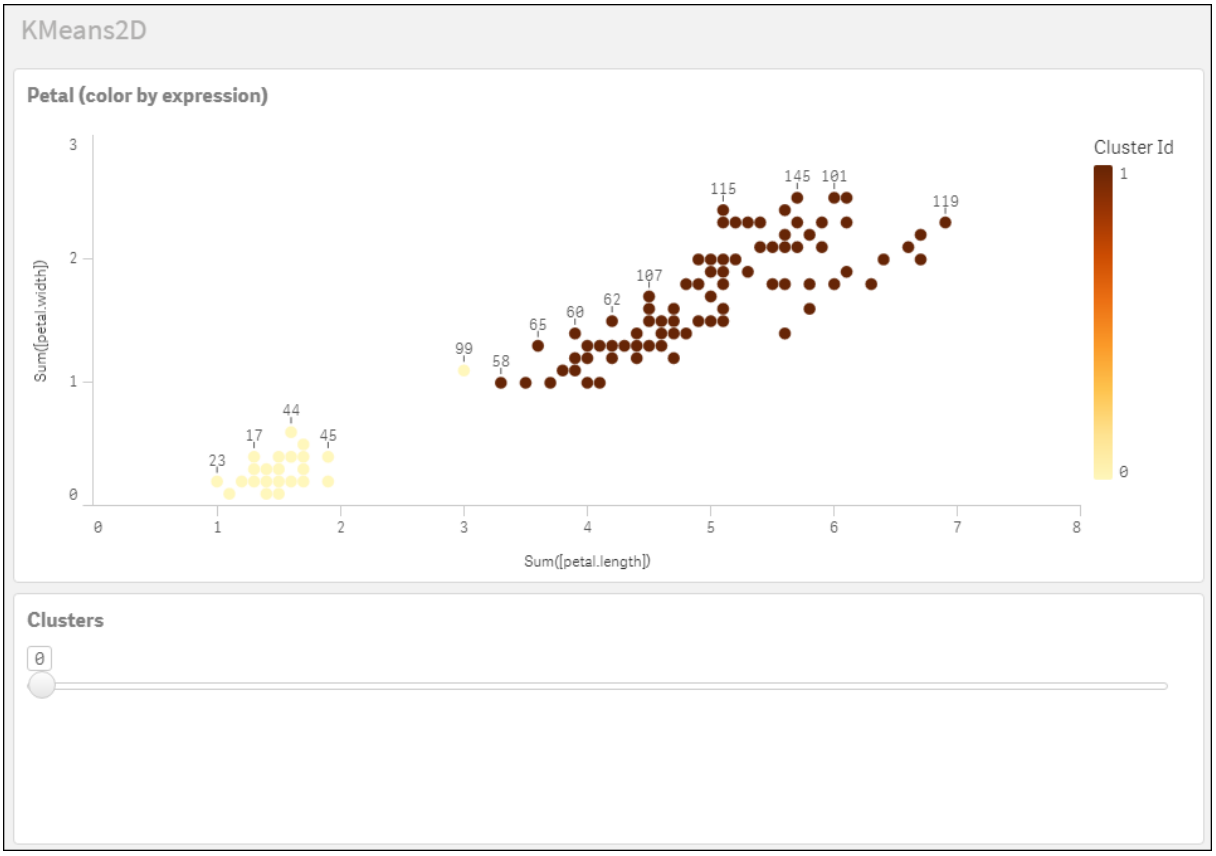
Кластеры, раскрашенные по выражению на диаграммы Лепесток (раскрашивание по выражению)



Автоматическая кластеризация

Функции **метода k-средних** поддерживают автоматическую кластеризацию с помощью метода, называемого разницей глубины (DeD). Когда пользователь задает количество кластеров равным 0, оптимальное количество кластеров определяется для набора данных. Обратите внимание, что хотя целое число для количества кластеров (*k*) явно не возвращается, оно вычисляется в алгоритме k-средних. Например, если 0 указан в функции для значения *KmeansPetalClusters* или установлен через поле ввода переменной, назначения кластеров автоматически вычисляются для набора данных на основе оптимального количества кластеров.

Метод разницы глубины k-средних определяет оптимальное количество кластеров, когда (k) установлен на 0



Набор данных Iris: встроенная загрузка для редактора загрузки данных в Qlik Sense

```
IrisData: Load * Inline [ sepal.length, sepal.width, petal.length, petal.width, variety, id
5.1, 3.5, 1.4, 0.2, Setosa, 1 4.9, 3, 1.4, 0.2, Setosa, 2 4.7, 3.2, 1.3, 0.2, Setosa, 3 4.6,
3.1, 1.5, 0.2, Setosa, 4 5, 3.6, 1.4, 0.2, Setosa, 5 5.4, 3.9, 1.7, 0.4, Setosa, 6 4.6, 3.4,
1.4, 0.3, Setosa, 7 5, 3.4, 1.5, 0.2, Setosa, 8 4.4, 2.9, 1.4, 0.2, Setosa, 9 4.9, 3.1, 1.5,
0.1, Setosa, 10 5.4, 3.7, 1.5, 0.2, Setosa, 11 4.8, 3.4, 1.6, 0.2, Setosa, 12 4.8, 3, 1.4,
0.1, Setosa, 13 4.3, 3, 1.1, 0.1, Setosa, 14 5.8, 4, 1.2, 0.2, Setosa, 15 5.7, 4.4, 1.5, 0.4,
Setosa, 16 5.4, 3.9, 1.3, 0.4, Setosa, 17 5.1, 3.5, 1.4, 0.3, Setosa, 18 5.7, 3.8, 1.7, 0.3,
Setosa, 19 5.1, 3.8, 1.5, 0.3, Setosa, 20 5.4, 3.4, 1.7, 0.2, Setosa, 21 5.1, 3.7, 1.5, 0.4,
Setosa, 22 4.6, 3.6, 1, 0.2, Setosa, 23 5.1, 3.3, 1.7, 0.5, Setosa, 24 4.8, 3.4, 1.9, 0.2,
Setosa, 25 5, 3, 1.6, 0.2, Setosa, 26 5, 3.4, 1.6, 0.4, Setosa, 27 5.2, 3.5, 1.5, 0.2, Setosa,
28 5.2, 3.4, 1.4, 0.2, Setosa, 29 4.7, 3.2, 1.6, 0.2, Setosa, 30 4.8, 3.1, 1.6, 0.2, Setosa,
31 5.4, 3.4, 1.5, 0.4, Setosa, 32 5.2, 4.1, 1.5, 0.1, Setosa, 33 5.5, 4.2, 1.4, 0.2, Setosa,
34 4.9, 3.1, 1.5, 0.1, Setosa, 35 5, 3.2, 1.2, 0.2, Setosa, 36 5.5, 3.5, 1.3, 0.2, Setosa, 37
4.9, 3.1, 1.5, 0.1, Setosa, 38 4.4, 3, 1.3, 0.2, Setosa, 39 5.1, 3.4, 1.5, 0.2, Setosa, 40 5,
3.5, 1.3, 0.3, Setosa, 41 4.5, 2.3, 1.3, 0.3, Setosa, 42 4.4, 3.2, 1.3, 0.2, Setosa, 43 5,
3.5, 1.6, 0.6, Setosa, 44 5.1, 3.8, 1.9, 0.4, Setosa, 45 4.8, 3, 1.4, 0.3, Setosa, 46 5.1,
3.8, 1.6, 0.2, Setosa, 47 4.6, 3.2, 1.4, 0.2, Setosa, 48 5.3, 3.7, 1.5, 0.2, Setosa, 49 5,
3.3, 1.4, 0.2, Setosa, 50 7, 3.2, 4.7, 1.4, Versicolor, 51 6.4, 3.2, 4.5, 1.5, Versicolor, 52
6.9, 3.1, 4.9, 1.5, Versicolor, 53 5.5, 2.3, 4, 1.3, Versicolor, 54 6.5, 2.8, 4.6, 1.5,
Versicolor, 55 5.7, 2.8, 4.5, 1.3, Versicolor, 56 6.3, 3.3, 4.7, 1.6, Versicolor, 57 4.9, 2.4,
3.3, 1, Versicolor, 58 6.6, 2.9, 4.6, 1.3, Versicolor, 59 5.2, 2.7, 3.9, 1.4, Versicolor, 60
5, 2, 3.5, 1, Versicolor, 61 5.9, 3, 4.2, 1.5, Versicolor, 62 6, 2.2, 4, 1, Versicolor, 63
6.1, 2.9, 4.7, 1.4, Versicolor, 64 5.6, 2.9, 3.6, 1.3, Versicolor, 65 6.7, 3.1, 4.4, 1.4,
Versicolor, 66 5.6, 3, 4.5, 1.5, Versicolor, 67 5.8, 2.7, 4.1, 1, Versicolor, 68 6.2, 2.2,
4.5, 1.5, Versicolor, 69 5.6, 2.5, 3.9, 1.1, Versicolor, 70 5.9, 3.2, 4.8, 1.8, Versicolor, 71
6.1, 2.8, 4, 1.3, Versicolor, 72 6.3, 2.5, 4.9, 1.5, Versicolor, 73 6.1, 2.8, 4.7, 1.2,
```

```
Versicolor, 74 6.4, 2.9, 4.3, 1.3, Versicolor, 75 6.6, 3, 4.4, 1.4, Versicolor, 76 6.8, 2.8,
4.8, 1.4, Versicolor, 77 6.7, 3, 5, 1.7, Versicolor, 78 6, 2.9, 4.5, 1.5, Versicolor, 79 5.7,
2.6, 3.5, 1, Versicolor, 80 5.5, 2.4, 3.8, 1.1, Versicolor, 81 5.5, 2.4, 3.7, 1, Versicolor,
82 5.8, 2.7, 3.9, 1.2, Versicolor, 83 6, 2.7, 5.1, 1.6, Versicolor, 84 5.4, 3, 4.5, 1.5,
Versicolor, 85 6, 3.4, 4.5, 1.6, Versicolor, 86 6.7, 3.1, 4.7, 1.5, Versicolor, 87 6.3, 2.3,
4.4, 1.3, Versicolor, 88 5.6, 3, 4.1, 1.3, Versicolor, 89 5.5, 2.5, 4, 1.3, Versicolor, 90
5.5, 2.6, 4.4, 1.2, Versicolor, 91 6.1, 3, 4.6, 1.4, Versicolor, 92 5.8, 2.6, 4, 1.2,
Versicolor, 93 5, 2.3, 3.3, 1, Versicolor, 94 5.6, 2.7, 4.2, 1.3, Versicolor, 95 5.7, 3, 4.2,
1.2, Versicolor, 96 5.7, 2.9, 4.2, 1.3, Versicolor, 97 6.2, 2.9, 4.3, 1.3, Versicolor, 98 5.1,
2.5, 3, 1.1, Versicolor, 99 5.7, 2.8, 4.1, 1.3, Versicolor, 100 6.3, 3.3, 6, 2.5, Virginica,
101 5.8, 2.7, 5.1, 1.9, Virginica, 102 7.1, 3, 5.9, 2.1, Virginica, 103 6.3, 2.9, 5.6, 1.8,
Virginica, 104 6.5, 3, 5.8, 2.2, Virginica, 105 7.6, 3, 6.6, 2.1, Virginica, 106 4.9, 2.5,
4.5, 1.7, Virginica, 107 7.3, 2.9, 6.3, 1.8, Virginica, 108 6.7, 2.5, 5.8, 1.8, Virginica, 109
7.2, 3.6, 6.1, 2.5, Virginica, 110 6.5, 3.2, 5.1, 2, Virginica, 111 6.4, 2.7, 5.3, 1.9,
Virginica, 112 6.8, 3, 5.5, 2.1, Virginica, 113 5.7, 2.5, 5, 2, Virginica, 114 5.8, 2.8, 5.1,
2.4, Virginica, 115 6.4, 3.2, 5.3, 2.3, Virginica, 116 6.5, 3, 5.5, 1.8, Virginica, 117 7.7,
3.8, 6.7, 2.2, Virginica, 118 7.7, 2.6, 6.9, 2.3, Virginica, 119 6, 2.2, 5, 1.5, Virginica,
120 6.9, 3.2, 5.7, 2.3, Virginica, 121 5.6, 2.8, 4.9, 2, Virginica, 122 7.7, 2.8, 6.7, 2,
Virginica, 123 6.3, 2.7, 4.9, 1.8, Virginica, 124 6.7, 3.3, 5.7, 2.1, Virginica, 125 7.2, 3.2,
6, 1.8, Virginica, 126 6.2, 2.8, 4.8, 1.8, Virginica, 127 6.1, 3, 4.9, 1.8, Virginica, 128
6.4, 2.8, 5.6, 2.1, Virginica, 129 7.2, 3, 5.8, 1.6, Virginica, 130 7.4, 2.8, 6.1, 1.9,
Virginica, 131 7.9, 3.8, 6.4, 2, Virginica, 132 6.4, 2.8, 5.6, 2.2, Virginica, 133 6.3, 2.8,
5.1, 1.5, Virginica, 134 6.1, 2.6, 5.6, 1.4, Virginica, 135 7.7, 3, 6.1, 2.3, Virginica, 136
6.3, 3.4, 5.6, 2.4, Virginica, 137 6.4, 3.1, 5.5, 1.8, Virginica, 138 6, 3, 4.8, 1.8,
Virginica, 139 6.9, 3.1, 5.4, 2.1, Virginica, 140 6.7, 3.1, 5.6, 2.4, Virginica, 141 6.9, 3.1,
5.1, 2.3, Virginica, 142 5.8, 2.7, 5.1, 1.9, Virginica, 143 6.8, 3.2, 5.9, 2.3, Virginica, 144
6.7, 3.3, 5.7, 2.5, Virginica, 145 6.7, 3, 5.2, 2.3, Virginica, 146 6.3, 2.5, 5, 1.9,
Virginica, 147 6.5, 3, 5.2, 2, Virginica, 148 6.2, 3.4, 5.4, 2.3, Virginica, 149 5.9, 3, 5.1,
1.8, Virginica, 150 ];
```

KMeansND – функция диаграммы

KMeansND() вычисляет строки диаграммы путем применения кластеризации методом k-средних; для каждой строки диаграммы отображается идентификатор кластера, которому была назначена эта точка диаграммы. Столбцы, используемые алгоритмом кластеризации, определяются соответственно параметрами coordinate_1 и coordinate_2 и т. д. до n столбцов. Все они являются агрегированиями. Количество созданных кластеров определяется параметром num_clusters.

KMeansND возвращает одно значение на точку диаграммы. Возвращенное значение – двойное и является целочисленным значением, соответствующим кластеру, которому была назначена каждая точка диаграммы.

Синтаксис:

```
KMeansND(num_clusters, num_iter, coordinate_1, coordinate_2 [,coordinate_3 [,
...]])
```

Возвращаемые типы данных: двойное значение

Аргументы:

Аргументы

Аргумент	Описание
num_clusters	Целое число, которое указывает количество кластеров.
num_iter	Количество итераций с переинициализированными центрами кластеров.
coordinate_1	Агрегирование, вычисляющее первую координату, обычно ось X (точечной диаграммы, которая может быть сделана из диаграммы). Дополнительные параметры вычисляют вторую, третью и четвертую координаты и т. д.

Пример: Выражение диаграммы
В этом примере создается точечная диаграмма с помощью набора данных *Iris*, и затем с помощью KMeans данные раскрашиваются по выражению.

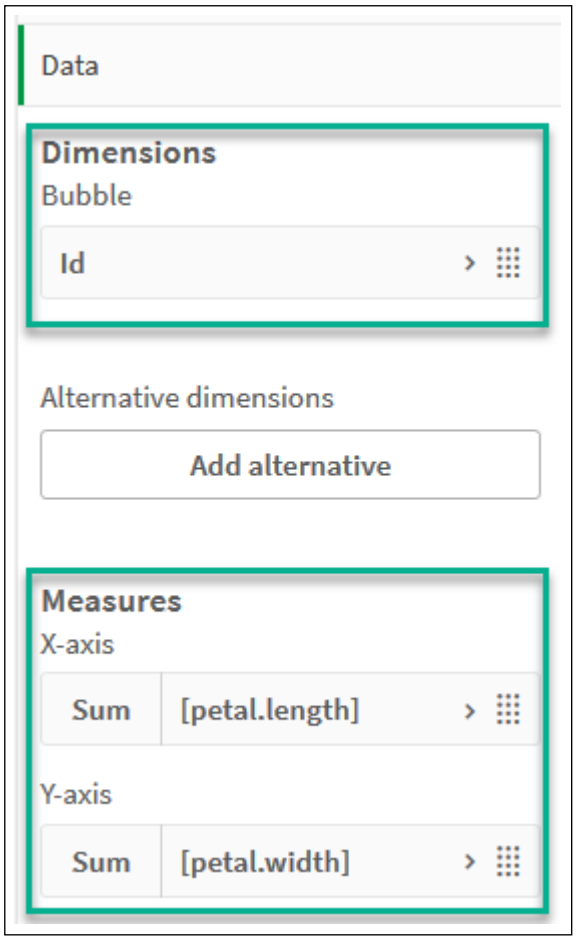
Также мы создаем переменную для аргумента *num_clusters* и затем используем поле ввода переменной, чтобы изменить количество кластеров.

Кроме того, мы создаем переменную для аргумента *num_iter* и затем используем второе поле ввода переменной, чтобы изменить количество итераций.

Набор данных *Iris* общедоступен в различных форматах. Данные предоставлены в виде встроенной таблицы для загрузки с помощью редактора загрузки данных в программе Qlik Sense. Обратите внимание, что к таблице данных для этого примера добавлен столбец *Идентификатор*.

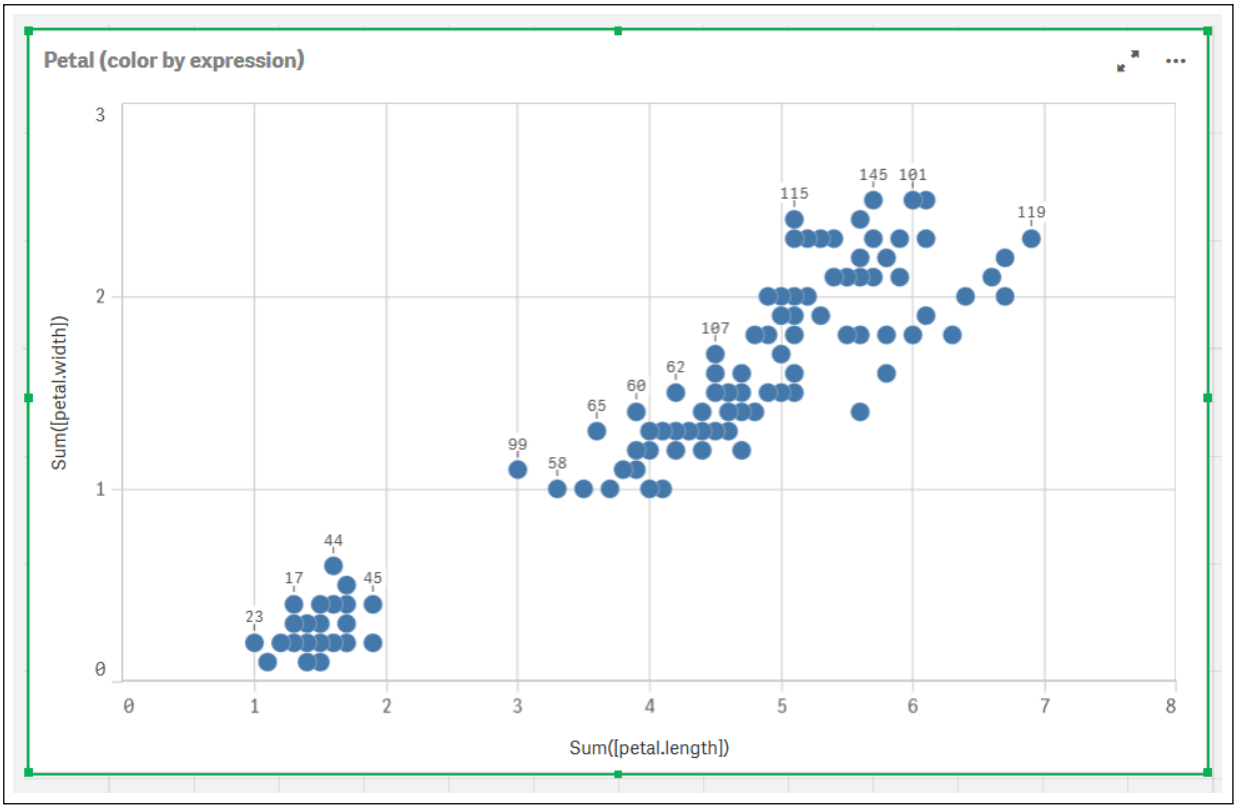
После загрузки данных в Qlik Sense нужно выполнить следующие действия.

1. Перетащите **точечную диаграмму** на новый лист. Назовите диаграмму *Лепесток (раскрашивание по выражению)*.
2. Создайте переменную для указания количества кластеров. Для переменной **Имя** введите *KmeansPetalClusters*. Для переменной **Определение** введите *=2*.
3. Создайте переменную для указания количества итераций. Для переменной **Имя** введите *KmeansNumberIterations*. Для переменной **Определение** введите *=1*.
4. Настройте **данные** для диаграммы.
    i. В разделе **Измерения** выберите *Идентификатор* поля для параметра **Пузырек**. Введите идентификатор кластера для метки.
    ii. В разделе **Меры** выберите *Sum([petal.length])* для выражения для параметра **Ось X**.
    iii. В разделе **Меры** выберите *Sum([petal.width])* для выражения для параметра **Ось Y**.
    Параметры данных для диаграммы Лепесток (раскрашивание по выражению)



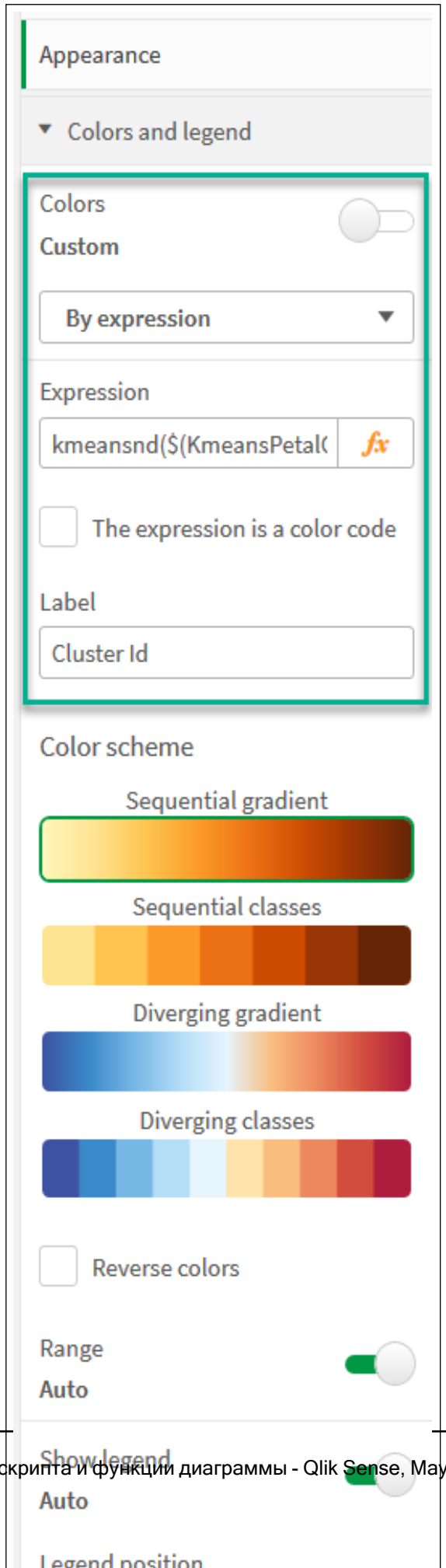
Точки диаграммы нанесены на диаграмму.

Точки диаграммы на диаграмме Лепесток (раскрашивание по выражению)

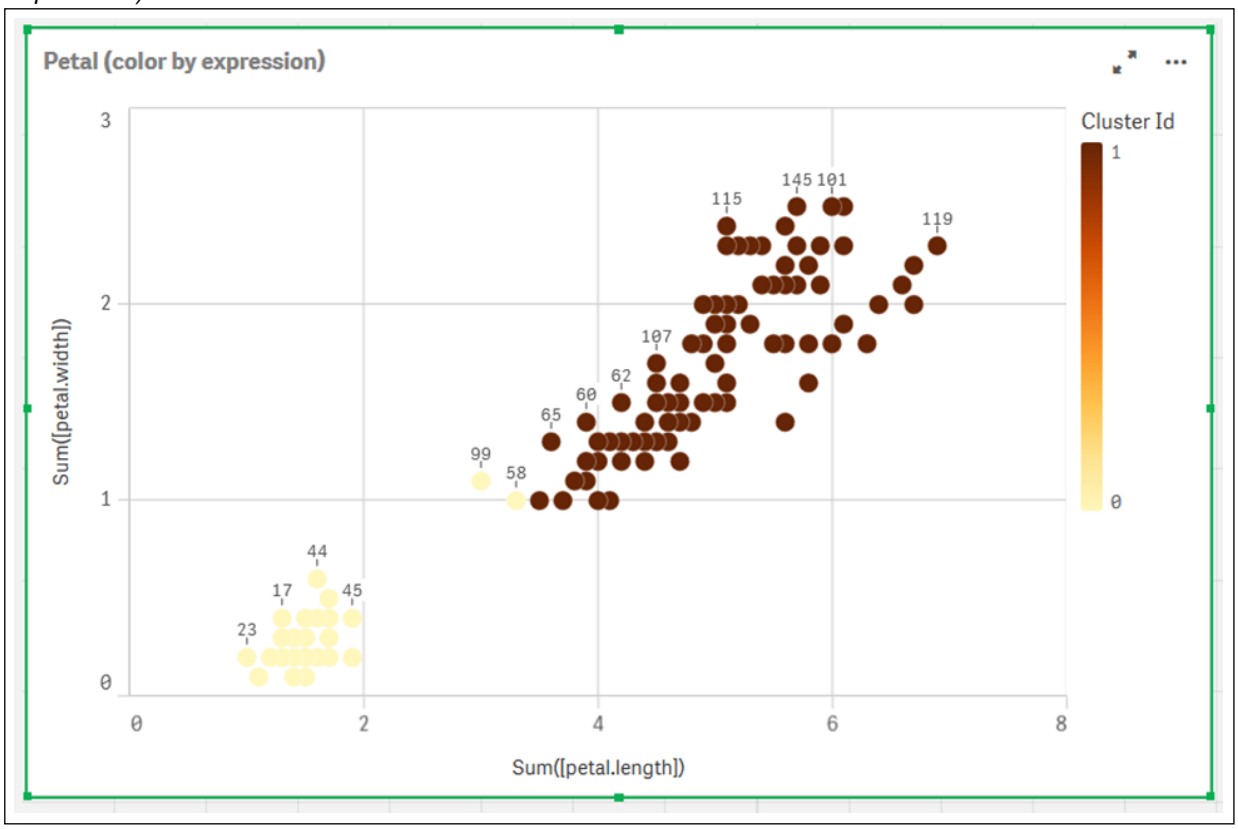


5. Настройте **Вид** диаграммы:
    i. В разделе **Цвета и легенда** выберите **Пользовательский** для параметра **Цвета**.
    ii. Выберите раскрашивание диаграммы **По выражению**.
    iii. Введите следующее для **выражения**: *kmeansnd ($(KmeansPetalClusters),$(KmeansNumberIterations), Sum([petal.length]), Sum ([petal.width]),Sum([sepal.length]), Sum([sepal.width]))*
    Обратите внимание, что *KmeansPetalClusters* – это переменная, для которой установлено *2*. *KmeansNumberIterations* – это переменная, для которой установлено *1*.
    Альтернативно введите следующее: *kmeansnd(2, 2, Sum([petal.length]), Sum ([petal.width]),Sum([sepal.length]), Sum([sepal.width]))*
    iv. Снимите флажок **Выражение является цветовым кодом**.
    v. Введите следующее для **метки**: *Идентификатор кластера*

Параметры вида для диаграммы Лепесток (раскрашивание по выражению)

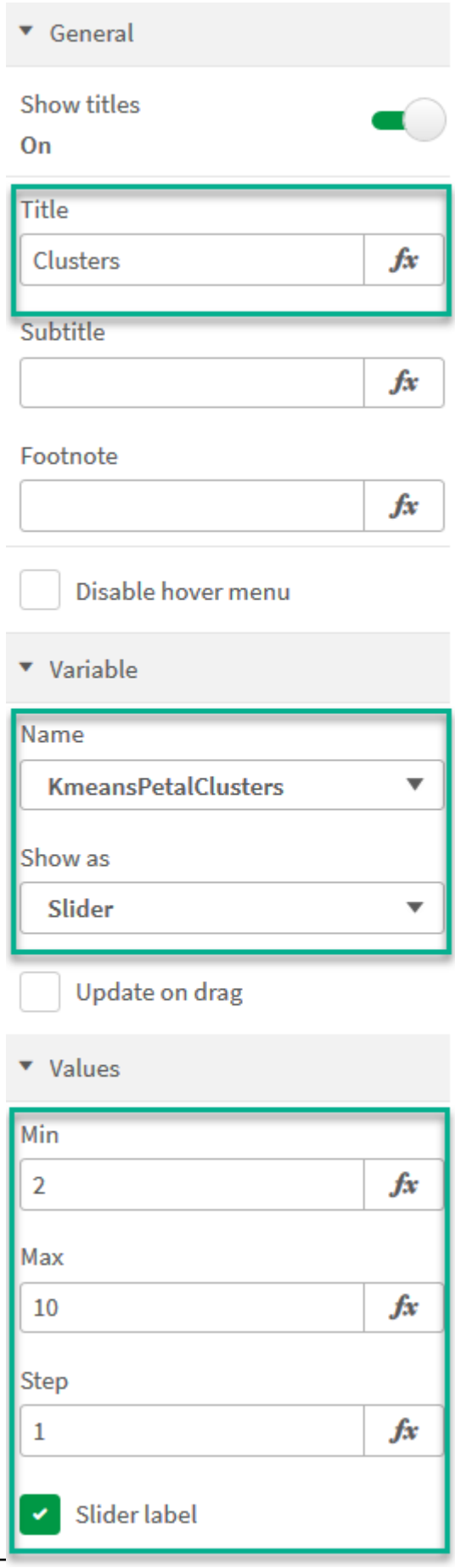

Appearance
Colors and legend
Colors
Custom
By expression
Expression
kmeansnd($(KmeansPetal(
The expression is a color code
Label
Cluster Id
Color scheme
Sequential gradient
Sequential classes
Diverging gradient
Diverging classes
Reverse colors
Range
Auto
Show legend
Auto
Legend position

Два кластера на диаграмме раскрашены по выражению KMeans.
Кластеры, раскрашенные по выражению на диаграммы Лепесток (раскрашивание по выражению)



6. Добавьте поле **Ввод переменной** для количества кластеров.
    i. В разделе **Пользовательские объекты** на панели **Ресурсы** выберите **Qlik Dashboard Bundle**. Если бы к пакету информационной панели не было доступа, количество кластеров можно бы было изменить с помощью созданной переменной или непосредственно с помощью целого числа в выражении.
    ii. Перетащите поле **Ввод переменной** на лист.
    iii. В разделе **Вид** щелкните **Общее**.
    iv. Введите следующее для параметра **Заголовок**: *Кластеры*
    v. Щелкните **Переменная**.
    vi. Выберите следующую переменную для параметра **Имя**: *KmeansPetalClusters*.
    vii. Выберите **Ползунок** для параметра **Показать в виде**.
    viii. Выберите **Значения** и настройте параметры как требуется,

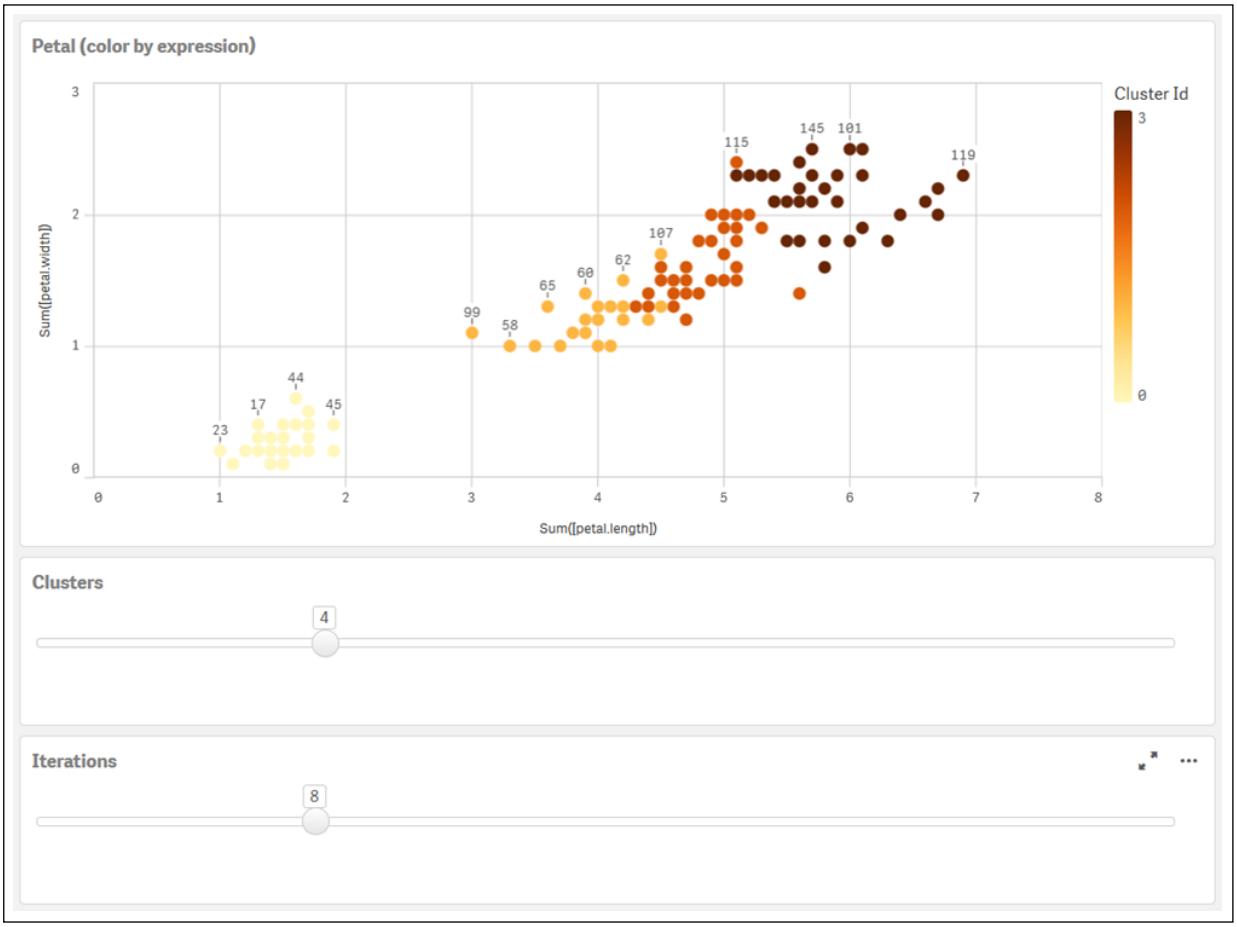
Вид поля ввода переменной Кластеры


General
Show titles
On
Title
Clusters
Subtitle
Footnote
Disable hover menu
Variable
Name
KmeansPetalClusters
Show as
Slider
Update on drag
Values
Min
2
Max
10
Step
1
Slider label

7. Добавьте поле **Ввод переменной** для количества итераций.
    i. Перетащите поле **Ввод переменной** на лист.
    ii. В разделе **Вид** выберите **Общее**.
    iii. Введите следующее для параметра **Заголовок**: *Итерации*
    iv. В разделе **Вид** выберите **Переменная**.
    v. Выберите следующую переменную для параметра **Имя**: *KmeansNumberIterations*.
    vi. Настройте дополнительные параметры как требуется,

    Теперь можно изменить количество кластеров и итераций с помощью ползунков в полях ввода переменной.

    Кластеры, раскрашенные по выражению на диаграммы Лепесток (раскрашивание по выражению)

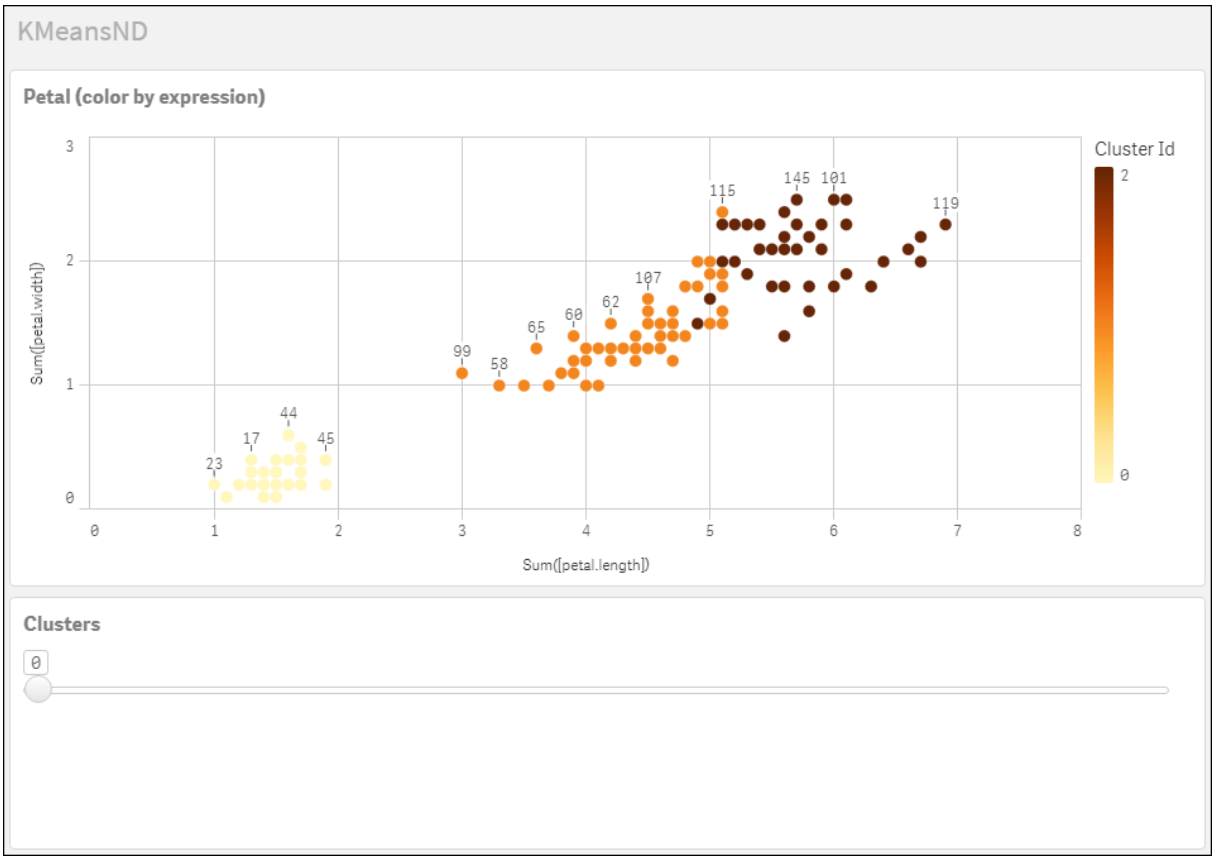


Автоматическая кластеризация

Функции **метода k-средних** поддерживают автоматическую кластеризацию с помощью метода, называемого разницей глубины (DeD). Когда пользователь задает количество кластеров равным 0, оптимальное количество кластеров определяется для набора данных. Обратите внимание, что хотя целое число для количества кластеров (*k*) явно не возвращается, оно вычисляется в алгоритме k-средних. Например, если 0 указан в функции для значения *KmeansPetalClusters* или

установлен через поле ввода переменной, назначения кластеров автоматически вычисляются для набора данных на основе оптимального количества кластеров. В наборе данных Iris, если количество кластеров задано равным 0, оптимальное количество кластеров (3) определяется (автоматическая кластеризация) алгоритмом для этого набора данных.

Метод разницы глубины k-средних определяет оптимальное количество кластеров, когда (k) установлен на 0.



Набор данных Iris: встроенная загрузка для редактора загрузки данных в Qlik Sense

```
IrisData: Load * Inline [ sepal.length, sepal.width, petal.length, petal.width, variety, id
5.1, 3.5, 1.4, 0.2, Setosa, 1 4.9, 3, 1.4, 0.2, Setosa, 2 4.7, 3.2, 1.3, 0.2, Setosa, 3 4.6,
3.1, 1.5, 0.2, Setosa, 4 5, 3.6, 1.4, 0.2, Setosa, 5 5.4, 3.9, 1.7, 0.4, Setosa, 6 4.6, 3.4,
1.4, 0.3, Setosa, 7 5, 3.4, 1.5, 0.2, Setosa, 8 4.4, 2.9, 1.4, 0.2, Setosa, 9 4.9, 3.1, 1.5,
0.1, Setosa, 10 5.4, 3.7, 1.5, 0.2, Setosa, 11 4.8, 3.4, 1.6, 0.2, Setosa, 12 4.8, 3, 1.4,
0.1, Setosa, 13 4.3, 3, 1.1, 0.1, Setosa, 14 5.8, 4, 1.2, 0.2, Setosa, 15 5.7, 4.4, 1.5, 0.4,
Setosa, 16 5.4, 3.9, 1.3, 0.4, Setosa, 17 5.1, 3.5, 1.4, 0.3, Setosa, 18 5.7, 3.8, 1.7, 0.3,
Setosa, 19 5.1, 3.8, 1.5, 0.3, Setosa, 20 5.4, 3.4, 1.7, 0.2, Setosa, 21 5.1, 3.7, 1.5, 0.4,
Setosa, 22 4.6, 3.6, 1, 0.2, Setosa, 23 5.1, 3.3, 1.7, 0.5, Setosa, 24 4.8, 3.4, 1.9, 0.2,
Setosa, 25 5, 3, 1.6, 0.2, Setosa, 26 5, 3.4, 1.6, 0.4, Setosa, 27 5.2, 3.5, 1.5, 0.2, Setosa,
28 5.2, 3.4, 1.4, 0.2, Setosa, 29 4.7, 3.2, 1.6, 0.2, Setosa, 30 4.8, 3.1, 1.6, 0.2, Setosa,
31 5.4, 3.4, 1.5, 0.4, Setosa, 32 5.2, 4.1, 1.5, 0.1, Setosa, 33 5.5, 4.2, 1.4, 0.2, Setosa,
34 4.9, 3.1, 1.5, 0.1, Setosa, 35 5, 3.2, 1.2, 0.2, Setosa, 36 5.5, 3.5, 1.3, 0.2, Setosa, 37
4.9, 3.1, 1.5, 0.1, Setosa, 38 4.4, 3, 1.3, 0.2, Setosa, 39 5.1, 3.4, 1.5, 0.2, Setosa, 40 5,
3.5, 1.3, 0.3, Setosa, 41 4.5, 2.3, 1.3, 0.3, Setosa, 42 4.4, 3.2, 1.3, 0.2, Setosa, 43 5,
3.5, 1.6, 0.6, Setosa, 44 5.1, 3.8, 1.9, 0.4, Setosa, 45 4.8, 3, 1.4, 0.3, Setosa, 46 5.1,
3.8, 1.6, 0.2, Setosa, 47 4.6, 3.2, 1.4, 0.2, Setosa, 48 5.3, 3.7, 1.5, 0.2, Setosa, 49 5,
```

```
3.3, 1.4, 0.2, Setosa, 50 7, 3.2, 4.7, 1.4, Versicolor, 51 6.4, 3.2, 4.5, 1.5, Versicolor, 52
6.9, 3.1, 4.9, 1.5, Versicolor, 53 5.5, 2.3, 4, 1.3, Versicolor, 54 6.5, 2.8, 4.6, 1.5,
Versicolor, 55 5.7, 2.8, 4.5, 1.3, Versicolor, 56 6.3, 3.3, 4.7, 1.6, Versicolor, 57 4.9, 2.4,
3.3, 1, Versicolor, 58 6.6, 2.9, 4.6, 1.3, Versicolor, 59 5.2, 2.7, 3.9, 1.4, Versicolor, 60
5, 2, 3.5, 1, Versicolor, 61 5.9, 3, 4.2, 1.5, Versicolor, 62 6, 2.2, 4, 1, Versicolor, 63
6.1, 2.9, 4.7, 1.4, Versicolor, 64 5.6, 2.9, 3.6, 1.3, Versicolor, 65 6.7, 3.1, 4.4, 1.4,
Versicolor, 66 5.6, 3, 4.5, 1.5, Versicolor, 67 5.8, 2.7, 4.1, 1, Versicolor, 68 6.2, 2.2,
4.5, 1.5, Versicolor, 69 5.6, 2.5, 3.9, 1.1, Versicolor, 70 5.9, 3.2, 4.8, 1.8, Versicolor, 71
6.1, 2.8, 4, 1.3, Versicolor, 72 6.3, 2.5, 4.9, 1.5, Versicolor, 73 6.1, 2.8, 4.7, 1.2,
Versicolor, 74 6.4, 2.9, 4.3, 1.3, Versicolor, 75 6.6, 3, 4.4, 1.4, Versicolor, 76 6.8, 2.8,
4.8, 1.4, Versicolor, 77 6.7, 3, 5, 1.7, Versicolor, 78 6, 2.9, 4.5, 1.5, Versicolor, 79 5.7,
2.6, 3.5, 1, Versicolor, 80 5.5, 2.4, 3.8, 1.1, Versicolor, 81 5.5, 2.4, 3.7, 1, Versicolor,
82 5.8, 2.7, 3.9, 1.2, Versicolor, 83 6, 2.7, 5.1, 1.6, Versicolor, 84 5.4, 3, 4.5, 1.5,
Versicolor, 85 6, 3.4, 4.5, 1.6, Versicolor, 86 6.7, 3.1, 4.7, 1.5, Versicolor, 87 6.3, 2.3,
4.4, 1.3, Versicolor, 88 5.6, 3, 4.1, 1.3, Versicolor, 89 5.5, 2.5, 4, 1.3, Versicolor, 90
5.5, 2.6, 4.4, 1.2, Versicolor, 91 6.1, 3, 4.6, 1.4, Versicolor, 92 5.8, 2.6, 4, 1.2,
Versicolor, 93 5, 2.3, 3.3, 1, Versicolor, 94 5.6, 2.7, 4.2, 1.3, Versicolor, 95 5.7, 3, 4.2,
1.2, Versicolor, 96 5.7, 2.9, 4.2, 1.3, Versicolor, 97 6.2, 2.9, 4.3, 1.3, Versicolor, 98 5.1,
2.5, 3, 1.1, Versicolor, 99 5.7, 2.8, 4.1, 1.3, Versicolor, 100 6.3, 3.3, 6, 2.5, Virginica,
101 5.8, 2.7, 5.1, 1.9, Virginica, 102 7.1, 3, 5.9, 2.1, Virginica, 103 6.3, 2.9, 5.6, 1.8,
Virginica, 104 6.5, 3, 5.8, 2.2, Virginica, 105 7.6, 3, 6.6, 2.1, Virginica, 106 4.9, 2.5,
4.5, 1.7, Virginica, 107 7.3, 2.9, 6.3, 1.8, Virginica, 108 6.7, 2.5, 5.8, 1.8, Virginica, 109
7.2, 3.6, 6.1, 2.5, Virginica, 110 6.5, 3.2, 5.1, 2, Virginica, 111 6.4, 2.7, 5.3, 1.9,
Virginica, 112 6.8, 3, 5.5, 2.1, Virginica, 113 5.7, 2.5, 5, 2, Virginica, 114 5.8, 2.8, 5.1,
2.4, Virginica, 115 6.4, 3.2, 5.3, 2.3, Virginica, 116 6.5, 3, 5.5, 1.8, Virginica, 117 7.7,
3.8, 6.7, 2.2, Virginica, 118 7.7, 2.6, 6.9, 2.3, Virginica, 119 6, 2.2, 5, 1.5, Virginica,
120 6.9, 3.2, 5.7, 2.3, Virginica, 121 5.6, 2.8, 4.9, 2, Virginica, 122 7.7, 2.8, 6.7, 2,
Virginica, 123 6.3, 2.7, 4.9, 1.8, Virginica, 124 6.7, 3.3, 5.7, 2.1, Virginica, 125 7.2, 3.2,
6, 1.8, Virginica, 126 6.2, 2.8, 4.8, 1.8, Virginica, 127 6.1, 3, 4.9, 1.8, Virginica, 128
6.4, 2.8, 5.6, 2.1, Virginica, 129 7.2, 3, 5.8, 1.6, Virginica, 130 7.4, 2.8, 6.1, 1.9,
Virginica, 131 7.9, 3.8, 6.4, 2, Virginica, 132 6.4, 2.8, 5.6, 2.2, Virginica, 133 6.3, 2.8,
5.1, 1.5, Virginica, 134 6.1, 2.6, 5.6, 1.4, Virginica, 135 7.7, 3, 6.1, 2.3, Virginica, 136
6.3, 3.4, 5.6, 2.4, Virginica, 137 6.4, 3.1, 5.5, 1.8, Virginica, 138 6, 3, 4.8, 1.8,
Virginica, 139 6.9, 3.1, 5.4, 2.1, Virginica, 140 6.7, 3.1, 5.6, 2.4, Virginica, 141 6.9, 3.1,
5.1, 2.3, Virginica, 142 5.8, 2.7, 5.1, 1.9, Virginica, 143 6.8, 3.2, 5.9, 2.3, Virginica, 144
6.7, 3.3, 5.7, 2.5, Virginica, 145 6.7, 3, 5.2, 2.3, Virginica, 146 6.3, 2.5, 5, 1.9,
Virginica, 147 6.5, 3, 5.2, 2, Virginica, 148 6.2, 3.4, 5.4, 2.3, Virginica, 149 5.9, 3, 5.1,
1.8, Virginica, 150 ];
```

KMeansCentroid2D – функция диаграммы

KMeansCentroid2D() вычисляет строки диаграммы путем применения кластеризации методом k-средних; для каждой строки диаграммы отображается желаемая координата кластера, которому была назначена эта точка диаграммы. Столбцы, используемые алгоритмом кластеризации, определяются соответственно параметрами coordinate_1 и coordinate_2. Они оба являются агрегированиями. Количество созданных кластеров определяется параметром num_clusters. Данные могут быть при необходимости нормализованы с помощью параметра нормы.

KMeansCentroid2D возвращает одно значение на точку диаграммы. Возвращенное значение – двойное и является одной из координат позиции, соответствующей центру кластера, которому была назначена эта точка диаграммы.

Синтаксис:

```
KMeansCentroid2D(num_clusters, coordinate_no, coordinate_1, coordinate_2 [, norm])
```

Возвращаемые типы данных: двойное значение

Аргументы:

Аргументы

Аргумент	Описание
num_clusters	Целое число, которое указывает количество кластеров.
coordinate_no	Желаемое количество координат средних точек (соответствующих, например, осям x, y или z).
coordinate_1	Агрегирование, вычисляющее первую координату, обычно ось X точечной диаграммы, которая может быть сделана из диаграммы. Дополнительный параметр coordinate_2 вычисляет вторую координату.
norm	Дополнительный метод нормализации применяется к наборам данных перед кластеризацией методом k-средних. Возможные значения: 0 или 'нет' при отсутствии нормализации 1 или ‘zscore’ для нормализации с помощью z-оценки 2 или ‘minmax’ для нормализации с помощью мин./макс. Если параметры не предоставлены или предоставленный параметр неправильный, нормализация не применяется. Z-оценка нормализует данные на основе среднего и стандартного отклонения признака. Z-оценка не гарантирует, что у каждого признака будет одинаковый масштаб, но при выбросах этот подход лучше, чем мин./макс. Нормализация с помощью мин./макс. гарантирует, что признаки имеют одинаковый масштаб; для этого берутся минимальное и максимальное значения каждого признака и каждая точка данных вычисляется заново.

Автоматическая кластеризация

Функции **метода k-средних** поддерживают автоматическую кластеризацию с помощью метода, называемого разницей глубины (DeD). Когда пользователь задает количество кластеров равным 0, оптимальное количество кластеров определяется для набора данных. Обратите внимание, что хотя целое число для количества кластеров (*k*) явно не возвращается, оно вычисляется в алгоритме k-средних. Например, если 0 указан в функции для значения *KmeansPetalClusters* или установлен через поле ввода переменной, назначения кластеров автоматически вычисляются для набора данных на основе оптимального количества кластеров.

KMeansCentroidND — функция диаграммы

KMeansCentroidND() вычисляет строки диаграммы путем применения кластеризации методом k-средних; для каждой строки диаграммы отображается желаемая координата кластера, которому была назначена эта точка диаграммы. Столбцы, используемые алгоритмом кластеризации, определяются соответственно параметрами coordinate_1 и coordinate_2 и т. д. до n столбцов. Все они являются агрегированиями. Количество созданных кластеров определяется параметром num_clusters.

KMeansCentroidND возвращает одно значение на строку. Возвращенное значение — двойное и является одной из координат позиции, соответствующей центру кластера, которому была назначена эта точка диаграммы.

Синтаксис:

```
KMeansCentroidND((num_clusters, num_iter, coordinate_no, coordinate_1,
coordinate_2 [,coordinate_3 [, ...]])
```

Возвращаемые типы данных: двойное значение

Аргументы:

Аргументы

Аргумент	Описание
num_clusters	Целое число, которое указывает количество кластеров.
num_iter	Количество итераций с переинициализированными центрами кластеров.
coordinate_no	Желаемое количество координат средних точек (соответствующих, например, осям x, y или z).
coordinate_1	Агрегирование, вычисляющее первую координату, обычно ось X (точечной диаграммы, которая может быть сделана из диаграммы). Дополнительные параметры вычисляют вторую, третью и четвертую координаты и т. д.

Автоматическая кластеризация

Функции **метода k-средних** поддерживают автоматическую кластеризацию с помощью метода, называемого разницей глубины (DeD). Когда пользователь задает количество кластеров равным 0, оптимальное количество кластеров определяется для набора данных. Обратите внимание, что хотя целое число для количества кластеров (*k*) явно не возвращается, оно вычисляется в алгоритме k-средних. Например, если 0 указан в функции для значения *KmeansPetalClusters* или установлен через поле ввода переменной, назначения кластеров автоматически вычисляются для набора данных на основе оптимального количества кластеров.

5.23 Функции статистического распределения

Функции статистического распределения DIST измеряют вероятность функции распределения в точке распределения, заданной предоставленным значением. Функции INV вычисляют значение, заданное вероятностью распределения. В отличие от них, группы функций статистического агрегирования вычисляют агрегированные значения последовательностей статистических тестовых значений для проверки различных статистических гипотез.

Все описанные ниже функции статистического распределения реализованы в программе Qlik Sense с помощью библиотеки функций Cephes. Ссылки и подробная информация об используемых алгоритмах, точности и т. д. см. на веб-странице Cephes library. Библиотека функций Cephes используется с разрешения.

Все функции можно использовать как в скрипте загрузки данных, так и в выражениях диаграмм.

Обзор функций статистического распределения

Каждая функция подробно описана после обзора. Также можно щелкнуть имя функции в синтаксисе, чтобы получить немедленный доступ к подробной информации об этой конкретной функции.

CHIDIST
CHIDIST() возвращает одностороннюю вероятность распределения chi^2. Распределение chi^2 ассоциируется с критерием ch^{i2}.

```
CHIDIST (value, degrees_freedom)
```

CHIINV
CHIINV() возвращает обратную одностороннюю вероятность распределения chi^2.

```
CHIINV (prob, degrees_freedom)
```

NORMDIST
NORMDIST() возвращает накопленное нормальное распределение указанного среднего значения и стандартного отклонения. Если значение mean = 0, а значение standard_dev = 1, функция возвращает стандартное нормальное распределение.

```
NORMDIST (value, mean, standard_dev)
```

NORMINV
NORMINV() возвращает противоположное нормальное распределение указанного среднего значения и стандартного отклонения.

```
NORMINV (prob, mean, standard_dev)
```

TDIST

TDIST() возвращает вероятность t-распределения, в котором числовое значение является вычисляемым значением t, для которого должна подсчитываться вероятность.

```
TDIST (value, degrees_freedom, tails)
```

TINV

TINV() возвращает t-значение t-распределения в виде функции вероятности и степеней свободы.

```
TINV (prob, degrees_freedom)
```

FDIST

FDIST() возвращает вероятность F-распределения.

```
FDIST (value, degrees_freedom1, degrees_freedom2)
```

FINV

FINV() возвращает обратную вероятность F-распределения.

```
FINV (prob, degrees_freedom1, degrees_freedom2)
```

См. также:

Функции статистического агрегирования (page 282)

CHIDIST

CHIDIST() возвращает одностороннюю вероятность распределения chi^2. Распределение chi^2 ассоциируется с критерием ch^{i2}.

Синтаксис:

```
CHIDIST(value, degrees_freedom)
```

Возвращаемые типы данных: число

Аргументы:

Аргументы

Аргумент	Описание
value	Значение, при котором необходимо оценить распределение. Значение не должно быть отрицательным.
degrees_ freedom	Положительное целое число, которое указывает число степеней свободы.

Эта функция связана с функцией **CHIINV** следующим образом:
`If prob = CHIDIST(value,df), then CHIINV(prob, df) = value`

Ограничения:

Все аргументы должны быть числовыми, в противном случае будет возвращено значение NULL.

Примеры и результаты:

Пример	Результат
CHIDIST(8, 15)	Возвращает 0,9238

CHIINV

CHIINV() возвращает обратную одностороннюю вероятность распределения chi^2.

Синтаксис:

```
CHIINV(prob, degrees_freedom)
```

Возвращаемые типы данных: число

Аргументы:

Аргументы

Аргумент	Описание
prob	Вероятность, связанная с распределением chi^2. Значение должно быть числом от 0 до 1.
degrees_ freedom	Целое число, которое указывает число степеней свободы.

Эта функция связана с функцией **CHIDIST** следующим образом:
`If prob = CHIDIST(value,df), then CHIINV(prob, df) = value`

Ограничения:

Все аргументы должны быть числовыми, в противном случае будет возвращено значение NULL.

Примеры и результаты:

Пример	Результат
CHIINV(0.9237827, 15)	Возвращает 8,0000

FDIST

FDIST() возвращает вероятность F-распределения.

Синтаксис:

```
FDIST(value, degrees_freedom1, degrees_freedom2)
```

Возвращаемые типы данных: число

Аргументы:

Аргументы

Аргумент	Описание
value	Значение, при котором необходимо оценить распределение. Элемент **Value** не должен быть отрицательным.
degrees_ freedom1	Положительное целое число, которое указывает число степеней свободы числителя.
degrees_ freedom2	Положительное целое число, которое указывает число степеней свободы знаменателя.

Эта функция связана с функцией **FINV** следующим образом:
`If prob = FDIST(value, df1, df2), then FINV(prob, df1, df2) = value`

Ограничения:

Все аргументы должны быть числовыми, в противном случае будет возвращено значение NULL.

Примеры и результаты:

Пример	Результат
FDIST(15, 8, 6)	Возвращает 0,0019

FINV

FINV() возвращает обратную вероятность F-распределения.

Синтаксис:

```
FINV(prob, degrees_freedom1, degrees_freedom2)
```

Возвращаемые типы данных: число

Аргументы:

Аргументы

Аргумент	Описание
prob	Вероятность, связанная с F-распределением, которая должна быть числом от 0 до 1.
degrees_ freedom	Целое число, которое указывает число степеней свободы.

Эта функция связана с функцией **FDIST** следующим образом:
`If prob = FDIST(value, df1, df2), then FINV(prob, df1, df2) = value`

Ограничения:

Все аргументы должны быть числовыми, в противном случае будет возвращено значение NULL.

Примеры и результаты:

Пример	Результат
FINV(0.0019369, 8, 6)	Возвращает 15,0000

NORMDIST

NORMDIST() возвращает накопленное нормальное распределение указанного среднего значения и стандартного отклонения. Если значение mean = 0, а значение standard_dev = 1, функция возвращает стандартное нормальное распределение.

Синтаксис:

```
NORMDIST(value, [mean], [standard_dev], [cumulative])
```

Возвращаемые типы данных: число

Аргументы:

Аргументы

Аргумент	Описание
value	Значение, при котором необходимо оценить распределение.
mean	Дополнительное значение, указывающее среднее арифметическое для распределения. Если данный аргумент не указан, значение по умолчанию – 0.
standard_ dev	Дополнительное положительное значение, указывающее стандартное отклонение распределения. Если данный аргумент не указан, значение по умолчанию – 1.
cumulative	При желании можно настроить использование стандартного нормального распределения или накопленного распределения. 0 = стандартное нормальное распределение 1 = накопленное распределение (по умолчанию)

Эта функция связана с функцией **NORMINV** следующим образом:
`If prob = NORMDIST(value, m, sd), then NORMINV(prob, m, sd) = value`

Ограничения:

Все аргументы должны быть числовыми, в противном случае будет возвращено значение NULL.

Примеры и результаты:

Пример	Результат
NORMDIST(0.5, 0, 1)	Возвращает 0,6915

NORMINV

NORMINV() возвращает противоположное нормальное распределение указанного среднего значения и стандартного отклонения.

Синтаксис:

```
NORMINV(prob, mean, standard_dev)
```

Возвращаемые типы данных: число

Аргументы:

Аргументы

Аргумент	Описание
prob	Вероятность, связанная с нормальным распределением. Значение должно быть числом от 0 до 1.
mean	Значение, указывающее среднее арифметическое для распределения.
standard_dev	Положительное значение, указывающее стандартное отклонение распределения.

Эта функция связана с функцией **NORMDIST** следующим образом:
`If prob = NORMDIST(value, m, sd), then NORMINV(prob, m, sd) = value`

Ограничения:

Все аргументы должны быть числовыми, в противном случае будет возвращено значение NULL.

Примеры и результаты:

Пример	Результат
NORMINV(0.6914625, 0, 1)	Возвращает 0,5000

TDIST

TDIST() возвращает вероятность t-распределения, в котором числовое значение является вычисляемым значением t, для которого должна подсчитываться вероятность.

Синтаксис:

```
TDIST(value, degrees_freedom, tails)
```

Возвращаемые типы данных: число

Аргументы:

Аргументы

Аргумент	Описание
value	Значение, при котором необходимо оценить распределение и которое не должно быть отрицательным.
degrees_ freedom	Положительное целое число, которое указывает число степеней свободы.
tails	Должно составлять 1 (одностороннее распределение) или 2 (двустороннее распределение).

Эта функция связана с функцией **TINV** следующим образом:
`If prob = TDIST(value, df ,2), then TINV(prob, df) = value`

Ограничения:

Все аргументы должны быть числовыми, в противном случае будет возвращено значение NULL.

Примеры и результаты:

Пример	Результат
TDIST(1, 30, 2)	Возвращает 0,3253

TINV

TINV() возвращает t-значение t-распределения в виде функции вероятности и степеней свободы.

Синтаксис:

```
TINV(prob, degrees_freedom)
```

Возвращаемые типы данных: число

Аргументы:

Аргументы

Аргумент	Описание
prob	Двусторонняя вероятность, связанная с t-распределением. Значение должно быть числом от 0 до 1.
degrees_ freedom	Целое число, которое указывает число степеней свободы.

Ограничения:

Все аргументы должны быть числовыми, в противном случае будет возвращено значение NULL.

Эта функция связана с функцией **TDIST** следующим образом:
`If prob = TDIST(value, df ,2), then TINV(prob, df) = value.`

Примеры и результаты:

Пример	Результат
TINV(0.3253086, 30)	Возвращает 1,0000

5.24 Строковые функции

В этом разделе описаны функции для обработки и управления строками.

Все функции можно использовать как в скрипте загрузки данных, так и в выражениях диаграмм, кроме функции **Evaluate**, которую можно использовать только в скрипте загрузки данных.

Обзор строковых функций

Каждая функция подробно описана после обзора. Также можно щелкнуть имя функции в синтаксисе, чтобы получить немедленный доступ к подробной информации об этой конкретной функции.

Capitalize

Capitalize() возвращает строку со всеми словами, которые начинаются с заглавной буквы.

```
Capitalize (text)
```

Chr

Chr() возвращает символ Unicode, соответствующий входному целому числу.

```
Chr (int)
```

Evaluate

Evaluate() определяет, можно ли входную текстовую строку рассматривать в качестве допустимого выражения Qlik Sense. Если да, то возвращает значение выражения в качестве строки. Если входная строка не является допустимым выражением, будет возвращено значение NULL.

```
Evaluate (expression_text)
```

FindOneOf

FindOneOf() выполняет поиск в строке, чтобы найти положение вхождения любого символа из набора указанных символов. Положение вхождения любого символа из набора для поиска возвращается, если указан третий аргумент (значение больше 1). Если совпадений не найдено, возвращается значение **0**.

```
FindOneOf (text, char_set[, count])
```

Hash128

Hash128() возвращает 128-разрядный хэш сочетания значений входного выражения. Результат – строка из 22 символов.

```
Hash128 (expr{, expression})
```

Hash160

Hash160() возвращает 160-разрядный хэш сочетания значений входного выражения. Результат – строка из 27 символов.

```
Hash160 (expr{, expression})
```

Hash256

Hash256() возвращает 256-разрядный хэш сочетания значений входного выражения. Результат – строка из 43 символов.

```
Hash256 (expr{, expression})
```

Index

Index() выполняет поиск в строке, чтобы найти n-ное положение вхождения указанной подстроки. Дополнительный третий аргумент определяет значение n (1, если игнорируется). Если указано отрицательное значение, поиск выполняется с конца строки. Позиции в строке нумеруются от **1** и далее.

```
Index  (text, substring[, count])
```

KeepChar

KeepChar() возвращает строку, состоящую из первой строки «text», за вычетом всех символов, НЕ содержащихся во второй строке «keep_chars».

```
KeepChar (text, keep_chars)
```

Left

Left() возвращает строку, состоящую из первых (крайних слева) символов входной строки, где число символов определяется вторым аргументом.

```
Left (text, count)
```

Len

Len() возвращает длину входной строки.

```
Len (text)
```

LevenshteinDist

LevenshteinDist() возвращает расстояние Levenshtein – разность между двумя строками. Оно определяется как минимальное количество односимвольных правок (вставки, удаления или замены), требуемых для превращения одной строки в другую. Функция полезна для нечеткого сравнения строк.

```
LevenshteinDist (text1, text2)
```

Lower

Lower() преобразует все символы входной строки в нижний регистр.

```
Lower (text)
```

LTrim

LTrim() возвращает входную строку без начальных пробелов.

```
LTrim (text)
```

Mid

Mid() возвращает часть входной строки, начинающуюся с символа, определенного вторым аргументом «start», и возвращает количество символов, определенных третьим аргументом «count». Если «count» отсутствует, возвращается остальная часть входной строки. Первый символ во входной строке имеет номер 1.

```
Mid (text, start[, count])
```

Ord

Ord() возвращает номер кодовой точки Unicode первого символа входной строки.

```
Ord (text)
```

PurgeChar

PurgeChar() возвращает строку, состоящую из всех символов входной строки («text»), кроме символов, указанных в строке второго аргумента («remove_chars»).

```
PurgeChar (text, remove_chars)
```

Repeat

Repeat() возвращает строку, состоящую из входной строки, повторяющейся столько раз, сколько указано вторым аргументом.

```
Repeat (text[, repeat_count])
```

Replace

Replace() возвращает строку после замены всех вхождений определенной подстроки во входной строке на другую подстроку. Функция нерекурсивная и работает слева направо.

```
Replace (text, from_str, to_str)
```

Right

Right() возвращает строку, состоящую из последних символов (справа) входной строки, где число символов определяется вторым аргументом.

```
Right (text, count)
```

RTrim

RTrim() возвращает входную строку без конечных пробелов.

```
RTrim (text)
```

SubField

SubField() используется для извлечения компонентов подстроки из поля родительской строки, где поля исходной записи состоят из двух или более частей, разделенных знаком разделителя.

```
SubField (text, delimiter[, field_no ])
```

SubStringCount

SubStringCount() возвращает количество вхождений указанной подстроки в тексте входной строки. Если совпадения отсутствуют, возвращается 0.

```
SubStringCount (text, substring)
```

TextBetween

TextBetween() возвращает текст входной строки, заключенный между символами, указанными в качестве разделителей.

```
TextBetween (text, delimiter1, delimiter2[, n])
```

Trim

Trim() возвращает входную строку без начальных и конечных пробелов.

```
Trim (text)
```

Upper

Upper() преобразует все символы входной строки в верхний регистр для всех буквенных символов в выражении. Цифры и символы игнорируются.

```
Upper (text)
```

Capitalize

Capitalize() возвращает строку со всеми словами, которые начинаются с заглавной буквы.

Синтаксис:

```
Capitalize(text)
```

Возвращаемые типы данных: строка

Пример: Выражения диаграммы

Пример	Результат
`Capitalize ( 'star trek' )`	Возвращает 'Star Trek'
`Capitalize ( 'AA bb cC Dd')`	Возвращает 'Aa Bb Cc Dd'

Пример: Скрипт загрузки

```
Load  String, Capitalize(String) Inline [String rHode iSland washingTon d.C. new york];
```

Результат

Строка	Capitalize(String)
rHode iSland	Rhode Island
washingTon d.C.	Washington D.C.
new york	New York

Chr

Chr() возвращает символ Unicode, соответствующий входному целому числу.

Синтаксис:

```
Chr(int)
```

Возвращаемые типы данных: строка

Примеры и результаты:

Пример	Результат
Chr(65)	Возвращает строку 'A'
Chr(163)	Возвращает строку '£'
Chr(35)	Возвращает строку '#'

Evaluate

Evaluate() определяет, можно ли входную текстовую строку рассматривать в качестве допустимого выражения Qlik Sense. Если да, то возвращает значение выражения в качестве строки. Если входная строка не является допустимым выражением, будет возвращено значение NULL.

Синтаксис:

```
Evaluate(expression_text)
```

Возвращаемые типы данных: двойное значение

Эта строковая функция не может использоваться в выражениях диаграмм.

Примеры и результаты:

Пример функции	Результат
`Evaluate ( 5 * 8 )`	Возвращает '40'

Пример скрипта загрузки

```
Load  Evaluate(String) as Evaluated, String Inline [String 4 5+3 0123456789012345678 Today()
];
```

Результат

Строка	Оценено
4	4
5+3	8
0123456789012345678	0123456789012345678
Today()	2022-02-02

FindOneOf

FindOneOf() выполняет поиск в строке, чтобы найти положение вхождения любого символа из набора указанных символов. Положение вхождения любого символа из набора для поиска возвращается, если указан третий аргумент (значение больше 1). Если совпадений не найдено, возвращается значение **0**.

Синтаксис:

```
FindOneOf(text, char_set[, count])
```

Возвращаемые типы данных: целое

Аргументы:

Аргументы

Аргумент	Описание
text	Оригинальная строка.
char_set	Набор символов для поиска в text.
count	Определяет, какое вхождение символов искать. Например, значение 2 служит для поиска второго вхождения.

Пример: Выражения диаграммы

Пример	Результат
`FindOneOf( 'my example text string', 'et%s')`	Возвращает 4, так как буква e – четвертый символ в строке примера.

Пример	Результат
`FindOneOf( 'my example text string', 'et%s', 3)`	Возвращает 12, так как поиск выполняется для любого из символов e, t, % или s, и «t» является третьим элементом и его позиция 12 в строке примера.
`FindOneOf( 'my example text string', '¤%&')`	Возвращает 0, так как в строке примера нет ни одного из символов ¤, % или &.

Пример: Скрипт загрузки

```
Load * Inline [SearchFor, Occurrence et%s,1 et%s,3 ¤%&,1]
```

Результат

SearchFor	Occurrence	FindOneOf('мой пример текстовой строки', SearchFor, Occurrence)
et%s	1	4
et%s	3	12
¤%&	1	0

Hash128

Hash128() возвращает 128-разрядный хэш сочетания значений входного выражения. Результат – строка из 22 символов.

Синтаксис:

```
Hash128(expr{, expression})
```

Возвращаемые типы данных: строка

Пример: Выражения диаграммы

Пример	Результат
Hash128 ('abc', 'xyz', '123')	Возвращает 'MA&5]6+3=:>:>G%S<U*S2+'.
Hash128 (Region, Year, Month) Note: Region, Year, and Month are table fields.	Возвращает 'G7*=6GKPJ(Z+)^KM?<$'A+'.

Пример: Скрипт загрузки

```
Hash_128: Load *,  Hash128(Region, Year, Month) as Hash128; Load * inline [ Region, Year, Month abc, xyz, 123 EU, 2022, 01 UK, 2022, 02 US, 2022, 02 ];
```

Результат

Регион	Год	Месяц	Hash128
abc	xyz	123	MA&5]6+3=:>;>G%S<U*S2+
EU	2022	01	B40^K&[T@!;VB'XR]<5=/$
UK	2022	02	O5T;+1?[B&"F&1//MA[MN!
US	2022	02	C6@#]4#_G-(]J7EQY#KRW0

Hash160

Hash160() возвращает 160-разрядный хэш сочетания значений входного выражения. Результат – строка из 27 символов.

Синтаксис:

```
Hash160(expr{, expression})
```

Возвращаемые типы данных: строка

Пример: Выражения диаграммы

Пример	Результат
Hash160 ('abc', 'xyz', '123')	Возвращает 'MA&5]6+3=:>;>G%S<U*S2I:`=X*'.
Hash160 (Region, Year, Month) Note: Region, Year, and Month are table fields.	Возвращает 'G7*=6GKPJ (Z+)^KM?<$'AI.)?U$'.

Пример: Скрипт загрузки

```
Hash_160: Load *,  Hash160(Region, Year, Month) as Hash160; Load * inline [ Region, Year,
Month abc, xyz, 123 EU, 2022, 01 UK, 2022, 02 US, 2022, 02 ];
```

Результат

Регион	Год	Месяц	Hash160
abc	xyz	123	MA&5]6+3=:>;>G%S<U*S2I:`=X*
EU	2022	01	B40^K&[T@!;VB'XR]<5=//_F853
UK	2022	02	O5T;+1?[B&"F&1//MA[MN!T"FWZ
US	2022	02	C6@#]4#_G-(]J7EQY#KRW`@KF+W

Hash256

Hash256() возвращает 256-разрядный хэш сочетания значений входного выражения. Результат – строка из 43 символов.

Синтаксис:

```
Hash256(expr{, expression})
```

Возвращаемые типы данных: строка

Пример: Выражения диаграммы

Пример	Результат
Hash256 ('abc', 'xyz', '123')	Возвращает 'MA&5]6+3=:>;>G%S<U*S2I:`=X*A.IO*8N\%Y7Q;YEJ'.
Hash256 (Region, Year, Month) Note: Region, Year, and Month are table fields.	Возвращает 'G7*=6GKPJ(Z+)^KM?<$'AI.)?U$#X2RB[:0ZP=+Z`F:'.

Пример: Скрипт загрузки

```
Hash_256: Load *,  Hash256(Region, Year, Month) as Hash256; Load * inline [ Region, Year, Month abc, xyz, 123 EU, 2022, 01 UK, 2022, 02 US, 2022, 02 ];
```

Результат

Регион	Год	Месяц	Hash256
abc	xyz	123	MA&5]6+3=:>;>G%S<U*S2I:`=X*A.IO*8N\%Y7Q;YEJ
EU	2022	01	B40^K&[T@!;VB'XR]<5=//_F853?BE6'G&,YH*T'MF)
UK	2022	02	O5T;+1?[B&"F&1//MA[MN!T"FWZT=4\#V`M%6_\0C>4
US	2022	02	C6@#]4#_G-(]J7EQY#KRW`@KF+W-0]`[Z8R+#'")=+0

Index

Index() выполняет поиск в строке, чтобы найти n-ное положение вхождения указанной подстроки. Дополнительный третий аргумент определяет значение n (1, если игнорируется). Если указано отрицательное значение, поиск выполняется с конца строки. Позиции в строке нумеруются от 1 и далее.

Синтаксис:

```
Index(text, substring[, count])
```

Возвращаемые типы данных: целое

Аргументы:

Аргументы

Аргумент	Описание
text	Оригинальная строка.
substring	Строка символов для поиска в text.
count	Определяет, какое вхождение символов **substring** искать. Например, значение 2 служит для поиска второго вхождения.

Примеры и результаты:

Пример	Результат
Index('abcdefg', 'cd')	Возвращает 3
Index('abcdabcd', 'b', 2)	Возвращает 6 (второе вхождение «b»)
Index('abcdabcd', 'b',-2)	Возвращает 2 (второе вхождение «b», начиная с конца)
`Left( Date, Index( Date,'-') -1 )` where **Date** = 1997-07-14	Возвращает 1997
`Mid( Date, Index( Date, '-', 2 ) -2, 2 )` where **Date** = 1997-07-14	Возвращает 07

Пример: Скрипт

```
T1: Load  *,  index(String, 'cd') as Index_CD,              // returns 3  in Index_CD index
(String, 'b') as Index_B,               // returns 2 in Index_B index(String, 'b', -1) as
Index_B2;           // returns 2 or 6 in Index_B2 Load * inline [ String abcdefg abcdabcd ];
```

KeepChar

KeepChar() возвращает строку, состоящую из первой строки «text», за вычетом всех символов, НЕ содержащихся во второй строке «keep_chars».

Синтаксис:

```
KeepChar(text, keep_chars)
```

Возвращаемые типы данных: строка

Аргументы:

Аргументы

Аргумент	Описание
text	Оригинальная строка.
keep_chars	Строка, содержащая символы в text, которую необходимо сохранить.

Пример: Выражения диаграммы

Пример	Результат
`KeepChar ( 'a1b2c3','123' )`	Возвращает «123».
`KeepChar ( 'a1b2c3','1234' )`	Возвращает «123».
`KeepChar ( 'a1b22c3','1234' )`	Возвращает «1223».
`KeepChar ( 'a1b2c3','312' )`	Возвращает «123».

Пример: Скрипт загрузки

```
T1: Load  *,  keepchar(String1, String2) as KeepChar;   Load * inline [ String1, String2
'a1b2c3', '123' ];
```

Результаты

В таблице Qlik Sense представлены результаты использования функции *KeepChar* в скрипте загрузки.

String1	String2	KeepChar
a1b2c3	123	123

См. также:

- *PurgeChar (page 848)*

Left

Left() возвращает строку, состоящую из первых (крайних слева) символов входной строки, где число символов определяется вторым аргументом.

Синтаксис:

```
Left(text, count)
```

Возвращаемые типы данных: строка

Аргументы:

Аргумент	Описание
text	Оригинальная строка.
count	Определяет количество символов, которые необходимо включить из левой части строки **text**.

Пример: Выражение диаграммы

Пример	Результат
`Left('abcdef', 3)`	Возвращает 'abc'

Пример: Скрипт загрузки

```
T1: Load  *,  left(Text,Start) as Left;        Load * inline [ Text, Start 'abcdef', 3 '2021-07-14', 4 '2021-07-14', 2 ];
```

Результат

В таблице Qlik Sense представлены результаты использования функции *Left* в скрипте загрузки.

Текст	Начало	Left
abcdef	3	'abc'
2021-07-14	4	2021
2021-07-14	2	20

См. также *Index (page 840)* для более сложного анализа строк.

Len

Len() возвращает длину входной строки.

Синтаксис:

```
Len(text)
```

Возвращаемые типы данных: целое число

Пример: Выражение диаграммы

Пример	Результат
`Len('Peter')`	Возвращает '5'

Пример: Скрипт загрузки

```
T1: Load String, First&Second as NewString; Load *, mid(String,len(First)+1) as Second; Load *, upper(left(String,1)) as First; Load  * inline [ String this is a sample text string capitalize first letter only ];
```

Результат

Строка	NewString
this is a sample text string	This is a sample text string
capitalize first letter only	Capitalize first letter only

LevenshteinDist

LevenshteinDist() возвращает расстояние Levenshtein – разность между двумя строками. Оно определяется как минимальное количество односимвольных правок (вставки, удаления или замены), требуемых для превращения одной строки в другую. Функция полезна для нечеткого сравнения строк.

Синтаксис:

```
LevenshteinDist(text1, text2)
```

Возвращаемые типы данных: целое число

Пример: Выражение диаграммы

Пример	Результат
`LevenshteinDist('Kitten','Sitting')`	Возвращает '3'

Пример: Скрипт загрузки

Скрипт загрузки

```
T1: Load *, recno() as ID; Load 'Silver' as String_1,* inline [ String_2 Sliver SSiver SSiveer ];  T1: Load *, recno()+3 as ID; Load 'Gold' as String_1,* inline [ String_2 Bold Bool Bond ];  T1: Load *, recno()+6 as ID; Load 'Ove' as String_1,* inline [ String_2 Ove Uve Üve ];  T1: Load *, recno()+9 as ID; Load 'ABC' as String_1,* inline [ String_2 DEFG abc ビビビ ];  set nullinterpret = '<NULL>'; T1: Load *, recno()+12 as ID; Load 'X' as String_1,* inline [ String_2 '' <NULL> 1 ];  R1: Load  ID,  String_1,  String_2,  LevenshteinDist(String_1, String_2) as LevenshteinDistance  resident T1;  Drop table T1;
```

Результат

Идентификатор	String_1	String_2	LevenshteinDistance
1	Silver	Sliver	2
2	Silver	SSiver	2
3	Silver	SSiveer	3
4	Gold	Полужирный	1
5	Gold	Bool	3
6	Gold	Bond	2
7	Ove	Ove	0
8	Ove	Uve	1
9	Ove	Üve	1
10	ABC	DEFG	4
11	ABC	'abc'	3
12	ABC	ビビビ	3
13	X		1
14	X	-	1
15	X	1	1

Lower

Lower() преобразует все символы входной строки в нижний регистр.

Синтаксис:

```
Lower(text)
```

Возвращаемые типы данных: строка

Пример: Выражение диаграммы

Пример	Результат
`Lower('abcD')`	Возвращает 'abcd'

Пример: Скрипт загрузки

```
Load  String, Lower(String) Inline [String rHode iSland washingTon d.C. new york];
```

Результат

Строка	Lower(String)
rHode iSland	rhode island
washingTon d.C.	washington d.c.
new york	new york

LTrim

LTrim() возвращает входную строку без начальных пробелов.

Синтаксис:

```
LTrim(text)
```

Возвращаемые типы данных: строка

Пример: Выражения диаграммы

Пример	Результат
`LTrim( ' abc' )`	Возвращает 'abc'
`LTrim( 'abc ' )`	Возвращает 'abc '

Пример: Скрипт загрузки

```
Set verbatim=1;  T1:  Load *, len(LtrimString) as LtrimStringLength;       Load *, ltrim
(String) as LtrimString; Load *, len(String) as StringLength;                Load * Inline [
String '   abc    ' ' def  '];
```

> *Оператор "Set verbatim=1" включен в пример для того, чтобы пробелы не были автоматически удалены до демонстрации функции ltrim. Для получения более подробной информации см. раздел Verbatim (page 156).*

Результат

Строка	StringLength	LtrimStringLength
def	6	5
abc	10	7

См. также:

- *RTrim (page 852)*

Mid

Mid() возвращает часть входной строки, начинающуюся с символа, определенного вторым аргументом «start», и возвращает количество символов, определенных третьим аргументом «count». Если «count» отсутствует, возвращается остальная часть входной строки. Первый символ во входной строке имеет номер 1.

Синтаксис:

```
Mid(text, start[, count])
```

Возвращаемые типы данных: строка

Аргументы:

Аргументы

Аргумент	Описание
text	Оригинальная строка.
start	Целое число, определяющее положение первого символа в text для добавления.
count	Определяет длину выводимой строки. Если не указано, добавляются все символы с позиции, определенные функцией **start**.

Пример: Выражения диаграммы

Пример	Результат
`Mid('abcdef',3 )`	Возвращает «cdef»
`Mid('abcdef',3, 2 )`	Возвращает «cd»

Пример: Скрипт загрузки

```
T1: Load *,  mid(Text,Start) as Mid1,            mid(Text,Start,Count) as Mid2;     Load *
inline [ Text, Start, Count 'abcdef', 3, 2 'abcdef', 2, 3 '210714', 3, 2 '210714', 2, 3 ];
```

Результат

В таблице Qlik Sense представлены результаты использования функции *Mid* в скрипте загрузки.

Текст	Начало	Mid1	Счетчик	Mid2
abcdef	2	bcdef	3	bcd
abcdef	3	cdef	2	cd
210714	2	10714	3	107
210714	3	0714	2	07

См. также:

- *Index (page 840)*

Ord

Ord() возвращает номер кодовой точки Unicode первого символа входной строки.

Синтаксис:

```
Ord(text)
```

Возвращаемые типы данных: целое

Примеры и результаты:

Пример: Выражение диаграммы

Пример	Результат
`Ord('A')`	Возвращает целое число 65.
`Ord('Ab')`	Возвращает целое число 65.

Пример: Скрипт загрузки

```
//Guqin (Chinese: 古琴) - 7-stringed zithers T2: Load *,  ord(Chinese) as OrdUnicode,
        ord(Western) as OrdASCII;                      Load * inline [ Chinese, Western 古琴,
Guqin ];
```

Результат:

Chinese	Western	OrdASCII	OrdUnicode
古琴	Guqin	71	21476

PurgeChar

PurgeChar() возвращает строку, состоящую из всех символов входной строки («text»), кроме символов, указанных в строке второго аргумента («remove_chars»).

Синтаксис:

```
PurgeChar(text, remove_chars)
```

Возвращаемые типы данных: строка

Аргументы:

Аргументы

Аргумент	Описание
text	Оригинальная строка.
remove_chars	Строка, содержащая символы в text, которую необходимо удалить.

Возвращаемые типы данных: строка

Пример: Выражения диаграммы

Пример	Результат
`PurgeChar ( 'a1b2c3','123' )`	Возвращает 'abc'.
`PurgeChar ( 'a1b2c3','312' )`	Возвращает 'abc'.

Пример: Скрипт загрузки

```
T1: Load  *,  purgechar(String1, String2) as PurgeChar;   Load * inline [ String1, String2
'a1b2c3', '123' ];
```

Результаты

В таблице Qlik Sense представлены результаты использования функции *PurgeChar* в скрипте загрузки.

String1	String2	PurgeChar
a1b2c3	123	'abc'

См. также:

- *KeepChar (page 841)*

Repeat

Repeat() возвращает строку, состоящую из входной строки, повторяющейся столько раз, сколько указано вторым аргументом.

Синтаксис:

```
Repeat(text[, repeat_count])
```

Возвращаемые типы данных: строка

Аргументы:

Аргументы

Аргумент	Описание
text	Оригинальная строка.
repeat_count	Определяет, сколько раз символы в строке **text** повторяются в выводимой строке.

Пример: Выражение диаграммы

Пример	Результат
`Repeat( ' * ', rating )` when **rating** = 4	Возвращает '****'

Пример: Скрипт загрузки

```
T1: Load *, repeat(String,2) as Repeat; Load * inline [ String hello world! hOw aRe you? ];
```

Результат

Строка	Повторять
hello world!	hello world!hello world!
hOw aRe you?	hOw aRe you?hOw aRe you?

Replace

Replace() возвращает строку после замены всех вхождений определенной подстроки во входной строке на другую подстроку. Функция нерекурсивная и работает слева направо.

Синтаксис:

```
Replace(text, from_str, to_str)
```

Возвращаемые типы данных: строка

Аргументы:

Аргументы

Аргумент	Описание
text	Оригинальная строка.
from_str	Строка, встречающаяся один или несколько раз во входной строке **text**.
to_str	Строка, заменяющая все вхождения **from_str** в строке **text**.

Примеры и результаты:

Пример	Результат
`Replace('abccde','cc','xyz')`	Возвращает 'abxyzde'

См. также:

Right

Right() возвращает строку, состоящую из последних символов (справа) входной строки, где число символов определяется вторым аргументом.

Синтаксис:

```
Right(text, count)
```

Возвращаемые типы данных: строка

Аргументы:

Аргументы

Аргумент	Описание
text	Оригинальная строка.
count	Определяет количество символов, которые необходимо включить из крайней правой части строки **text**.

Пример: Выражение диаграммы

Пример	Результат
`Right('abcdef', 3)`	Возвращает 'def'

Пример: Скрипт загрузки

```
T1: Load  *,  right(Text,Start) as Right;        Load * inline [ Text, Start 'abcdef', 3
'2021-07-14', 4 '2021-07-14', 2 ];
```

Результат

В таблице Qlik Sense представлены результаты использования функции *Right* в скрипте загрузки.

Текст	Начало	Right
abcdef	3	def
2021-07-14	4	7-14
2021-07-14	2	14

RTrim

RTrim() возвращает входную строку без конечных пробелов.

Синтаксис:

```
RTrim(text)
```

Возвращаемые типы данных: строка

Пример: Выражения диаграммы

Пример	Результат
`RTrim( ' abc' )`	Возвращает ' abc'
`RTrim( 'abc ' )`	Возвращает 'abc'

Пример: Скрипт загрузки

```
Set verbatim=1;    T1:  Load *, len(RtrimString) as RtrimStringLength;         Load *, rtrim
(String) as RtrimString; Load *, len(String) as StringLength;              Load * Inline [
String '   abc    ' ' def  '];
```

> *Оператор "Set verbatim=1" включен в пример для того, чтобы пробелы не были автоматически удалены до демонстрации функции rtrim. Для получения более подробной информации см. раздел Verbatim (page 156).*

Результат

Строка	StringLength	RtrimStringLength
def	6	4
abc	10	6

См. также:

- *LTrim (page 846)*

SubField

SubField() используется для извлечения компонентов подстроки из поля родительской строки, где поля исходной записи состоят из двух или более частей, разделенных знаком разделителя.

Функцию **Subfield()** можно использовать, например для извлечения имени или фамилии из списка записей, состоящего из полных имен, отдельных частей имени пути или для извлечения данных из таблиц с данными, разделенными запятыми.

Если используется функция **Subfield()** в операторе **LOAD** и дополнительный параметр field_no не указан, для каждой подстроки будет создана одна полная запись. Если с помощью функции **Subfield()** загружено несколько полей, будет создано декартово произведение всех возможных комбинаций.

Синтаксис:

```
SubField(text, delimiter[, field_no ])
```

Возвращаемые типы данных: строка

Аргументы:

Аргументы

Аргумент	Описание
text	Оригинальная строка. Это может быть неизменяемый текст, переменная, расширение со знаком доллара или другое выражение.
delimiter	Символ во входной строке **text**, разделяющий строку на части.
field_no	Дополнительный третий аргумент, являющийся целым числом, который указывает, какие подстроки родительской строки **text** необходимо вернуть. Используйте значение 1 для возврата первой подстроки, значение 2 для возврата второй подстроки и так далее. • Если **field_no** является положительным значением, подстроки извлекаются слева направо. • Если **field_no** является отрицательным значением, подстроки извлекаются справа налево.

Функцию SubField() можно использовать вместо сложных комбинаций таких функций, как Len(), Right(), Left(), Mid() и другие строковые функции.

Примеры: Выражения скрипта и диаграммы с использованием SubField

Примеры. Выражения скрипта и диаграммы

Базовые примеры

Пример	Результат
`SubField(S, ';' ,2)`	Возвращает 'cde', если **S** = 'abc;cde;efg'.
`SubField(S, ';' ,1)`	Возвращает пустую строку, если элемент **S** – пустая строка.

Пример	Результат
`SubField(S, ';' ,1)`	Возвращает пустую строку, если элемент **S** =';'.
Предполагается, что уже есть переменная, которая содержит имя пути vMyPath, `Set vMyPath=\Users\ext_jrb\Documents\Qlik\Sense\Apps;.`	В диаграмме текста и изображения можно добавить меру, такую как: `SubField(vMyPath, '\',-3)`, что даст результат «Qlik», потому что это третья подстрока, если считать с правого конца переменной vMyPath.

Пример скрипта 1

Скрипт загрузки

Загрузите следующие выражения скрипта и данные в редакторе загрузки данных.

```
FullName:                                    LOAD * inline [ Name 'Dave Owen' 'Joe Tem' ];  SepNames:  Lo
(Name, ' ',1) as FirstName, SubField(Name, ' ',-1) as Surname Resident FullName; Drop Table
FullName;
```

Создать визуализацию

Создайте на листе Qlik Sense визуализацию таблицы с измерениями **Name**, **FirstName** и **SurName**.

Результат

Name	FirstName	SurName
Dave Owen	Dave	Owen
Joe Tem	Joe	Tem

Объяснение

Функция **SubField()** извлекает первую подстроку из **Name**, задавая аргументу **field_no** значение 1. Так как **field_no** имеет положительное значение, при извлечении подстроки соблюдается порядок «слева направо». Второй вызов функции извлекает вторую подстроку, задавая аргументу **field _no** значение -1, при этом соблюдается порядок «справа налево».

Пример скрипта 2

Скрипт загрузки

Загрузите следующие выражения скрипта и данные в редакторе загрузки данных.

```
LOAD DISTINCT Instrument, SubField(Player,',') as Player, SubField(Project,',') as Project;
Load * inline [ Instrument|Player|Project Guitar|Neil,Mike|Music,Video Guitar|Neil|Music,OST
Synth|Neil,Jen|Music,Video,OST Synth|Jo|Music Guitar|Neil,Mike|Music,OST ] (delimiter is '|');
```

Создать визуализацию

Создайте на листе Qlik Sense визуализацию таблицы с измерениями **Instrument**, **Player** и **Project**.

Результат

Instrument	Player	Project
Guitar	Mike	Music
Guitar	Mike	Video
Guitar	Mike	OST
Guitar	Neil	Music
Guitar	Neil	Video
Guitar	Neil	OST
Synth	Jen	Music
Synth	Jen	Video
Synth	Jen	OST
Synth	Jo	Music
Synth	Neil	Music
Synth	Neil	Video
Synth	Neil	OST

Объяснение

В этом примере показано, как с помощью нескольких экземпляров функции **Subfield()**, в каждом из которых не указан параметр field_no, из одного оператора **LOAD** создать декартово произведение всех возможных комбинаций. Параметр **DISTINCT** используется, чтобы не допустить создания дубликатов записей.

SubStringCount

SubStringCount() возвращает количество вхождений указанной подстроки в тексте входной строки. Если совпадения отсутствуют, возвращается 0.

Синтаксис:

```
SubStringCount(text, sub_string)
```

Возвращаемые типы данных: целое

Аргументы:

Аргумент	Описание
text	Оригинальная строка.
sub_string	Строка, встречающаяся один или несколько раз во входной строке **text**.

Пример: Выражения диаграммы

Пример	Результат
`SubStringCount ( 'abcdefgcdxyz', 'cd' )`	Возвращает '2'
`SubStringCount ( 'abcdefgcdxyz', 'dc' )`	Возвращает '0'

Пример: Скрипт загрузки

```
T1: Load *,  substringcount(upper(Strings),'AB') as SubStringCount_AB; Load * inline [ Strings
ABC:DEF:GHI:AB:CD:EF:GH aB/cd/ef/gh/Abc/abandoned ];
```

Результат

Строки	SubStringCount_AB
aB/cd/ef/gh/Abc/abandoned	3
ABC:DEF:GHI:AB:CD:EF:GH	2

TextBetween

TextBetween() возвращает текст входной строки, заключенный между символами, указанными в качестве разделителей.

Синтаксис:

```
TextBetween(text, delimiter1, delimiter2[, n])
```

Возвращаемые типы данных: строка

Аргументы:

Аргумент	Описание
text	Оригинальная строка.
delimiter1	Указывает первый символ-разделитель (или строку) для поиска в **text**.
delimiter2	Указывает второй символ-разделитель (или строку) для поиска в **text**.
n	Указывает, между каким вхождением пары разделителей выполнять поиск. Например, значение 2 возвращает символы между вторым вхождением разделителя1 и вторым вхождением разделителя2.

Пример: Выражения диаграммы

Пример	Результат
`TextBetween('<abc>', '<', '>')`	Возвращает 'abc'
`TextBetween('<abc><de>', '<', '>',2)`	Возвращает 'de'

Пример	Результат
`TextBetween('abc', '<', '>')` `TextBetween('<a<b', '<', '>')`	Оба примера возвращают NULL. Если в строке отсутствуют разделители, возвращается NULL.
`TextBetween('<>', '<', '>')`	Возвращается строка нулевой длины.
`TextBetween('<abc>', '<', '>', 2)`	Возвращается NULL, так как n превышает число вхождений разделителей.

Пример: Скрипт загрузки

```
Load *, textbetween(Text,'<','>') as TextBetween, textbetween(Text,'<','>',2) as
SecondTextBetween; Load * inline [ Text <abc><de> <def><ghi><jkl> ];
```

Результат

Текст	TextBetween	SecondTextBetween
<abc><de>	abc	de
<def><ghi><jkl>	def	ghi

Trim

Trim() возвращает входную строку без начальных и конечных пробелов.

Синтаксис:

```
Trim(text)
```

Возвращаемые типы данных: строка

Примеры и результаты:

Пример: Выражение диаграммы

Пример	Результат
`Trim( ' abc' )`	Возвращает 'abc'
`Trim( 'abc ' )`	Возвращает 'abc'
`Trim( ' abc ' )`	Возвращает 'abc'

Пример: Скрипт загрузки

```
Set verbatim=1;                                          T1: Load *, len(TrimString) as TrimStringLength;
(String) as TrimString; Load *, len(String) as StringLength;               Load * inline [
String '   abc    ' ' def  '](delimiter is '\t');
```

Оператор "Set verbatim=1" включен в пример для того, чтобы пробелы не были автоматически удалены до демонстрации функции trim. Для получения более подробной информации см. раздел Verbatim (page 156).

Результат:

Строка	StringLength	TrimStringLength
def	6	3
abc	10	3

Upper

Upper() преобразует все символы входной строки в верхний регистр для всех буквенных символов в выражении. Цифры и символы игнорируются.

Синтаксис:

```
Upper(text)
```

Возвращаемые типы данных: строка

Пример: Выражение диаграммы

Пример	Результат
`Upper(' abcD')`	Возвращает 'ABCD'

Пример: Скрипт загрузки

```
Load  String,Upper(String) Inline [String rHode iSland washingTon d.C. new york];
```

Результат

Строка	upper(String)
rHode iSland	RHODE ISLAND
washingTon d.C.	WASHINGTON D.C.
new york	NEW YORK

5.25 Системные функции

Системные функции обеспечивают возможность доступа к системе, устройству и свойствам приложения Qlik Sense.

Обзор системных функций

Некоторые функции подробно описаны после обзора. Для этих функций можно щелкнуть имя функции в синтаксисе, чтобы получить немедленный доступ к подробной информации об этой конкретной функции.

Author()

Эта функция скрипта возвращает строку, содержащую свойство автора текущего приложения. Это можно использовать как в скрипте загрузки данных, так и в выражении диаграмм.

Не удалось задать свойство автора в текущей версии Qlik Sense. При перемещении документа QlikView свойство автора сохранится.

ClientPlatform()

Эта функция возвращает строку агента пользователя браузера клиента. Это можно использовать как в скрипте загрузки данных, так и в выражении диаграмм.

Пример:

```
Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/35.0.1916.114 Safari/537.36
```

ComputerName

Эта функция возвращает строку, содержащую имя компьютера, возвращенное операционной системой. Это можно использовать как в скрипте загрузки данных, так и в выражении диаграмм.

Если длина имени компьютера превышает 15 символов, строка будет содержать только первые 15 символов.

```
ComputerName( )
```

DocumentName

Эта функция возвращает строку, содержащую имя текущего приложения Qlik Sense, с расширением, но без пути. Это можно использовать как в скрипте загрузки данных, так и в выражении диаграмм.

```
DocumentName( )
```

DocumentPath

Эта функция возвращает строку, содержащую полный путь к текущему приложению Qlik Sense. Это можно использовать как в скрипте загрузки данных, так и в выражении диаграмм.

```
DocumentPath( )
```

Эта функция не поддерживается в стандартном режиме.

DocumentTitle

Эта функция возвращает строку, содержащую заголовок текущего приложения Qlik Sense. Это можно использовать как в скрипте загрузки данных, так и в выражении диаграмм.

```
DocumentTitle( )
```

EngineVersion

Эта функция возвращает полную версию подсистемы Qlik Sense в виде строки.

```
EngineVersion ()
```

GetCollationLocale

Эта функция скрипта возвращает имя культуры используемой локали сортировки. Если переменная CollationLocale не задана, возвращается локаль машины фактического пользователя.

```
GetCollationLocale( )
```

GetObjectField

GetObjectField() возвращает имя измерения. **Index** – дополнительное целое число, обозначающее измерение, которое необходимо вернуть.

```
GetObjectField – функция диаграммы([index])
```

GetRegistryString

Эта функция возвращает значение ключа в реестре Windows. Это можно использовать как в скрипте загрузки данных, так и в выражении диаграмм.

```
GetRegistryString(path, key)
```

Эта функция не поддерживается в стандартном режиме.

IsPartialReload

Эта функция возвращает -1 (True), если текущая перезагрузка является частичной, в противном случае возвращает 0 (False).

```
IsPartialReload ()
```

OSUser

Эта функция возвращает строку, содержащую имя текущего пользователя. Это можно использовать как в скрипте загрузки данных, так и в выражении диаграмм.

```
OSUser( )
```

В Qlik Sense Desktop и Qlik Sense Mobile Client Managed эта функция всегда возвращает «Personal\Me».

ProductVersion

Эта функция возвращает полную версию программы Qlik Sense и номер сборки в виде строки.

Эта функция устарела и заменена функцией **EngineVersion()**.

```
ProductVersion ()
```

ReloadTime

Эта функция возвращает метку времени завершения последней загрузки данных. Это можно использовать как в скрипте загрузки данных, так и в выражении диаграмм.

```
ReloadTime( )
```

StateName

Элемент **StateName()** возвращает название другого состояния визуализации, в которой он используется. Например, элемент StateName можно использовать для создания визуализаций с динамическим текстом и цветами, отражающими изменение состояния визуализации. Данную функцию можно использовать в выражениях диаграмм, но нельзя применять для определения состояния, к которому относится выражение.

```
StateName – функция диаграммы()
```

EngineVersion

Эта функция возвращает полную версию подсистемы Qlik Sense в виде строки.

Синтаксис:

```
EngineVersion()
```

IsPartialReload

Эта функция возвращает -1 (True), если текущая перезагрузка является частичной, в противном случае возвращает 0 (False).

Синтаксис:

```
IsPartialReload()
```

ProductVersion

Эта функция возвращает полную версию программы Qlik Sense и номер сборки в виде строки. Эта функция устарела и заменена функцией **EngineVersion()**.

Синтаксис:

```
ProductVersion()
```

StateName – функция диаграммы

Элемент **StateName()** возвращает название другого состояния визуализации, в которой он используется. Например, элемент StateName можно использовать для создания визуализаций с динамическим текстом и цветами, отражающими изменение состояния визуализации. Данную функцию можно использовать в выражениях диаграмм, но нельзя применять для определения состояния, к которому относится выражение.

Синтаксис:

```
StateName ()
```

Example 1:

Динамический текст

```
='Region - ' & if(StateName() = '$', 'Default', StateName())
```

Example 2:

Динамические цвета

```
if(StateName() = 'Group 1', rgb(152, 171, 206),
      if(StateName() = 'Group 2', rgb(187, 200, 179),
            rgb(210, 210, 210)
      )
)
```

5.26 Функции таблиц

Функции таблиц извлекают информацию о таблице данных, из которой в настоящее время производится считывание. Если имя таблицы не указано, и функция используется в операторе **LOAD**, то рассматривается текущая таблица.

Все функции можно использовать как в скрипте загрузки данных, но только функцию **NoOfRows** можно использовать в выражении диаграмм.

Обзор функций таблицы

Некоторые функции подробно описаны после обзора. Для этих функций можно щелкнуть имя функции в синтаксисе, чтобы получить немедленный доступ к подробной информации об этой конкретной функции.

FieldName

Функция скрипта **FieldName** возвращает имя поля с указанным номером в ранее загруженной таблице. Если функция используется в операторе **LOAD**, то она не должна ссылаться на таблицу, загружаемую в настоящее время.

```
FieldName (field_number ,table_name)
```

FieldNumber

Функция скрипта **FieldNumber** возвращает номер указанного поля в ранее загруженной таблице. Если функция используется в операторе **LOAD**, то она не должна ссылаться на таблицу, загружаемую в настоящее время.

```
FieldNumber (field_name ,table_name)
```

NoOfFields

Функция скрипта NoOfFields возвращает число полей в ранее загруженной таблице. Если функция используется в операторе LOAD, то она не должна ссылаться на таблицу, загружаемую в настоящее время.

```
NoOfFields (table_name)
```

NoOfRows

Функция скрипта NoOfRows возвращает число строк (записей) в ранее загруженной таблице. Если функция используется в операторе LOAD, то она не должна ссылаться на таблицу, загружаемую в настоящее время.

```
NoOfRows (table_name)
```

NoOfTables

Эта функция скрипта возвращает число ранее загруженных таблиц.

```
NoOfTables()
```

TableName

Эта функция скрипта возвращает имя таблицы с указанным номером.

```
TableName(table_number)
```

TableNumber

Эта функция скрипта возвращает номер указанной таблицы. Первая таблица имеет число 0.

Если table_name не существует, возвращается значение NULL.

```
TableNumber(table_name)
```

Пример:

В этом примере мы хотим создать таблицу с информацией о загруженных таблицах и полях.

Сначала мы загрузим несколько данных образца. В результате будут созданы две таблицы, которые будут использоваться для иллюстрации функций таблицы, описанных в этом разделе.

```
Characters:
Load Chr(RecNo()+Ord('A')-1) as Alpha, RecNo() as Num autogenerate 26;

ASCII:
Load
 if(RecNo()>=65 and RecNo()<=90,RecNo()-64) as Num,
 Chr(RecNo()) as AsciiAlpha,
 RecNo() as AsciiNum
autogenerate 255
 Where (RecNo()>=32 and RecNo()<=126) or RecNo()>=160 ;
```

Далее мы повторяем это во всех таблицах, загруженных с помощью функции NoOfTables, и во всех полях каждой таблицы с помощью функции NoOfFields, затем мы загружаем информацию с помощью функций таблицы.

```
//Iterate through the loaded tables
For t = 0 to NoOfTables() - 1

//Iterate through the fields of table
 For f = 1 to NoOfFields(TableName($(t)))
  Tables:
  Load
   TableName($(t)) as Table,
   TableNumber(TableName($(t))) as TableNo,
   NoOfRows(TableName($(t))) as TableRows,
   FieldName($(f),TableName($(t))) as Field,
   FieldNumber(FieldName($(f),TableName($(t))),TableName($(t))) as FieldNo
   Autogenerate 1;
 Next f
Next t;
```

Результирующая таблица Tables будет выглядеть так:

Load table

Table	TableNo	TableRows	Field	FieldNo
Characters	0	26	Alpha	1
Characters	0	26	Num	2
ASCII	1	191	Num	1
ASCII	1	191	AsciiAlpha	2
ASCII	1	191	AsciiNum	3

FieldName

Функция скрипта **FieldName** возвращает имя поля с указанным номером в ранее загруженной таблице. Если функция используется в операторе **LOAD**, то она не должна ссылаться на таблицу, загружаемую в настоящее время.

Синтаксис:

```
FieldName(field_number ,table_name)
```

Аргументы:

Аргументы

Аргумент	Описание
field_number	Номер поля, на которое должна быть ссылка.
table_name	Таблица с полем, на которое должна быть ссылка.

Пример:

```
LET a = FieldName(4,'tab1');
```

FieldNumber

Функция скрипта **FieldNumber** возвращает номер указанного поля в ранее загруженной таблице. Если функция используется в операторе **LOAD**, то она не должна ссылаться на таблицу, загружаемую в настоящее время.

Синтаксис:

```
FieldNumber(field_name ,table_name)
```

Аргументы:

Аргументы

Аргумент	Описание
field_name	Имя поля.
table_name	Имя таблицы с полем.

Если поле field_name не существует в элементе table_name, или элемент table_name не существует, функция возвращает 0.

Пример:

```
LET a = FieldNumber('Customer','tab1');
```

NoOfFields

Функция скрипта **NoOfFields** возвращает число полей в ранее загруженной таблице. Если функция используется в операторе **LOAD**, то она не должна ссылаться на таблицу, загружаемую в настоящее время.

Синтаксис:

```
NoOfFields(table_name)
```

Аргументы:

Аргументы

Аргумент	Описание
table_name	Имя таблицы.

Пример:

```
LET a = NoOfFields('tab1');
```

NoOfRows

Функция скрипта **NoOfRows** возвращает число строк (записей) в ранее загруженной таблице. Если функция используется в операторе **LOAD**, то она не должна ссылаться на таблицу, загружаемую в настоящее время.

Синтаксис:

```
NoOfRows(table_name)
```

Аргументы:

Аргументы

Аргумент	Описание
table_name	Имя таблицы.

Пример:

```
LET a = NoOfRows('tab1');
```

5.27 Тригонометрические и гиперболические функции

В этом разделе описаны функции для выполнения тригонометрических и гиперболических функций. Во всех функциях аргументы являются выражениями, определяемыми по углам, измеренным в радианах, где элемент **x** должен интерпретироваться как действительное число.

Все углы измеряются в радианах.

Все функции можно использовать как в скрипте загрузки данных, так и в выражениях диаграмм.

cos

Косинус **x**. Результат находится в интервале от -1 до +1.

```
cos( x )
```

acos

Арккосинус **x**. Функция определяется, только если -1≤**x**≤1. Результат находится в интервале от 0 до π.

```
acos( x )
```

sin

Синус **x**. Результат находится в интервале от -1 до +1.

```
sin( x )
```

asin

Арксинус **x**. Функция определяется, только если -1≤**x**≤1. Результат находится в интервале от -π/2 до π/2.

```
asin( x )
```

tan

Тангенс **x**. Результат – действительное число.

```
tan( x )
```

atan

Арктангенс **x**. Результат находится в интервале от -π/2 до π/2.

```
atan( x )
```

atan2

Двухмерное представление функции арктангенса. Возвращает угол между началом координат и точкой с координатами **x** и **y**. Результат находится в интервале от -π до +π.

```
atan2( y,x )
```

cosh

Гиперболический косинус **x**. Результат – положительное действительное число.

```
cosh( x )
```

sinh

Гиперболический синус **x**. Результат – действительное число.

```
sinh( x )
```

tanh

Гиперболический тангенс **x**. Результат – действительное число.

```
tanh( x )
```

acosh

Гиперболический арккосинус **x** Результат – положительное действительное число.

```
acosh( x )
```

asinh

Гиперболический арксинус **x** Результат – действительное число.

```
asinh( x )
```

atanh

Гиперболический арктангенс **x**. Результат – действительное число.

```
atanh( x )
```

Примеры:

В следующем коде скрипта загружается таблица образца, а затем загружается таблица, содержащая вычисляемые тригонометрические и гиперболические операции со значениями.

```
SampleData:
LOAD * Inline
[Value
-1
0
1];

Results:
Load *,
cos(Value),
acos(Value),
sin(Value),
asin(Value),
tan(Value),
atan(Value),
atan2(Value, Value),
cosh(Value),
sinh(Value),
tanh(Value)
RESIDENT SampleData;

Drop Table SampleData;
```

6 Ограничение доступа к файловой системе

В целях безопасности Qlik Sense в стандартном режиме не поддерживает пути в скрипте загрузки данных или функции и переменные, предоставляющие доступ к файловой системе.

Однако поскольку пути файловой системы поддерживались в QlikView, можно отключить стандартный режим и использовать устаревший режим, чтобы повторно использовать скрипты загрузки QlikView.

Отключение стандартного режима создает угрозу безопасности из-за предоставления доступа к файловой системе.

Отключение стандартного режима (page 877)

6.1 Аспекты безопасности при подключении к файлу на основе подключений данных ODBC и OLE DB

Подключения к данным ODBC и OLE DB с помощью драйверов на основе файлов покажут путь к подключенному файлу данных в строке подключения. Путь может быть показан во время редактирования подключения, в диалоговом окне выборки данных или в специальных запросах SQL. Это применяется как в стандартном, так и в устаревшем режимах.

Если необходимо показать путь к файлу данных, рекомендуется подключиться к файлу данных с помощью подключения к данным папки, если это возможно.

6.2 Ограничения в стандартном режиме

В стандартном режиме некоторые операторы, переменные и функции нельзя использовать либо существуют ограничения на их использование. Использование неподдерживаемых операторов в скрипте загрузки данных приводит к возникновению ошибки при запуске этого скрипта. Сообщение об ошибке можно найти в файле журнала скрипта. Использование неподдерживаемых переменных и функций не приводит к возникновению ошибки или записи в файле журнала. Вместо этого функция возвращает значение NULL.

При редактировании скрипта загрузки данных неподдерживаемые переменные, операторы или функции никак не обозначаются.

Системные переменные

Системные переменные

Переменная	Стандартный режим	Устаревший режим	Определение
Floppy	Не поддерживается	Поддерживается	Возвращает буквенное обозначение первого найденного дисковода гибких дисков, обычно *a:*.
CD	Не поддерживается	Поддерживается	Возвращает буквенное обозначение первого найденного дисковода CD-ROM. Если дисковод CD-ROM не найден, возвращается *c:*.
QvPath	Не поддерживается	Поддерживается	Возвращает строку обзора в выполняемый модуль Qlik Sense.
QvRoot	Не поддерживается	Поддерживается	Возвращает корневой каталог выполняемого модуля Qlik Sense.
QvWorkPath	Не поддерживается	Поддерживается	Возвращает строку обзора в текущее приложение Qlik Sense.
QvWorkRoot	Не поддерживается	Поддерживается	Возвращает корневой каталог текущего приложения Qlik Sense.
WinPath	Не поддерживается	Поддерживается	Возвращает строку обзора в Windows.
WinRoot	Не поддерживается	Поддерживается	Возвращает корневой каталог Windows.

Переменная	Стандартный режим	Устаревший режим	Определение
$(include=...)	Поддерживаемый ввод: путь с использованием подключения к библиотеке	Поддерживаемый ввод: путь с использованием подключения к библиотеке или файловой системы	Переменная **Include/Must_Include** указывает файл, содержащий текст, который необходимо включить в скрипт и который рассматривается в качестве кода скрипта. Она не используется для добавления данных. Можно сохранять часть кода скрипта в отдельный текстовый файл и использовать его в разных приложениях. Эта переменная определяется пользователем.

Обычные операторы скриптов

Обычные операторы скриптов

Оператор	Стандартный режим	Устаревший режим	Определение
Binary	Поддерживаемый ввод: путь с использованием подключения к библиотеке	Поддерживаемый ввод: путь с использованием подключения к библиотеке или файловой системы	Оператор **binary** используется для загрузки данных из другого приложения.

Оператор	Стандартный режим	Устаревший режим	Определение
Connect	Поддерживаемый ввод: путь с использованием подключения к библиотеке	Поддерживаемый ввод: путь с использованием подключения к библиотеке или файловой системы	Оператор **CONNECT** используется для определения доступа программы Qlik Sense к общей базе данных с помощью интерфейса OLE DB/ODBC. Для интерфейса ODBC необходимо сначала задать источник данных с помощью администратора ODBC.
Directory	Поддерживаемый ввод: путь с использованием подключения к библиотеке	Поддерживаемый ввод: путь с использованием подключения к библиотеке или файловой системы	Оператор **Directory** задает каталог, в котором будет выполняться поиск файлов данных в последующих операторах **LOAD** до создания нового оператора **Directory**.
Execute	Не поддерживается	Поддерживаемый ввод: путь с использованием подключения к библиотеке или файловой системы	Оператор **Execute** используется для запуска других программ в ходе загрузки данных Qlik Sense. Например, для выполнения необходимых преобразований.

Оператор	Стандартный режим	Устаревший режим	Определение
LOAD from ...	Поддерживаемый ввод: путь с использованием подключения к библиотеке	Поддерживаемый ввод: путь с использованием подключения к библиотеке или файловой системы	Оператор **LOAD** загружает поля из файла, из определенных в скрипте данных, из ранее загруженной таблицы, из веб-страницы, из результата последующего оператора **SELECT** или путем создания данных.
Store into ...	Поддерживаемый ввод: путь с использованием подключения к библиотеке	Поддерживаемый ввод: путь с использованием подключения к библиотеке или файловой системы	Оператор **Store** создает файл QVD, CSV или text.

Операторы управления скриптом

Операторы управления скриптом

Оператор	Стандартный режим	Устаревший режим	Определение
For each... filelist mask/dirlist mask	Поддерживаемый ввод: путь с использованием подключения к библиотеке Выходные данные: подключение к библиотеке	Поддерживаемый ввод: путь с использованием подключения к библиотеке или файловой системы Выходные данные: путь с использованием подключения к библиотеке или файловой системы, в зависимости от ввода	Синтаксис filelist mask создает разделенный запятыми список всех файлов в текущем каталоге, соответствующих маске имени файла **filelist mask**. Синтаксис **dirlist mask** создает разделенный запятыми список всех каталогов в текущем каталоге, соответствующих маске имени каталога.

Функции файлов

Функции файлов

Функция	Стандартный режим	Устаревший режим	Определение
Attribute()	Поддерживаемый ввод: путь с использованием подключения к библиотеке	Поддерживаемый ввод: путь с использованием подключения к библиотеке или файловой системы	Возвращает значение мета-тегов различных медиафайлов в виде текста.
ConnectString()	Выходные данные: имя подключения к библиотеке	Имя подключения к библиотеке или фактическое подключение, в зависимости от ввода	Возвращает активную строку подключения для подключений ODBC или OLE DB.
FileDir()	Выходные данные: подключение к библиотеке	Выходные данные: путь с использованием подключения к библиотеке или файловой системы, в зависимости от ввода	Функция **FileDir** возвращает строку, содержащую путь к каталогу табличного файла, читаемого в текущий момент.
FilePath()	Выходные данные: подключение к библиотеке	Выходные данные: путь с использованием подключения к библиотеке или файловой системы, в зависимости от ввода	Функция **FilePath** возвращает строку, содержащую полный путь табличного файла, читаемого в текущий момент.
FileSize()	Поддерживаемый ввод: путь с использованием подключения к библиотеке	Поддерживаемый ввод: путь с использованием подключения к библиотеке или файловой системы	Функция **FileSize** возвращает целое, содержащее размер в байтах файла filename, или, если не указан файл filename, табличного файла, читаемого в текущий момент.

Функция	Стандартный режим	Устаревший режим	Определение
FileTime()	Поддерживаемый ввод: путь с использованием подключения к библиотеке	Поддерживаемый ввод: путь с использованием подключения к библиотеке или файловой системы	Функция **FileTime** возвращает метку времени для даты и времени последнего исправления файла filename. Если не указан файл в поле filename, функция ссылается на табличный файл, читаемый в текущий момент.
GetFolderPath()	Не поддерживается	Выходные данные: абсолютный путь	Функция **GetFolderPath** возвращает значение функции Microsoft Windows *SHGetFolderPath*. Данная функция берет в качестве значения ввода имя папки Microsoft Windows и возвращает полный путь папки.
QvdCreateTime()	Поддерживаемый ввод: путь с использованием подключения к библиотеке	Поддерживаемый ввод: путь с использованием подключения к библиотеке или файловой системы	Эта функция скрипта возвращает метку времени верхнего колонтитула XML из файла QVD при его наличии, в противном случае она возвращает значение NULL.
QvdFieldName()	Поддерживаемый ввод: путь с использованием подключения к библиотеке	Поддерживаемый ввод: путь с использованием подключения к библиотеке или файловой системы	Эта функция скрипта возвращает имя числа поля **fieldno** в файле QVD. Если поле не существует, возвращается значение NULL.

Функция	Стандартный режим	Устаревший режим	Определение
QvdNoOfFields()	Поддерживаемый ввод: путь с использованием подключения к библиотеке	Поддерживаемый ввод: путь с использованием подключения к библиотеке или файловой системы	Эта функция скрипта возвращает число полей в файле QVD.
QvdNoOfRecords()	Поддерживаемый ввод: путь с использованием подключения к библиотеке	Поддерживаемый ввод: путь с использованием подключения к библиотеке или файловой системы	Эта функция скрипта возвращает число записей, находящихся в настоящее время в файле QVD.
QvdTableName()	Поддерживаемый ввод: путь с использованием подключения к библиотеке	Поддерживаемый ввод: путь с использованием подключения к библиотеке или файловой системы	Эта функция скрипта возвращает имя таблицы, хранящейся в файле QVD.

Системные функции

Системные функции

Функция	Стандартный режим	Устаревший режим	Определение
DocumentPath()	Не поддерживается	Выходные данные: абсолютный путь	Эта функция возвращает строку, содержащую полный путь к текущему приложению Qlik Sense.
GetRegistryString()	Не поддерживается	Поддерживается	Возвращает значение именованного раздела реестра с указанным путем реестра. Эта функция также может использоваться в диаграммах и скриптах.

6.3 Отключение стандартного режима

Можно отключить стандартный режим или, другими словами, установить устаревший режим, чтобы повторно использовать скрипты загрузки QlikView, которые ссылаются на абсолютные или относительные пути файлов, а также подключения к библиотекам.

Отключение стандартного режима создает угрозу безопасности из-за предоставления доступа к файловой системе.

Qlik Sense

Для Qlik Sense стандартный режим можно отключить в QMC, используя свойство **Стандартный режим**.

Qlik Sense Desktop

В Qlik Sense Desktop стандартный/устаревший режим можно установить в файле *Settings.ini*.

Если для установки Qlik Sense Desktop использовалось расположение установки по умолчанию, файл *Settings.ini* находится в расположении *C:\Users\{user}\Documents\Qlik\Sense\Settings.ini*. Если для установки Qlik Sense Desktop использовалась выбранная папка, файл *Settings.ini* находится в папке пути установки *Engine*.

Выполните следующие действия.

1. Откройте файл *Settings.ini* в текстовом редакторе.
2. Измените *StandardReload=1* на *StandardReload=0*.
3. Сохраните файл и запустите Qlik Sense Desktop.

Теперь Qlik Sense Desktop работает в устаревшем режиме.

Параметры

Для StandardReload доступны следующие параметры:

- 1 (стандартный режим)
- 0 (устаревший режим)

7 Функции и операторы QlikView, не поддерживаемые в Qlik Sense

Большинство функций и операторов, которые можно использовать в скриптах загрузки QlikView и выражениях диаграмм, также поддерживаются в Qlik Sense, но есть несколько исключений.

7.1 Операторы скрипта, не поддерживаемые в Qlik Sense

Операторы скрипта QlikView, не поддерживаемые в Qlik Sense

Оператор	Комментарии
Command	Используйте **SQL**.
InputField	

7.2 Функции, не поддерживаемые в Qlik Sense

В этом списке представлены функции скрипта и диаграммы QlikView, которые не поддерживаются в Qlik Sense.

- **GetCurrentField**
- **GetExtendedProperty**
- **Input**
- **InputAvg**
- **InputSum**
- **MsgBox**
- **NoOfReports**
- **ReportComment**
- **ReportId**
- **ReportName**
- **ReportNumber**

7.3 Префиксы, не поддерживаемые в Qlik Sense

В этом списке представлены префиксы QlikView, которые не поддерживаются в Qlik Sense.

- **Bundle**
- **Image_Size**
- **Info**

8 Функции и операторы, не рекомендуемые в Qlik Sense

Большинство функций и операторов можно использовать в скриптах загрузки QlikView, выражения диаграмм также поддерживаются в Qlik Sense, но некоторые из них не рекомендуется использовать в Qlik Sense. Существуют также функции и операторы, доступные в предыдущих версиях Qlik Sense, которые были исключены.

В целях сравнения они будут работать согласно своему предназначению, но мы рекомендуем обновить код согласно рекомендациям в данном разделе, поскольку в следующих версиях они могут быть удалены.

8.1 Операторы скрипта, не рекомендуемые в Qlik Sense

В этой таблице содержатся операторы скрипта, которые не рекомендуются для использования в Qlik Sense.

Операторы скрипта, которые не рекомендуются

Оператор	Рекомендация
Command	Используйте **SQL**.
CustomConnect	Используйте **Custom Connect**.

8.2 Параметры оператора скрипта, не рекомендуемые в Qlik Sense

В этой таблице описаны параметры оператора скрипта, которые не рекомендуются для использования в Qlik Sense.

Параметры оператора скрипта, которые не рекомендуются

Оператор	Параметры
Buffer	Используйте **Incremental** вместо: • **Inc** (не рекомендуется) • **Incr** (не рекомендуется)

Оператор	Параметры
LOAD	Следующие ключевые слова параметра создаются мастерами трансформации файла QlikView. При загрузке данных функциональность сохраняется, но Qlik Sense не обеспечивает поддержку/мастеров для создания оператора с этими параметрами: • Bottom • Cellvalue • Col • Colmatch • Colsplit • Colxtr • Compound • Contain • Equal • Every • Expand • Filters • Intarray • Interpret • Length • Longer • Numerical • Pos • Remove • Rotate • Row • Rowcnd • Shorter • Start • Strcnd • Top • Transpose • Unwrap • XML: XMLSAX and Pattern is Path

8.3 Функции, не рекомендуемые в Qlik Sense

В этой таблице описаны функции скрипта и диаграммы, которые не рекомендуются для использования в Qlik Sense.

Функции, которые не рекомендуются

Функция	Рекомендация
NumAvg **NumCount** **NumMax** **NumMin** **NumSum**	Используйте функции над выборкой. *Функции над выборкой (page 743)*
Color() **QliktechBlue** **QliktechGray**	Используйте другие функции цвета. Для получения этих цветов **QliktechBlue()** можно заменить **RGB(8, 18, 90)**, а **QliktechGray** можно заменить **RGB(158, 148, 137)**. *Функции цвета (page 443)*
QlikViewVersion	Используйте **EngineVersion**. *EngineVersion (page 861)*
ProductVersion	Используйте **EngineVersion**. *EngineVersion (page 861)*
QVUser	
Year2Date	Используйте **YearToDate**.
Vrank	Используйте **Rank**.
WildMatch5	Используйте **WildMatch**.

Классификатор ALL

В QlikView классификатор **ALL** можно указывать перед выражением. Это эквивалентно использованию **{1} TOTAL**. В этом случае вычисление будет выполнено для всех значений поля в документе, измерения диаграммы и текущие выборки будут проигнорированы. Всегда возвращается одинаковое значение независимо от логического состояния документа. При использовании классификатора **ALL** выражение множества не может использоваться, поскольку классификатор **ALL** уже определяет множество. По причинам совместимости классификатор **ALL** работает в данной версии Qlik Sense, но может быть удален в следующих версиях.