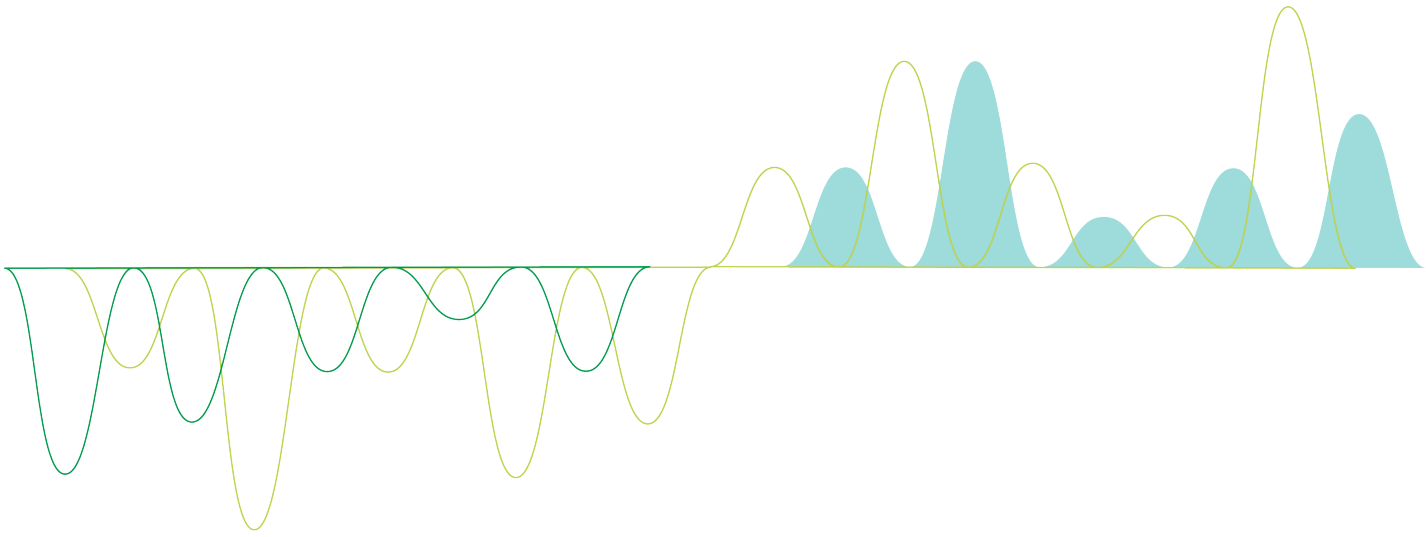
Синтаксис скрипта и функции диаграммы

Qlik Sense®
November 2022
© QlikTech International AB, 1993-2023. Все права защищены.



© 2023 QlikTech International AB. All rights reserved. All company and/or product names may be trade names, trademarks and/or registered trademarks of the respective owners with which they are associated.

Contents

1 Что такое Qlik Sense? 16
1.1 Что можно сделать с помощью программы Qlik Sense? 16
1.2 Как работает программа Qlik Sense? 16
Модель приложения 16
Ассоциативная работа 16
Совместная работа и мобильность 17
1.3 Как развернуть программу Qlik Sense? 17
Qlik Sense Desktop 17
Qlik Sense Enterprise 17
1.4 Как осуществлять контроль и управление сайтом Qlik Sense 17
1.5 Расширение возможностей Qlik Sense и адаптация под ваши требования 17
Построение расширений и гибридных веб-приложений 17
Построение клиентов 18
Построение инструментов сервера 18
Подключение к другим источникам данных 18
2 Обзор синтаксиса скрипта 19
2.1 Введение в синтаксис скрипта 19
2.2 Что такое форма Backus-Naur? 19
2 Операторы и ключевые слова скрипта 21
2.3 Операторы управления скриптом 21
Обзор операторов управления скриптом 21
Call 23
Do..loop 24
End 25
Exit 25
Exit script 25
For..next 26
For each..next 27
If..then..elseif..else..end if 30
Next 31
Sub..end sub 31
Switch..case..default..end switch 33
To 33
2.4 Префиксы скрипта 34
Обзор префиксов скрипта 34
Add 38
Buffer 40
Concatenate 42
Crosstable 48
First 57
Generic 60
Hierarchy 66
HierarchyBelongsTo 68
Inner 70
IntervalMatch 71
Join 74
Keep 85

Left ... 86
Сопоставление ... 87
Merge ... 88
NoConcatenate ... 93
Only ... 102
Outer ... 102
Частичная перезагрузка ... 103
Replace ... 106
Right ... 108
Sample ... 109
Semantic ... 112
Unless ... 116
When ... 122
2.5 Обычные операторы скриптов ... 129
Обзор обычных операторов скриптов ... 129
Alias ... 135
AutoNumber ... 136
Binary ... 139
Comment field ... 141
Comment table ... 141
Connect ... 142
Declare ... 144
Derive ... 147
Directory ... 147
Disconnect ... 148
Drop ... 149
Drop table ... 150
Execute ... 151
Field/Fields ... 152
FlushLog ... 152
Force ... 152
From ... 154
Load ... 154
Let ... 173
Loosen Table ... 173
Map ... 174
NullAsNull ... 175
NullAsValue ... 176
Qualify ... 176
Rem ... 177
Rename ... 178
Search ... 180
Section ... 181
Select ... 181
Set ... 184
Sleep ... 185
SQL ... 185
SQLColumns ... 186

SQLTables 187
SQLTypes 187
Star 188
Store 190
Table/Tables 192
Tag 193
Trace 193
Unmap 194
Unqualify 195
Untag 195
2.6 Рабочий каталог 196
Рабочий каталог Qlik Sense Desktop 196
Рабочий каталог Qlik Sense 196
2 Работа с переменными в редакторе загрузки данных 197
2.7 Обзор 197
2.8 Определение переменной 197
2.9 Удаление переменной 198
2.10 Загрузка значения переменной в качестве значения поля 198
2.11 Вычисление переменной 198
2.12 Системные переменные 199
Обзор системных переменных 200
CreateSearchIndexOnReload 202
HidePrefix 203
HideSuffix 203
Include 204
OpenUrlTimeout 205
StripComments 205
Verbatim 206
2.13 Переменные обработки значений 206
Обзор значений, обрабатывающих переменные 206
NullDisplay 207
NullInterpret 207
NullValue 207
OtherSymbol 208
2.14 Переменные интерпретации числа 208
Форматирование валюты 208
Форматирование чисел 209
Форматирование времени 209
BrokenWeeks 211
DateFormat 211
DayNames 218
DecimalSep 222
FirstWeekDay 225
LongDayNames 229
LongMonthNames 232
MoneyDecimalSep 236
MoneyFormat 240

MoneyThousandSep 245
MonthNames 249
NumericalAbbreviation 255
ReferenceDay 256
ThousandSep 260
TimeFormat 267
TimestampFormat 267
2.15 Переменные Direct Discovery 270
Системные переменные Direct Discovery 270
Переменные чередования запросов Teradata 271
Direct DiscoveryСимвольные переменные 272
Переменные интерпретации числа Direct Discovery 273
2.16 Ошибка переменных 274
Обзор ошибок переменных 274
ErrorMode 275
ScriptError 275
ScriptErrorCount 276
ScriptErrorList 277
2 Выражения скрипта 278
3 Выражения диаграммы 279
3.1 Определение объема агрегирования 279
3.2 Анализ множеств 282
Выражения множества 282
Примеры 283
Натуральные множества 283
Идентификаторы множества 286
Операторы множеств 287
Модификаторы множества 288
Внутренние и внешние выражения множества 311
Учебное пособие – создание выражения множества 314
Синтаксис выражений множества 323
3.3 Общий синтаксис выражений диаграммы 323
3.4 Общий синтаксис для агрегирования 324
4 Операторы 325
4.1 Побитовые операторы 325
4.2 Логические операторы 326
4.3 Числовые операторы 326
4.4 Реляционные операторы 327
4.5 Строковые операторы 329
& 329
like 329
5 Функции скрипта и диаграммы 330
5.1 Аналитические подключения для серверных расширений (SSE) 330
5.2 Функции агрегирования 330
Использование функций агрегирования в скрипте загрузки данных 331
Использование функций агрегирования в выражениях диаграмм 331

Как вычисляются агрегирования 331
Агрегирование ключевых полей 332
Базовые функции агрегирования 333
Функции агрегирования счетчика 357
Функции финансового агрегирования 376
Функции статистического агрегирования 399
Статистические тестовые функции 470
Строковые функции агрегирования 538
Функции синтетических измерений 551
Вложенные агрегирования 554
5.3 Aggr – функция диаграммы 555
Примеры: Выражения диаграммы с использованием Aggr 557
5.4 Функции цвета 560
Предопределенные функции цвета 562
ARGB 563
RGB 564
HSL 566
5.5 Условные функции 566
Обзор условных функций 567
alt 568
class 569
coalesce 570
if 571
match 574
mixmatch 577
pick 580
wildmatch 581
5.6 Функции счетчика 584
Обзор функций счетчика 584
autonumber 585
autonumberhash128 587
autonumberhash256 589
IterNo 592
RecNo 592
RowNo 594
RowNo – функция диаграммы 595
5.7 Функции даты и времени 597
Обзор функций даты и времени 598
addmonths 607
addyears 617
age 625
converttolocaltime 626
day 629
dayend 636
daylightsaving 644
dayname 644
daynumberofquarter 646
daynumberofyear 653

daystart 659
firstworkdate 667
GMT 668
hour 673
inday 676
indaytotime 685
inlunarweek 695
inlunarweektodate 707
inmonth 719
inmonths 727
inmonthstodate 741
inmonthtodate 755
inquarter 765
inquartertodate 778
inweek 791
inweektodate 808
inyear 822
inyeartodate 835
lastworkdate 848
localtime 858
lunarweekend 859
lunarweekname 872
lunarweekstart 884
makedate 897
maketime 903
makeweekdate 910
minute 918
month 924
monthend 930
monthname 940
monthsend 948
monthsname 961
monthsstart 975
monthstart 989
networkdays 999
now 1009
quarterend 1016
quartername 1029
quarterstart 1041
second 1054
setdateyear 1059
setdateyearmonth 1061
timezone 1063
today 1063
UTC 1070
week 1070
weekday 1088
weekend 1097

weekname 1109
weekstart 1123
weekyear 1135
year 1145
yearend 1152
yearname 1164
yearstart 1177
yeartodate 1189
5.8 Экспоненциальные и логарифмические функции 1205
5.9 Функции поля 1206
Функции счетчика 1206
Функции поля и выборки 1207
GetAlternativeCount – функция диаграммы 1207
GetCurrentSelections – функция диаграммы 1209
GetExcludedCount – функция диаграммы 1210
GetFieldSelections – функция диаграммы 1212
GetNotSelectedCount – функция диаграммы 1213
GetObjectDimension – функция диаграммы 1214
GetObjectField – функция диаграммы 1215
GetObjectMeasure – функция диаграммы 1216
GetPossibleCount – функция диаграммы 1216
GetSelectedCount – функция диаграммы 1218
5.10 Функции файлов 1219
Обзор функций файла 1219
Attribute 1221
ConnectString 1228
FileBaseName 1229
FileDir 1229
FileExtension 1230
FileName 1230
FilePath 1230
FileSize 1231
FileTime 1232
GetFolderPath 1233
QvdCreateTime 1234
QvdFieldName 1235
QvdNoOfFields 1235
QvdNoOfRecords 1236
QvdTableName 1237
5.11 Финансовые функции 1238
Обзор финансовых функций 1239
BlackAndSchole 1239
FV 1240
nPer 1241
Pmt 1242
PV 1243
Rate 1244
5.12 Функции форматирования 1245

Обзор функций форматирования 1246
ApplyCodepage 1247
Date 1248
Dual 1250
Interval 1251
Money 1252
Num 1254
Time 1256
Timestamp 1258
5.13 Общие числовые функции 1259
Обзор общих числовых функций 1259
Функции сочетаний и перестановок 1260
Функции Modulo 1260
Функции четности 1260
Функции округления 1261
BitCount 1261
Ceil 1262
Combin 1263
Div 1263
Even 1264
Fabs 1264
Fact 1265
Floor 1265
Fmod 1266
Frac 1267
Mod 1268
Odd 1269
Permut 1269
Round 1270
Sign 1272
5.14 Геопространственные функции 1272
Обзор геопространственных функций 1273
GeoAggrGeometry 1274
GeoBoundingBox 1275
GeoCountVertex 1276
GeoGetBoundingBox 1277
GeoGetPolygonCenter 1277
GeoInvProjectGeometry 1278
GeoMakePoint 1279
GeoProject 1279
GeoProjectGeometry 1280
GeoReduceGeometry 1281
5.15 Функции интерпретации 1282
Обзор функций интерпретации 1283
Date# 1284
Interval# 1285
Money# 1286
Num# 1287

Text 1288
Time# 1288
Timestamp# 1290
5.16 Функции между записями 1291
Функции строки 1291
Функции столбца 1292
Функции поля 1293
Функции сводной таблицы 1293
Функции между записями в скрипте загрузки данных 1294
Above – функция диаграммы 1295
Below – функция диаграммы 1300
Bottom – функция диаграммы 1304
Column – функция диаграммы 1308
Dimensionality – функция диаграммы 1310
Exists 1312
FieldIndex 1315
FieldValue 1317
FieldValueCount 1318
LookUp 1320
NoOfRows – функция диаграммы 1322
Peek 1324
Previous 1330
Top – функция диаграммы 1331
SecondaryDimensionality – функция диаграммы 1336
After – функция диаграммы 1336
Before – функция диаграммы 1337
First – функция диаграммы 1339
Last – функция диаграммы 1340
ColumnNo – функция диаграммы 1341
NoOfColumns – функция диаграммы 1342
5.17 Логические функции 1342
5.18 Функции сопоставления 1343
Обзор функций сопоставления 1343
ApplyMap 1344
MapSubstring 1346
5.19 Математические функции 1347
5.20 Функции NULL 1348
Обзор функций NULL 1348
EmptyIsNull 1349
IsNull 1349
NULL 1350
5.21 Функции над выборкой 1351
Базовые функции над выборкой 1351
Функции над выборкой счетчика 1352
Статистические функции над выборкой 1353
Финансовые функции над выборкой 1353
RangeAvg 1354
RangeCorrel 1356

RangeCount ... 1359
RangeFractile ... 1361
RangeIRR ... 1363
RangeKurtosis ... 1364
RangeMax ... 1365
RangeMaxString ... 1367
RangeMin ... 1369
RangeMinString ... 1371
RangeMissingCount ... 1372
RangeMode ... 1374
RangeNPV ... 1376
RangeNullCount ... 1377
RangeNumericCount ... 1379
RangeOnly ... 1380
RangeSkew ... 1381
RangeStdev ... 1382
RangeSum ... 1384
RangeTextCount ... 1386
RangeXIRR ... 1388
RangeXNPV ... 1389
5.22 Функции ранжирования и кластеризации ... 1390
Функции ранжирования в диаграммах ... 1390
Функции кластеризации в диаграммах ... 1390
Rank — функция диаграммы ... 1392
HRank — функция диаграммы ... 1396
Оптимизация методом k-средних: пример из реальной жизни ... 1398
KMeans2D — функция диаграммы ... 1408
KMeansND — функция диаграммы ... 1421
KMeansCentroid2D — функция диаграммы ... 1434
KMeansCentroidND — функция диаграммы ... 1436
5.23 Функции статистического распределения ... 1437
Обзор функций статистического распределения ... 1437
BetaDensity ... 1440
BetaDist ... 1440
BetaInv ... 1441
BinomDist ... 1441
BinomFrequency ... 1441
BinomInv ... 1442
ChiDensity ... 1442
ChiDist ... 1443
ChiInv ... 1443
FDensity ... 1444
FDist ... 1445
FInv ... 1445
GammaDensity ... 1446
GammaDist ... 1446
GammaInv ... 1447
NormDist ... 1447

NormInv 1448
PoissonDist 1449
PoissonFrequency 1450
PoissonInv 1450
TDensity 1450
TDist 1451
TInv 1452
5.24 Строковые функции 1452
Обзор строковых функций 1452
Capitalize 1456
Chr 1457
Evaluate 1457
FindOneOf 1458
Hash128 1459
Hash160 1460
Hash256 1460
Index 1461
IsJson 1462
JsonGet 1463
JsonSet 1464
KeepChar 1465
Left 1466
Len 1467
LevenshteinDist 1468
Lower 1469
LTrim 1470
Mid 1471
Ord 1472
PurgeChar 1472
Repeat 1473
Replace 1474
Right 1475
RTrim 1476
SubField 1476
SubStringCount 1479
TextBetween 1480
Trim 1481
Upper 1482
5.25 Системные функции 1482
Обзор системных функций 1483
EngineVersion 1485
IsPartialReload 1485
ProductVersion 1485
StateName – функция диаграммы 1485
5.26 Функции таблиц 1486
Обзор функций таблицы 1486
FieldName 1488
FieldNumber 1489

NoOfFields ... 1489
NoOfRows ... 1490
5.27 Тригонометрические и гиперболические функции ... 1490
6 Ограничение доступа к файловой системе ... 1493
6.1 Аспекты безопасности при подключении к файлу на основе подключений данных ODBC и OLE DB ... 1493
6.2 Ограничения в стандартном режиме ... 1493
Системные переменные ... 1494
Обычные операторы скриптов ... 1495
Операторы управления скриптом ... 1497
Функции файлов ... 1498
Системные функции ... 1500
6.3 Отключение стандартного режима ... 1501
Qlik Sense ... 1501
Qlik Sense Desktop ... 1501
6 Скрипты на уровне диаграммы ... 1502
6.4 Операторы управления ... 1502
Обзор операторов управления для модификаторов диаграммы ... 1502
Call ... 1504
Do..loop ... 1505
End ... 1506
Exit ... 1506
Exit script ... 1506
For..next ... 1507
For each..next ... 1508
If..then..elseif..else..end if ... 1511
Next ... 1512
Sub..end sub ... 1512
Switch..case..default..end switch ... 1514
To ... 1515
6.5 Префиксы ... 1515
Обзор префиксов для модификаторов диаграммы ... 1515
Add ... 1515
Replace ... 1516
6.6 Обычные операторы ... 1516
Обзор обычных операторов для модификаторов диаграммы ... 1516
Load ... 1517
Let ... 1522
Set ... 1523
Put ... 1523
HCValue ... 1524
7 Функции и операторы QlikView, не поддерживаемые в Qlik Sense ... 1526
7.1 Операторы скрипта, не поддерживаемые в Qlik Sense ... 1526
7.2 Функции, не поддерживаемые в Qlik Sense ... 1526
7.3 Префиксы, не поддерживаемые в Qlik Sense ... 1526
8 Функции и операторы, не рекомендуемые в Qlik Sense ... 1527
8.1 Операторы скрипта, не рекомендуемые в Qlik Sense ... 1527

8.2 Параметры оператора скрипта, не рекомендуемые в Qlik Sense 1527
8.3 Функции, не рекомендуемые в Qlik Sense 1528
Классификатор ALL 1529

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 загрузки **crosstable** используется для транспонирования структурированных данных «перекрестной таблицы» или «сводной таблицы». Данные, структурированные таким образом, обычно встречаются при работе с такими источниками, как электронные таблицы. Результатом и целью префикса загрузки **crosstable** является транспонирование таких структур в эквивалент обычной столбцовой таблицы, поскольку эта структура обычно лучше подходит для анализа в Qlik Sense.

```
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

Префикс загрузки Generic позволяет преобразовывать смоделированные данные типа «объект-атрибут-значение» (EAV) в традиционную нормализованную структуру реляционной таблицы. Моделирование EAV также называют «обобщенным моделированием данных» или «открытой схемой».

```
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 103) 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

`Concatenate` — это префикс загрузки скрипта, который позволяет добавлять набор данных в уже существующую в памяти таблицу. Он часто используется для добавления разных наборов транзакционных данных в одну центральную таблицу фактов или для создания общих справочных наборов данных определенного типа на основе нескольких источников. По функциональным возможностям он напоминает оператор SQL UNION.

Таблица, полученная в результате операции `concatenate`, будет содержать исходный набор данных с новыми строками данных, добавленными в конце этой таблицы. В исходной и целевой таблицах могут присутствовать несовпадающие поля. Когда поля не совпадают, результирующая таблица будет расширена путем объединения всех полей, имеющихся в исходной и целевой таблицах.

Синтаксис:

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

Аргументы

Аргумент	Описание
`tablename`	Имя существующей таблицы. Таблица с указанным именем будет целью для операции `Concatenate`, и все загруженные записи данных будут добавлены в эту таблицу. Если параметр `tablename` не задан, в качестве целевой будет использована таблица, загруженная последней перед этим оператором.
`loadstatement/selectstatement`	Аргумент `loadstatement/selectstatement`, который следует за аргументом `tablename`, будет объединен с указанной таблицей.

Региональные настройки

Если не указано иное, в примерах, приведенных в данном разделе, используется следующий формат даты: ММ/ДД/ГГГГ. Формат даты указан в операторе `SET DateFormat` скрипта загрузки данных. В вашей системе может быть установлен другой формат даты по умолчанию в зависимости от региональных настроек и других факторов. Можно изменить формат в примерах в соответствии с потребностями. Или можно изменить форматы в скрипте загрузки в соответствии с этими примерами.

Региональные настройки по умолчанию в приложениях основаны на системных региональных настройках компьютера или сервера, где установлено ПО Qlik Sense. Если на сервере Qlik Sense, к которому обращается пользователь, выбран шведский язык, то редактор загрузки данных будет использовать шведские региональные настройки для даты, времени и валюты. Эти параметры регионального формата не связаны с языком, отображаемым в интерфейсе пользователя Qlik Sense. Qlik Sense будет отображаться на языке, который используется в браузере.

Пример функции

Пример	Результат
`Concatenate (Transactions) Load …. ;`	Данные, загруженные при выполнении оператора load под префиксом `Concatenate`, будут добавлены в существующую в памяти таблицу под именем `Transactions` (предполагается, что таблица под именем `Transactions` уже загружена до этого этапа скрипта загрузки).

Пример 1. Добавление нескольких наборов данных в целевую таблицу с помощью префикса загрузки Concatenate

Скрипт загрузки и результаты

Обзор

В этом примере будут загружены два скрипта, один за другим.

- Первый скрипт загрузки содержит первоначальный набор данных с датами и суммами, который отправляется в таблицу под именем `Transactions`.
- Второй скрипт загрузки содержит следующее:
    - Второй набор данных, прибавляемый к исходному набору данных с помощью префикса `Concatenate`. Этот набор данных содержит дополнительное поле `type`, отсутствующее в начальном наборе данных.
    - Префикс `Concatenate`.

Откройте редактор загрузки данных и добавьте приведенный ниже скрипт загрузки на новую вкладку.

Первый скрипт загрузки

```
Transactions:
Load * Inline [

id, date, amount
3750, 08/30/2018, 23.56
3751, 09/07/2018, 556.31
3752, 09/16/2018, 5.75
3753, 09/22/2018, 125.00
3754, 09/22/2018, 484.21
3756, 09/22/2018, 59.18
3757, 09/23/2018, 177.42
];
```

Результаты

Загрузите данные и откройте лист. Создайте новую таблицу и добавьте эти поля как измерения:

- `id`
- `date`

- amount

Таблица результатов первого скрипта загрузки

id	date	amount
3750	08/30/2018	23.56
3751	09/07/2018	556.31
3752	09/16/2018	5.75
3753	09/22/2018	125.00
3754	09/22/2018	484.21
3756	09/22/2018	59.18
3757	09/23/2018	177.42

В таблице отображается первоначальный набор данных.

Второй скрипт загрузки

Откройте редактор загрузки данных и добавьте приведенный ниже скрипт загрузки.

```
Concatenate(Transactions)
Load * Inline [
id, date, amount, type
3758, 10/01/2018, 164.27, Internal
3759, 10/03/2018, 384.00, External
3760, 10/06/2018, 25.82, Internal
3761, 10/09/2018, 312.00, Internal
3762, 10/15/2018, 4.56, Internal
3763, 10/16/2018, 90.24, Internal
3764, 10/18/2018, 19.32, External
];
```

Результаты

Загрузите данные и перейдите на лист. Создайте это поле как измерение:

- type

Таблица результатов второго скрипта загрузки

id	date	amount	type
3750	08/30/2018	23.56	-
3751	09/07/2018	556.31	-
3752	09/16/2018	5.75	-
3753	09/22/2018	125.00	-

id	date	amount	type
3754	09/22/2018	484.21	-
3756	09/22/2018	59.18	-
3757	09/23/2018	177.42	-
3758	10/01/2018	164.27	Внутренняя
3759	10/03/2018	384.00	Внешний
3760	10/06/2018	25.82	Внутренняя
3761	10/09/2018	312.00	Внутренняя
3762	10/15/2018	4.56	Внутренняя
3763	10/16/2018	90.24	Внутренняя
3764	10/18/2018	19.32	Внешний

Обратите внимание на нулевые значения (null) в поле `type` для первых семи загруженных записей, где не был определен `type`.

Пример 2. Добавление нескольких наборов данных в целевую таблицу путем неявного объединения

Скрипт загрузки и результаты

Обзор

Типичным примером применения неявного добавления данных является загрузка нескольких файлов с идентичной структурой данных с целью их добавления в целевую таблицу.

Например, для этого можно использовать `wildcards` в именах файлов с применением такого синтаксиса, как:

```
myTable:
Load * from [myFile_*.qvd] (qvd);
```

или в циклах с применением таких конструкций, как:

```
for each file in filelist('myFile_*.qvd')

myTable:
Load * from [$(file)] (qvd);

next file
```

Неявное объединение происходит между любыми двумя таблицами, которые загружаются с одинаково названными полями, даже если они не определены друг за другом в скрипте. Это можно приводить к непреднамеренному добавлению данных в таблицы. Если не требуется добавлять второстепенную таблицу с идентичными полями таким образом, используйте префикс загрузки `NoConcatenate`*. Переименования таблицы с указанием другого тега имени таблицы недостаточно для предотвращения неявного объединения. Для получения дополнительной информации см. NoConcatenate (page 93).*

В этом примере будут загружены два скрипта, один за другим.

- Первый скрипт загрузки содержит первоначальный набор с четырьмя полями, который отправляется в таблицу под именем `Transactions`.
- Второй скрипт загрузки содержит набор данных с такими же полями, как в первом наборе данных.

Откройте редактор загрузки данных и добавьте приведенный ниже скрипт загрузки на новую вкладку.

Первый скрипт загрузки

```
Transactions:
Load * Inline [
id, date, amount, type
3758, 10/01/2018, 164.27, Internal
3759, 10/03/2018, 384.00, External
3760, 10/06/2018, 25.82, Internal
3761, 10/09/2018, 312.00, Internal
3762, 10/15/2018, 4.56, Internal
3763, 10/16/2018, 90.24, Internal
3764, 10/18/2018, 19.32, External
];
```

Результаты

Загрузите данные и откройте лист. Создайте новую таблицу и добавьте эти поля как измерения:

- `id`
- `date`
- `amount`
- `type`

Таблица результатов первого скрипта загрузки

id	date	type	amount
3758	10/01/2018	Внутренняя	164.27
3759	10/03/2018	Внешний	384.00

id	date	type	amount
3760	10/06/2018	Внутренняя	25.82
3761	10/09/2018	Внутренняя	312.00
3762	10/15/2018	Внутренняя	4.56
3763	10/16/2018	Внутренняя	90.24
3764	10/18/2018	Внешний	19.32

В таблице отображается первоначальный набор данных.

Второй скрипт загрузки

Откройте редактор загрузки данных и добавьте приведенный ниже скрипт загрузки.

```
Load * Inline [
id, date, amount, type
3765, 11/03/2018, 129.40, Internal
3766, 11/05/2018, 638.50, External
];
```

Результаты

Загрузите данные и откройте лист.

Таблица результатов второго скрипта загрузки

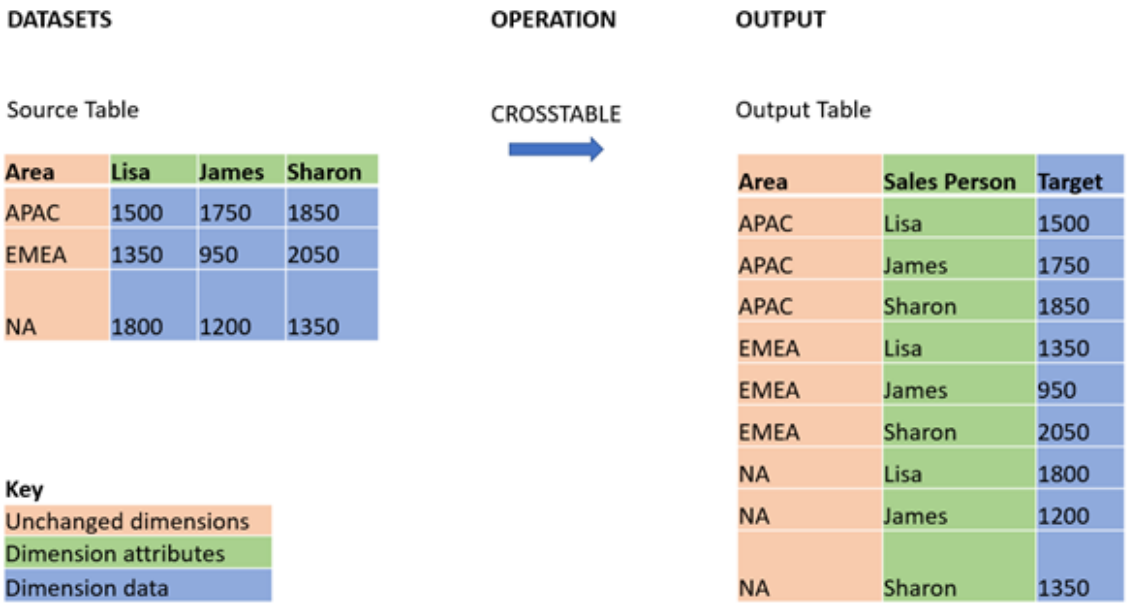
id	date	type	amount
3758	10/01/2018	Внутренняя	164.27
3759	10/03/2018	Внешний	384.00
3760	10/06/2018	Внутренняя	25.82
3761	10/09/2018	Внутренняя	312.00
3762	10/15/2018	Внутренняя	4.56
3763	10/16/2018	Внутренняя	90.24
3764	10/18/2018	Внешний	19.32
3765	11/03/2018	Внутренняя	129.40
3766	11/05/2018	Внешний	638.50

Выполнено неявное объединение второго набора данных с первоначальным, так как они имеют идентичные поля.

Crosstable

Префикс загрузки **crosstable** используется для транспонирования структурированных данных «перекрестной таблицы» или «сводной таблицы». Данные, структурированные таким образом, обычно встречаются при работе с такими источниками, как электронные таблицы. Результатом и целью префикса загрузки **crosstable** является транспонирование таких структур в эквивалент обычной столбцовой таблицы, поскольку эта структура обычно лучше подходит для анализа в Qlik Sense.

Пример данных, структурированных как перекрестная таблица, и эквивалентная структура после преобразования перекрестной таблицы



Синтаксис:

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

Аргументы

Аргумент	Описание
attribute field name	Желаемое имя выходного поля, описывающее горизонтально ориентированное измерение, которое необходимо преобразовать (строка заголовка).
data field name	Желаемое имя выходного поля, описывающее горизонтально ориентированные данные измерения, которое необходимо преобразовать (матрица значений данных под строкой заголовка).
n	Число полей-классификаторов, или неизмененных измерений, перед таблицей, которые следует преобразовать в общий формат. Значение по умолчанию – 1.

Эта функция скрипта связана со следующими функциями:

Связанные функции

Функция	Взаимодействие
Generic (page 60)	Префикс загрузки преобразования, который принимает структурированный набор данных «объект-атрибут-значение» и преобразует его в обычную структуру реляционной таблицы, распределяя каждый встречающийся атрибут в новое поле или столбец данных.

Пример 1. Преобразование сводных данных о продажах (простое)

Скрипты загрузки и результаты

Обзор

Откройте редактор загрузки данных и добавьте первый приведенный ниже скрипт загрузки на новую вкладку.

Первый скрипт загрузки содержит набор данных, к которому позднее будет применен префикс скрипта `crosstable`, при этом раздел, применяющий `crosstable`, закомментирован. Это означает, что для отключения этого раздела в скрипте загрузки использовался синтаксис комментария.

Второй скрипт загрузки такой же, как и первый, но с применением раскомментированного `crosstable` (для этого удаляется синтаксис комментария). Скрипты показаны таким образом, чтобы подчеркнуть пользу этой функции скрипта при преобразовании данных.

Первый скрипт загрузки (функция не применяется)

```
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;
```

Результаты

Загрузите данные и откройте лист. Создайте новую таблицу и добавьте эти поля как измерения:

- Product
- Jan 2021
- Feb 2021
- Mar 2021
- Apr 2021
- May 2021
- Jun 2021

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

Этот скрипт позволяет создать 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 – в таблице с тремя столбцами. В следующем разделе объясняется, как выполнить это преобразование перекрестной таблицы.

Второй скрипт загрузки (функция применяется)

Раскомментируйте скрипт, удалив //. Скрипт загрузки будет выглядеть так:

```
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;
```

Результаты

Загрузите данные и откройте лист. Создайте новую таблицу и добавьте эти поля как измерения:

- Product
- Month

- `Sales`

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

После применения префикса скрипта перекрестная таблица трансформируется в прямую таблицу, в которой один столбец будет для `Month`, а другой для `Sales`. Это улучшает восприятие данных.

Пример 2. Преобразование сводных данных о целевых объемах продаж в вертикальную структуру таблицы (промежуточный вариант)

Скрипт загрузки и выражение диаграммы

Обзор

Откройте редактор загрузки данных и добавьте приведенный ниже скрипт загрузки на новую вкладку.

Скрипт загрузки содержит следующее:

- Набор данных, загруженный в таблицу с именем Targets.
- Префикс загрузки `crosstable`, который переносит сводные данные об именах продавцов в отдельное поле с именем `Sales Person`.
- Связанные данные о целевых объемах продаж, сгруппированные в поле с именем `Target`.

Скрипт загрузки

```
SalesTargets:
CROSSTABLE([Sales Person],Target,1)
LOAD
*
INLINE [
Area, Lisa, James, Sharon
APAC, 1500, 1750, 1850
EMEA, 1350, 950, 2050
NA, 1800, 1200, 1350
];
```

Результаты

Загрузите данные и откройте лист. Создайте новую таблицу и добавьте эти поля как измерения:

- `Area`
- `Sales Person`

Добавьте эту меру:

```
=Sum(Target)
```

Результирующая таблица

Область	Менеджер по продажам	=Sum(Target)
APAC	James	1750
APAC	Lisa	1500
APAC	Sharon	1850
EMEA	James	950
EMEA	Lisa	1350
EMEA	Sharon	2050
NA	James	1200
NA	Lisa	1800
NA	Sharon	1350

Если требуется скопировать отображение данных в виде сводной входной таблицы, можно создать эквивалентную сводную таблицу на листе.

Выполните следующие действия.

1. Скопируйте и вставьте только что созданную таблицу на лист.
2. Перетащите объект диаграммы **Сводная таблица** над только что созданной копией таблицы. Выберите **Преобразовать**.
3. Щелкните ✔ **Изменение завершено**.
4. Перетащите поле `Sales Person` с полки вертикального столбца на полку горизонтального столбца.

В следующей таблице показаны данные в форме исходной таблицы, как они отображаются в Qlik Sense:

Исходная таблица результатов, как показано в Qlik Sense

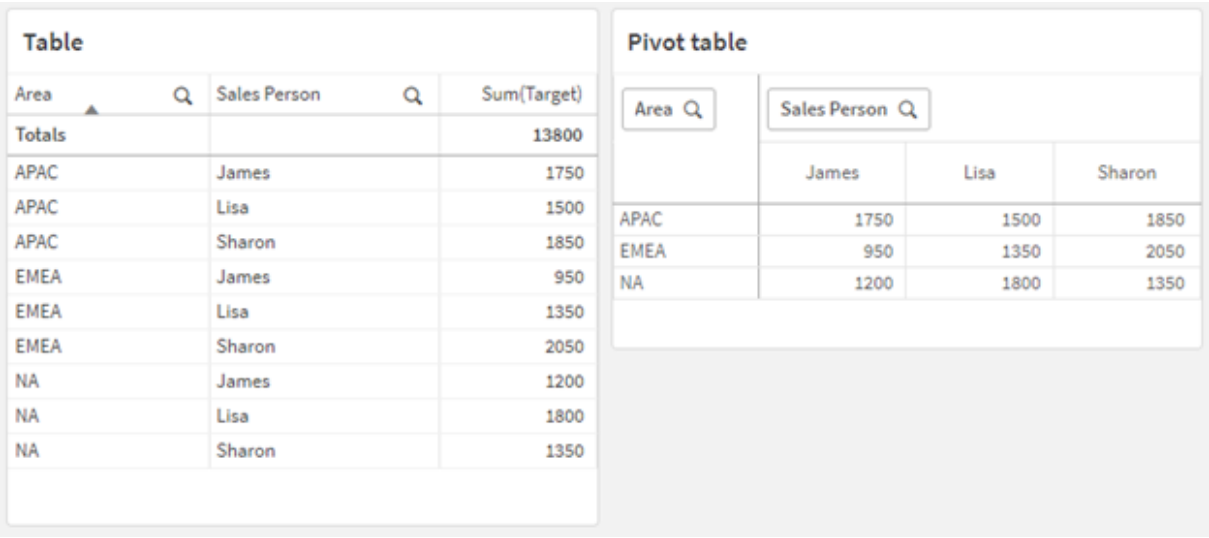
Область	Менеджер по продажам	=Sum(Target)
Итоги	-	13800
APAC	James	1750
APAC	Lisa	1500
APAC	Sharon	1850
EMEA	James	950
EMEA	Lisa	1350
EMEA	Sharon	2050
NA	James	1200
NA	Lisa	1800
NA	Sharon	1350

Эквивалентная сводная таблица выглядит примерно так, как показано ниже: столбцы для имени каждого продавца включены в более крупную строку для `Sales Person`:

Эквивалентная сводная таблица с горизонтальной ориентацией поля `Sales Person`

Область	James	Lisa	Sharon
APAC	1750	1500	1850
EMEA	950	1350	2050
NA	1350	1350	1350

Пример данных, отображаемых в виде таблицы, и эквивалентная сводная таблица с горизонтальной ориентацией поля `Sales Person`



Table

Area	Sales Person	Sum(Target)
Totals		13800
APAC	James	1750
APAC	Lisa	1500
APAC	Sharon	1850
EMEA	James	950
EMEA	Lisa	1350
EMEA	Sharon	2050
NA	James	1200
NA	Lisa	1800
NA	Sharon	1350

Pivot table

Area	Sales Person		
	James	Lisa	Sharon
APAC	1750	1500	1850
EMEA	950	1350	2050
NA	1200	1800	1350

Пример 3. Преобразование сводных данных о целевых объемах продаж в вертикальную структуру таблицы (расширенный вариант)

Скрипт загрузки и выражение диаграммы

Обзор

Откройте редактор загрузки данных и добавьте приведенный ниже скрипт загрузки на новую вкладку.

Скрипт загрузки содержит следующее:

- Набор данных, представляющий данные о продажах и целевых показателях, организованный по регионам и месяцам года. Набор данных загружается в таблицу под именем `SalesAndTargets`.
- Префикс загрузки `crosstable`. Он используется для переноса измерения `Month Year` из сводной области в специальное поле, а также для переноса матрицы продаж и целевых сумм в специальное поле с именем `Amount`.
- Преобразование поля `Month Year` из текста в правильную дату с использованием функции преобразования текста в дату `date#`. Это поле `Month Year`, преобразованное в дату, снова присоединяется к таблице `SalesAndTarget` с помощью префикса загрузки `Join`.

Скрипт загрузки

```
SalesAndTargets:
CROSSTABLE(MonthYearAsText,Amount,2)
LOAD
       *
INLINE [
Area   Type    Jan-22  Feb-22  Mar-22  Apr-22  May-22  Jun-22  Jul-22  Aug-22  Sep-22  Oct-22  Nov-22  Dec-22
```

```
APAC    Target  425     425     425     425     425     425     425     425     425     425     425     425
APAC    Actual  435     434     397     404     458     447     413     458     385     421     448     397
EMEA    Target  362.5   362.5   362.5   362.5   362.5   362.5   362.5   362.5   362.5   362.5   362.5   362.5
EMEA    Actual  363.5   359.5   337.5   361.5   341.5   337.5   379.5   352.5   327.5   337.5   360.5   334.5
NA      Target  375     375     375     375     375     375     375     375     375     375     375     375
NA      Actual  378     415     363     356     403     343     401     365     393     340     360     405
] (delimiter is '\t');

tmp:
LOAD DISTINCT MonthYearAsText,date#(MonthYearAsText,'MMM-YY') AS [Month Year]
RESIDENT SalesAndTargets;

JOIN (SalesAndTargets)
LOAD * RESIDENT tmp;

DROP TABLE tmp;
DROP FIELD MonthYearAsText;
```

Результаты

Загрузите данные и откройте лист. Создайте новую таблицу и добавьте эти поля как измерения:

- `Area`
- `Month Year`

Создайте следующую меру с меткой `Actual`:

```
=Sum({<Type={'Actual'}>} Amount)
```

Также создайте эту меру с меткой `Target`:

```
=Sum({<Type={'Target'}>} Amount)
```

Таблица результатов (обрезанная)

Область	Месяц года	Actual	Цель
APAC	Jan-22	435	425
APAC	Feb-22	434	425
APAC	Mar-22	397	425
APAC	Apr-22	404	425
APAC	May-22	458	425
APAC	Jun-22	447	425
APAC	Jul-22	413	425
APAC	Aug-22	458	425
APAC	Sep-22	385	425
APAC	Oct-22	421	425

Область	Месяц года	Actual	Цель
APAC	Nov-22	448	425
APAC	Dec-22	397	425
EMEA	Jan-22	363.5	362.5
EMEA	Feb-22	359.5	362.5

Если требуется скопировать отображение данных в виде сводной входной таблицы, можно создать эквивалентную сводную таблицу на листе.

Выполните следующие действия.

1. Скопируйте и вставьте только что созданную таблицу на лист.
2. Перетащите объект диаграммы **Сводная таблица** над только что созданной копией таблицы. Выберите **Преобразовать**.
3. Щелкните ✔ **Изменение завершено**.
4. Перетащите поле `Month Year` с полки вертикального столбца на полку горизонтального столбца.
5. Перетащите поле `Values` с полки горизонтального столбца на полку вертикального столбца.

В следующей таблице показаны данные в форме исходной таблицы, как они отображаются в Qlik Sense:

Исходная таблица результатов (обрезанная), как показано в Qlik Sense

Область	Month Year	Actual	Цель
Итоги	-	13812	13950
APAC	Jan-22	435	425
APAC	Feb-22	434	425
APAC	Mar-22	397	425
APAC	Apr-22	404	425
APAC	May-22	458	425
APAC	Jun-22	447	425
APAC	Jul-22	413	425
APAC	Aug-22	458	425
APAC	Sep-22	385	425
APAC	Oct-22	421	425
APAC	Nov-22	448	425

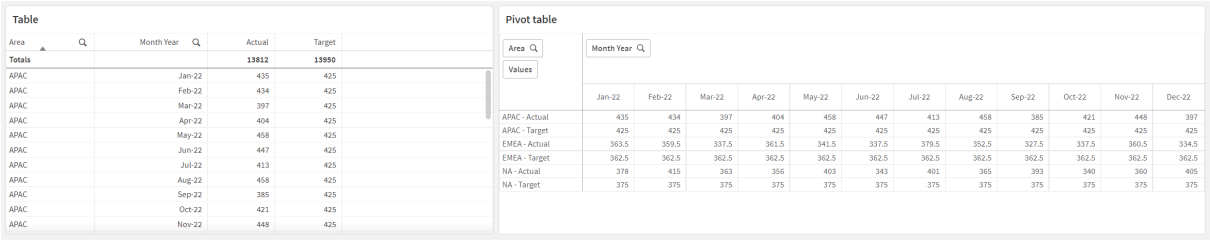
Область	Month Year	Actual	Цель
APAC	Dec-22	397	425
EMEA	Jan-22	363.5	362.5
EMEA	Feb-22	359.5	362.5

Эквивалентная сводная таблица выглядит примерно так, как показано ниже: столбцы для каждого месяца включены в более крупную строку для `Month Year`:

Эквивалентная сводная таблица (обрезанная) с горизонтальной ориентацией поля `Month Year`

Area (Values)	Jan-22	Feb-22	Mar-22	Apr-22	May-22	Jun-22	Jul-22	Aug-22	Sep-22	Oct-22	Nov-22	Dec-22
APAC - Actual	435	434	397	404	458	447	413	458	385	421	448	397
APAC - Target	425	425	425	425	425	425	425	425	425	425	425	425
EMEA - Actual	363.5	359.5	337.5	361.5	341.5	337.5	379.5	352.5	327.5	337.5	360.5	334.5
EMEA - Target	362.5	362.5	362.5	362.5	362.5	362.5	362.5	362.5	362.5	362.5	362.5	362.5
NA - Actual	378	415	363	356	403	343	401	365	393	340	360	405
NA - Target	375	375	375	375	375	375	375	375	375	375	375	375

Пример данных, отображаемых в виде таблицы, и эквивалентная сводная таблица с горизонтальной ориентацией поля `Month Year`



First

Префикс `First` оператора `LOAD` или `SELECT` (SQL) используется для загрузки заданного максимального числа записей из таблицы источника данных. Типичным примером использования префикса `First` является ситуация, когда требуется получить

небольшое подмножество записей из большого и (или) медленного этапа загрузки данных. Как только будет загружено определенное «n» количество записей, этап загрузки преждевременно завершается, а остальная часть выполнения скрипта продолжается в обычном режиме.

Синтаксис:

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

Аргументы

Аргумент	Описание
n	Произвольное выражение, результатом которого является целое число, обозначающее максимальное число считываемых записей. Элемент n может быть заключен в скобки: (n).
loadstatement | selectstatement	Элемент load statement/select statement, который следует за аргументом n, определяет указанную таблицу, которая должна быть загружена с учетом заданного максимального количества записей.

Региональные настройки

Если не указано иное, в примерах, приведенных в данном разделе, используется следующий формат даты: ММ/ДД/ГГГГ. Формат даты указан в операторе SET DateFormat скрипта загрузки данных. В вашей системе может быть установлен другой формат даты по умолчанию в зависимости от региональных настроек и других факторов. Можно изменить формат в примерах в соответствии с потребностями. Или можно изменить форматы в скрипте загрузки в соответствии с этими примерами.

Региональные настройки по умолчанию в приложениях основаны на системных региональных настройках компьютера или сервера, где установлено ПО Qlik Sense. Если на сервере Qlik Sense, к которому обращается пользователь, выбран шведский язык, то редактор загрузки данных будет использовать шведские региональные настройки для даты, времени и валюты. Эти параметры регионального формата не связаны с языком, отображаемым в интерфейсе пользователя Qlik Sense. Qlik Sense будет отображаться на языке, который используется в браузере.

Примеры функции

Пример	Результат
FIRST 10 LOAD * from abc.csv;	В этом примере извлекаются первые десять строк из файла Excel.
FIRST (1) SQL SELECT * from Orders;	В этом примере будет извлечена первая выбранная строка из набора данных Orders.

Пример. Загрузка первых пяти строк

Скрипт загрузки и результаты

Обзор

Откройте редактор загрузки данных и добавьте приведенный ниже скрипт загрузки на новую вкладку.

Скрипт загрузки содержит следующее:

- Набор данных с датами за первые две недели 2020 года.
- Переменная `First`, которая указывает приложению загружать только первые пять записей.

Скрипт загрузки

```
Sales:
FIRST 5
LOAD
*
Inline [
date,sales
01/01/2020,6000
01/02/2020,3000
01/03/2020,6000
01/04/2020,8000
01/05/2020,5000
01/06/2020,7000
01/07/2020,3000
01/08/2020,5000
01/09/2020,9000
01/10/2020,5000
01/11/2020,7000
01/12/2020,7000
01/13/2020,7000
01/14/2020,7000
];
```

Результаты

Загрузите данные и откройте лист. Создайте новую таблицу и добавьте `Date` в качестве поля и `sum(sales)` в качестве меры:

Результирующая таблица

Date	sum(sales)
01/01/2020	6000
01/02/2020	3000
01/03/2020	6000
01/04/2020	8000
01/05/2020	5000

Скрипт загружает только первые пять записей таблицы `Sales`.

Generic

Префикс загрузки **Generic** позволяет преобразовывать смоделированные данные типа «объект-атрибут-значение» (EAV) в традиционную нормализованную структуру реляционной таблицы. Моделирование EAV также называют «обобщенным моделированием данных» или «открытой схемой».

Пример смоделированных данных EAV и эквивалентной денормализованной реляционной таблицы

Product ID	Attribute	Value
13	Status	Discontinued
13	Colour	Brown
20	Colour	White
13	Size	13-15
20	Size	16-18

Product ID	Status	Colour	Size
13	Discontinued	Brown	13-15
20		White	16-18

Пример смоделированных данных EAV и эквивалентного набора нормализованных реляционных таблиц

Product ID	Attribute	Value
13	Status	Discontinued
13	Colour	Brown
20	Colour	White
13	Size	13-15
20	Size	16-18

Product ID	Status
13	Discontinued

Product ID	Colour
13	Brown
20	White

Product ID	Size
13	13-15
20	16-18

Хотя технически возможно загружать и анализировать смоделированные данные EAV в Qlik, часто проще работать с эквивалентной традиционной реляционной структурой данных.

Синтаксис:

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

Эти темы помогут вам в работе с этой функцией:

Связанные темы

Тема	Описание
Crosstable (page 48)	Префикс загрузки `Crosstable` преобразует данные, ориентированные по горизонтали, в данные, ориентированные по вертикали. С чисто функциональной точки зрения, он выполняет преобразование, противоположное префиксу загрузки `Generic`, хотя префиксы обычно используются в совершенно разных практических ситуациях.
Универсальные базы данных в *Управление данными*	Модели структурированных данных EAV более подробно описаны здесь.

Пример 1. Преобразование структурированных данных EAV с префиксом загрузки Generic

Скрипт загрузки и выражение диаграммы

Обзор

Откройте редактор загрузки данных и добавьте приведенный ниже скрипт загрузки на новую вкладку.

Скрипт загрузки содержит набор данных, который загружается в таблицу с именем `Transactions`. Набор данных включает поле даты. По умолчанию используется определение `MonthNames`.

Скрипт загрузки

```
Products:
Generic
Load * inline [
Product ID, Attribute, Value
13, Status, Discontinued
13, Color, Brown
20, Color, White
13, Size, 13-15
20, Size, 16-18
2, Status, Discontinued
5, Color, Brown
2, Color, White
44, Color, Brown
45, Size, 16-18
45, Color, Brown
];
```

Результаты

Загрузите данные и откройте лист. Создайте новую таблицу и добавьте это поле как измерение: `Color`.

Добавьте эту меру:

```
=Count([Product ID])
```

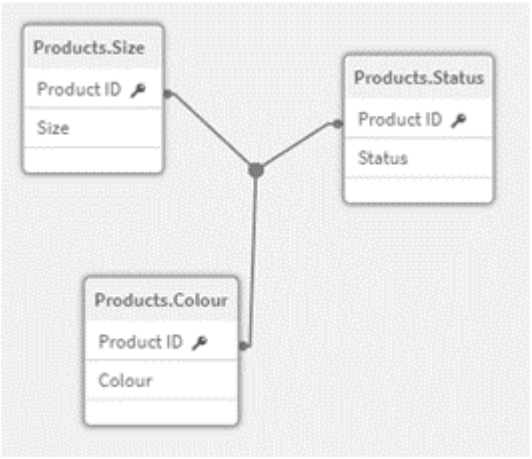
Теперь можно проверить количество товаров по цвету.

Результирующая таблица

Цвет	=Count([Product ID])
Brown	4
White	2

Обратите внимание на форму модели данных, где каждый атрибут вынесен в отдельную таблицу, названную в соответствии с тегом исходной целевой таблицы `Product`. Каждая таблица имеет атрибут в виде суффикса. Одним из примеров этого является `Product.Color`. Результирующие выходные записи Product Attribute связаны с помощью `Product ID`.

Представление результатов в средстве просмотра модели данных



Результирующая таблица записей: Products.Status

Product ID	Состояние
13	Discontinued
2	Discontinued

Результирующая таблица записей: Products.Size

Product ID	Размер
13	13-15
20	16-18
45	16-18

Результирующая таблица записей: Products.Color

Product ID	Цвет
13	Brown
5	Brown
44	Brown
45	Brown
20	White
2	White

Пример 2. Преобразование структурированных данных EAV без префикса загрузки Generic

Скрипт загрузки и выражение диаграммы

Обзор

В этом примере показано, как анализировать структурированные данные EAV в их исходной форме.

Откройте редактор загрузки данных и добавьте приведенный ниже скрипт загрузки на новую вкладку.

Скрипт загрузки содержит набор данных, который загружается в таблицу с именем `Products` в структуре EAV.

В этом примере мы по-прежнему считаем товары по атрибуту цвета. Для анализа данных, структурированных таким образом, потребуется применить фильтрацию на уровне выражения для продуктов, у которых атрибут имеет значение `Color`.

Кроме того, отдельные атрибуты недоступны для выбора в качестве измерений или полей, в результате чего сложнее определить, как создать эффективные визуализации.

Скрипт загрузки

```
Products:
Load * Inline
[
Product ID, Attribute, Value
13, Status, Discontinued
13, Color, Brown
20, Color, White
13, Size, 13-15
20, Size, 16-18
2, Status, Discontinued
5, Color, Brown
2, Color, White
44, Color, Brown
45, Size, 16-18
45, Color, Brown
];
```

Результаты

Загрузите данные и откройте лист. Создайте новую таблицу и добавьте это поле как измерение: `Value`.

Создайте следующую меру:

```
=Count({<Attribute={'Color'}>} [Product ID])
```

Теперь можно проверить количество товаров по цвету.

Результирующая таблица записей: Products.Status

Значение	=Count({<Attribute={'Color'}>} [Product ID])
Brown	4
White	2

Пример 3. Денормализация результирующих выходных таблиц из загрузки с префиксом Generic (расширенный вариант)

Скрипт загрузки и выражение диаграммы

Обзор

В этом примере мы показываем, как нормализованная структура данных, создаваемая префиксом загрузки `Generic`, может быть денормализована обратно в консолидированную таблицу измерений `Product`. Это расширенный метод моделирования, который можно использовать в рамках настройки производительности модели данных.

Откройте редактор загрузки данных и добавьте приведенный ниже скрипт загрузки на новую вкладку.

Скрипт загрузки

```
Products:

Generic
Load * inline [
Product ID, Attribute, Value
13, Status, Discontinued
13, Color, Brown
20, Color, White
13, Size, 13-15
20, Size, 16-18
2, Status, Discontinued
5, Color, Brown
2, Color, White
44, Color, Brown
45, Size, 16-18
45, Color, Brown
];

RENAME TABLE Products.Color TO Products;

OUTER JOIN (Products)
LOAD * RESIDENT Products.Size;

OUTER JOIN (Products)
LOAD * RESIDENT Products.Status;
DROP TABLES Products.Size,Products.Status;
```

Результаты

Откройте средство просмотра модели данных и обратите внимание на форму результирующей модели данных. Присутствует только одна денормализованная таблица. Это комбинация трех промежуточных выходных таблиц: `Products.Size`, `Products.Status` и `Products.Color`.

Результирующая внутренняя модель данных

Products
Product ID
Состояние
Цвет
Размер

Результирующая таблица записей: Products

Product ID	Состояние	Цвет	Размер
13	Discontinued	Brown	13-15
20	-	White	16-18
2	Discontinued	White	-
5	-	Brown	-
44	-	Brown	-
45	-	Brown	16-18

Загрузите данные и откройте лист. Создайте новую таблицу и добавьте это поле как измерение: `Color`.

Добавьте эту меру:

```
=Count([Product ID])
```

Результирующая таблица

Цвет	=Count([Product ID])
Brown	4
White	2

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	Germany	Europe	Germany	-	Europe	Europe\Germany	2
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.
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

NodeID	AncestorID	NodeName	AncestorName	DepthDiff
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.

Аргумент	Описание
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.

Эффект объединения данных заключается в расширении целевой таблицы дополнительным набором полей или атрибутов, а именно теми, которых еще нет в целевой таблице. Любые общие имена полей между исходным набором данных и целевой таблицей используются для определения того, как связать новые входящие записи. Это обычно называют «естественным соединением». Операция Qlik соединения может привести к тому, что результирующая целевая таблица будет иметь больше или меньше записей, чем она содержала в начале, в зависимости от уникальности ассоциации соединения и используемого типа соединения.

Существует четыре типа соединений:

Left join

Левые соединения являются наиболее распространенным типом соединений. Например, если имеется набор данных о транзакциях и требуется объединить его со справочным набором данных, обычно используется `Left Join`. Сначала загружается таблица транзакций, а затем справочный набор данных, который присоединяется через префикс `Left Join` к уже загруженной таблице транзакций. `Left Join` сохранит все транзакции как есть и добавит дополнительные поля справочных данных, где найдено совпадение.

Inner join

Если имеется два набора данных, где вас интересуют только результаты, в которых есть совпадающая связь, попробуйте использовать `Inner Join`. Это приведет к удалению всех записей

как из загруженных исходных данных, так и из целевой таблицы, если совпадений не будет найдено. В результате в целевой таблице может остаться меньше записей, чем было до выполнения операции соединения.

Outer join

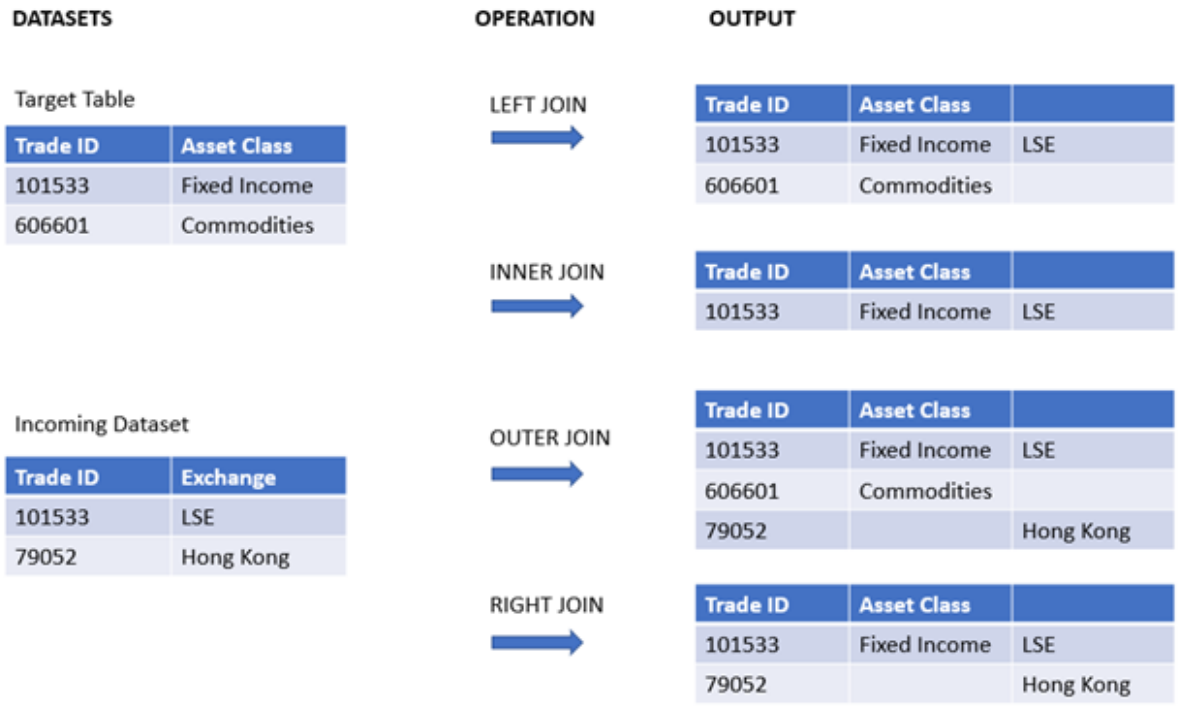
Если требуется сохранить как целевые записи, так и все входящие записи, используйте `Outer Join`. Если совпадений не найдено, каждый набор записей по-прежнему сохраняется, а поля с противоположной стороны объединения остаются незаполненными (null). Внешние соединения обычно не имеют широкого практического применения.

Right join

Этот тип соединения сохраняет все записи, которые должны быть загружены, при этом количество записей в целевой таблице объединения, сокращается, так как включаются только те записи, для которых есть совпадающая связь во входящих записях. Это нишевый тип соединения, который иногда используется как средство сокращения предварительно загруженной таблицы записей до требуемого подмножества.

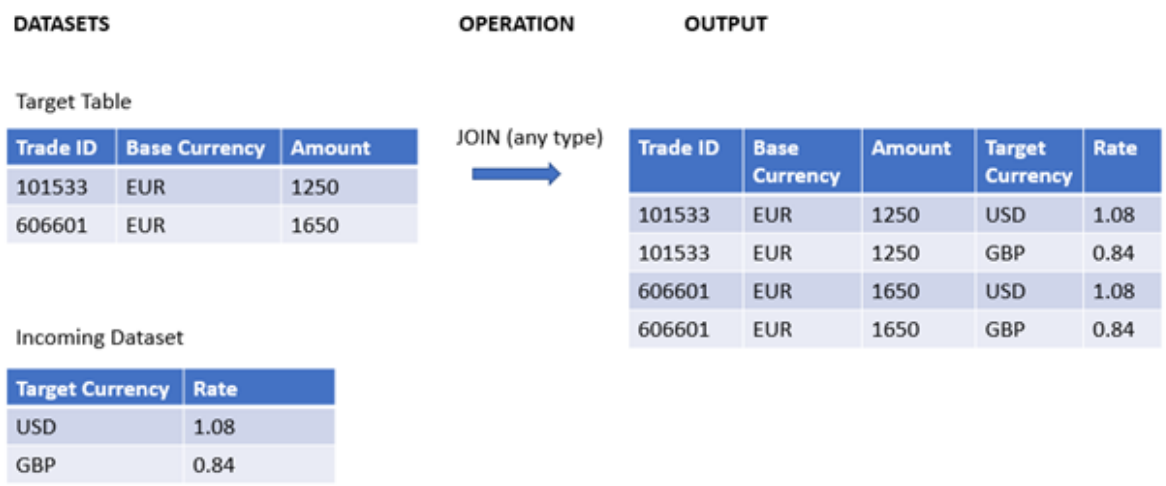
Если ключевое слово type опущено, `Inner Join` является типом соединения по умолчанию.

Примеры наборов результатов для различных типов операции соединения



Если между исходной и целевой таблицей операции соединения нет общих имен полей, результатом соединения будет декартово произведение всех строк – это называется «перекрестным соединением».

Пример набора результатов операции «перекрестное соединение»



Синтаксис:

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

Аргументы

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

Эти темы помогут вам в работе с этой функцией:

Связанные темы

Тема	Описание
Соединение таблиц с помощью операторов Join и Keep в *Управление данными*	В этом разделе дается дальнейшее объяснение понятий «соединения» (Join) и «сохранения» (Keep) наборов данных.
Keep (page 85)	Префикс загрузки `Keep` подобен префиксу `Join`, но он не соединяет исходный и целевой наборы данных. Вместо этого он обрезает каждый набор данных в соответствии с типом принятой операции (внутреннее, внешнее, левое или правое соединение).

Пример 1. Левое соединение: дополнение целевой таблицы справочным набором данных

Скрипт загрузки и результаты

Обзор

Откройте редактор загрузки данных и добавьте приведенный ниже скрипт загрузки на новую вкладку.

Скрипт загрузки содержит следующее:

- Набор данных, представляющий записи об изменениях, который загружается в таблицу с именем `Changes`. Она включает ключевое поле Status ID.
- Второй набор данных, представляющий состояния изменений, который загружается и объединяется с исходными записями об изменениях путем левого соединения с использованием префикса загрузки left `Join`.

Это левое соединение гарантирует, что записи об изменениях останутся нетронутыми при добавлении атрибутов состояния, когда поиск совпадения во входящих записях состояния осуществляется на основе общего идентификатора состояния (столбец Status ID).

Скрипт загрузки

```
Changes:
Load * inline [
Change ID      Status ID       Scheduled Start Date    Scheduled End Date      Business Impact
10030  4       19/01/2022      23/02/2022      None
10015  3       04/01/2022      15/02/2022      Low
10103  1       02/04/2022      29/05/2022      Medium
10185  2       23/06/2022      08/09/2022      None
10323  1       08/11/2022      26/11/2022      High
10326  2       11/11/2022      05/12/2022      None
10138  2       07/05/2022      03/08/2022      None
10031  3       20/01/2022      25/03/2022      Low
10040  1       29/01/2022      22/04/2022      None
10134  1       03/05/2022      08/07/2022      Low
10334  2       19/11/2022      06/02/2023      Low
10220  2       28/07/2022      06/09/2022      None
10264  1       10/09/2022      17/10/2022      Medium
10116  1       15/04/2022      24/04/2022      None
10187  2       25/06/2022      24/08/2022      Low
] (delimiter is '\t');

Status:
Join (Changes)
Load * inline [
Status ID      Status  Sub Status
1      Open    Not Started
2      Open    Started
3      Closed  Complete
```

```
4      Closed  Cancelled
] (delimiter is '\t');
```

Результаты

Откройте средство просмотра модели данных и обратите внимание на форму модели данных. Присутствует только одна денормализованная таблица. Это комбинация всех исходных записей об изменениях с соответствующими атрибутами состояния, присоединенными к каждой записи об изменении.

Результирующая внутренняя модель данных

Changes (Изменения)
Change ID (ИД изменения)
Status ID (ИД состояния)
Scheduled Start Date (Запланированная дата начала)
Scheduled End Date (Запланированная дата окончания)
Business Impact (Воздействие на бизнес)
Состояние
Sub Status (Дополнительное состояние)

Если развернуть окно предварительного просмотра в средстве просмотра модели данных, в нем будет отображаться увидите часть этого полного набора результатов, организованного в виде таблицы:

Предварительный просмотр таблицы Changes в средстве просмотра модели данных

Change ID (ИД изменения)	**Status ID (ИД состояния)**	**Scheduled Start Date (Запланированная дата начала)**	**Scheduled End Date (Запланированная дата окончания)**	**Business Impact (Воздействие на бизнес)**	**Состояние**	**Sub Status (Дополнительное состояние)**
10015	3	04/01/2022	15/02/2022	Низкое	Closed	Завершено
10030	4	19/01/2022	23/02/2022	Отсутствует	Closed	Отменено
10031	3	20/01/2022	25/03/2022	Низкое	Closed	Завершено
10040	1	29/01/2022	22/04/2022	Отсутствует	Открыть	Not Started
10103	1	02/04/2022	29/05/2022	Средний	Открыть	Not Started
10116	1	15/04/2022	24/04/2022	Отсутствует	Открыть	Not Started

Change ID (ИД изменения)	Status ID (ИД состояния)	Scheduled Start Date (Запланированная дата начала)	Scheduled End Date (Запланированная дата окончания)	Business Impact (Воздействие на бизнес)	Состояние	Sub Status (Дополнительное состояние)
10134	1	03/05/2022	08/07/2022	Низкое	Открыть	Not Started
10138	2	07/05/2022	03/08/2022	Отсутствует	Открыть	Выполнение начато
10185	2	23/06/2022	08/09/2022	Отсутствует	Открыть	Выполнение начато
10187	2	25/06/2022	24/08/2022	Низкое	Открыть	Выполнение начато
10220	2	28/07/2022	06/09/2022	Отсутствует	Открыть	Выполнение начато
10264	1	10/09/2022	17/10/2022	Средний	Открыть	Not Started
10323	1	08/11/2022	26/11/2022	Высокое	Открыть	Not Started
10326	2	11/11/2022	05/12/2022	Отсутствует	Открыть	Выполнение начато
10334	2	19/11/2022	06/02/2023	Низкое	Открыть	Выполнение начато

Откройте редактор загрузки данных. Загрузите данные и откройте лист. Создайте новую таблицу и добавьте это поле как измерение: `Status`.

Добавьте эту меру:

```
=Count([Change ID])
```

Теперь можно проверить количество изменений по состоянию.

Результирующая таблица

Состояние	=Count([Change ID])
Открыть	12
Closed	3

Пример 2. Внутреннее соединение: соединение только совпадающих записей

Скрипт загрузки и результаты

Обзор

Откройте редактор загрузки данных и добавьте приведенный ниже скрипт загрузки на новую вкладку.

Скрипт загрузки содержит следующее:

- Набор данных, представляющий записи об изменениях, который загружается в таблицу с именем `Changes`.
- Второй набор данных, представляющий записи об изменениях из исходной системы `JIRA`. Он загружается и объединяется с исходными записями путем соединения с использованием префикса загрузки `Inner Join`.

Это `Inner Join` гарантирует, что будут сохранены только пять записей об изменениях, найденных в обоих наборах данных.

Скрипт загрузки

```
Changes:
Load * inline [
Change ID      Status ID       Scheduled Start Date    Scheduled End Date      Business Impact
10030  4       19/01/2022      23/02/2022      None
10015  3       04/01/2022      15/02/2022      Low
10103  1       02/04/2022      29/05/2022      Medium
10185  2       23/06/2022      08/09/2022      None
10323  1       08/11/2022      26/11/2022      High
10326  2       11/11/2022      05/12/2022      None
10138  2       07/05/2022      03/08/2022      None
10031  3       20/01/2022      25/03/2022      Low
10040  1       29/01/2022      22/04/2022      None
10134  1       03/05/2022      08/07/2022      Low
10334  2       19/11/2022      06/02/2023      Low
10220  2       28/07/2022      06/09/2022      None
10264  1       10/09/2022      17/10/2022      Medium
10116  1       15/04/2022      24/04/2022      None
10187  2       25/06/2022      24/08/2022      Low
] (delimiter is '\t');

JIRA_changes:
Inner Join (Changes)
Load
  [Ticket ID] AS [Change ID],
  [Source System]
inline
[
Ticket ID      Source System
10030  JIRA
```

```
10323  JIRA
10134  JIRA
10334  JIRA
10220  JIRA
10187  JIRA
] (delimiter is '\t');
```

Результаты

Загрузите данные и откройте лист. Создайте новую таблицу и добавьте эти поля как измерения:

- `Source System`
- `Change ID`
- `Business Impact`

Теперь можно просмотреть пять полученных записей.

Результирующая таблица

Source System (Исходная система)	Change ID (ИД изменения)	Business Impact (Воздействие на бизнес)
JIRA	10030	Отсутствует
JIRA	10134	Низкое
JIRA	10220	Отсутствует
JIRA	10323	Высокое
JIRA	10334	Низкое

Пример 3. Внешнее соединение: соединение перекрывающихся наборов записей

Скрипт загрузки и результаты

Обзор

Откройте редактор загрузки данных и добавьте приведенный ниже скрипт загрузки на новую вкладку.

Скрипт загрузки содержит следующее:

- Набор данных, представляющий записи об изменениях, который загружается в таблицу с именем `Changes`.
- Второй набор данных, представляющий записи об изменениях из исходной системы `JIRA`, который загружается и объединяется с исходными записями путем соединения с использованием префикса загрузки `Outer Join`.

Это гарантирует сохранение всех перекрывающихся записей об изменениях из обоих наборов данных.

Скрипт загрузки

```
// 8 Change records

Changes:
Load * inline [
Change ID      Status ID       Scheduled Start Date    Scheduled End Date      Business Impact
10030  4       19/01/2022      23/02/2022      None
10015  3       04/01/2022      15/02/2022      Low
10138  2       07/05/2022      03/08/2022      None
10031  3       20/01/2022      25/03/2022      Low
10040  1       29/01/2022      22/04/2022      None
10134  1       03/05/2022      08/07/2022      Low
10334  2       19/11/2022      06/02/2023      Low
10220  2       28/07/2022      06/09/2022      None
] (delimiter is '\t');

// 6 Change records

JIRA_changes:
Outer Join (Changes)
Load
  [Ticket ID] AS [Change ID],
  [Source System]
inline
[
Ticket ID      Source System
10030  JIRA
10323  JIRA
10134  JIRA
10334  JIRA
10220  JIRA
10597  JIRA
] (delimiter is '\t');
```

Результаты

Загрузите данные и откройте лист. Создайте новую таблицу и добавьте эти поля как измерения:

- Source System
- Change ID
- Business Impact

Теперь можно просмотреть 10 полученных записей.

Результирующая таблица

Source System (Исходная система)	Change ID (ИД изменения)	Business Impact (Воздействие на бизнес)
JIRA	10030	Отсутствует
JIRA	10134	Низкое
JIRA	10220	Отсутствует
JIRA	10323	-
JIRA	10334	Низкое
JIRA	10597	-
-	10015	Низкое
-	10031	Низкое
-	10040	Отсутствует
-	10138	Отсутствует

Пример 4. Правое соединение: сокращение целевой таблицы вторичным основным набором данных

Скрипт загрузки и результаты

Обзор

Откройте редактор загрузки данных и добавьте приведенный ниже скрипт загрузки на новую вкладку.

Скрипт загрузки содержит следующее:

- Набор данных, представляющий записи об изменениях, который загружается в таблицу с именем `Changes`.
- Второй набор данных, представляющий записи об изменениях из исходной системы `Teamwork`. Он загружается и объединяется с исходными записями путем соединения с использованием префикса загрузки `Right Join`.

Это гарантирует, что будут сохранены только записи об изменениях `Teamwork`, при этом не будут потеряны никакие записи `Teamwork`, если в целевой таблице нет соответствующих `Change ID`.

Скрипт загрузки

```
Changes:

Load * inline [
Change ID      Status ID       Scheduled Start Date    Scheduled End Date      Business Impact
10030  4       19/01/2022      23/02/2022      None
10015  3       04/01/2022      15/02/2022      Low
```

```
10103  1       02/04/2022      29/05/2022      Medium
10185  2       23/06/2022      08/09/2022      None
10323  1       08/11/2022      26/11/2022      High
10326  2       11/11/2022      05/12/2022      None
10138  2       07/05/2022      03/08/2022      None
10031  3       20/01/2022      25/03/2022      Low
10040  1       29/01/2022      22/04/2022      None
10134  1       03/05/2022      08/07/2022      Low
10334  2       19/11/2022      06/02/2023      Low
10220  2       28/07/2022      06/09/2022      None
10264  1       10/09/2022      17/10/2022      Medium
10116  1       15/04/2022      24/04/2022      None
10187  2       25/06/2022      24/08/2022      Low
] (delimiter is '\t');

Teamwork_changes:
Right Join (Changes)
Load
  [Ticket ID] AS [Change ID],
  [Source System]
inline
[
Ticket ID      Source System
10040  Teamwork
10015  Teamwork
10103  Teamwork
10031  Teamwork
50231  Teamwork
] (delimiter is '\t');
```

Результаты

Загрузите данные и откройте лист. Создайте новую таблицу и добавьте эти поля как измерения:

- Source System
- Change ID
- Business Impact

Теперь можно просмотреть пять полученных записей.

Результирующая таблица

Source System (Исходная система)	Change ID (ИД изменения)	Business Impact (Воздействие на бизнес)
Teamwork (Коллективная работа)	10015	Низкое
Teamwork (Коллективная работа)	10031	Низкое

Source System (Исходная система)	Change ID (ИД изменения)	Business Impact (Воздействие на бизнес)
Teamwork (Коллективная работа)	10040	Отсутствует
Teamwork (Коллективная работа)	10103	Средний
Teamwork (Коллективная работа)	50231	-

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 1466)

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

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

Update: 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.

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

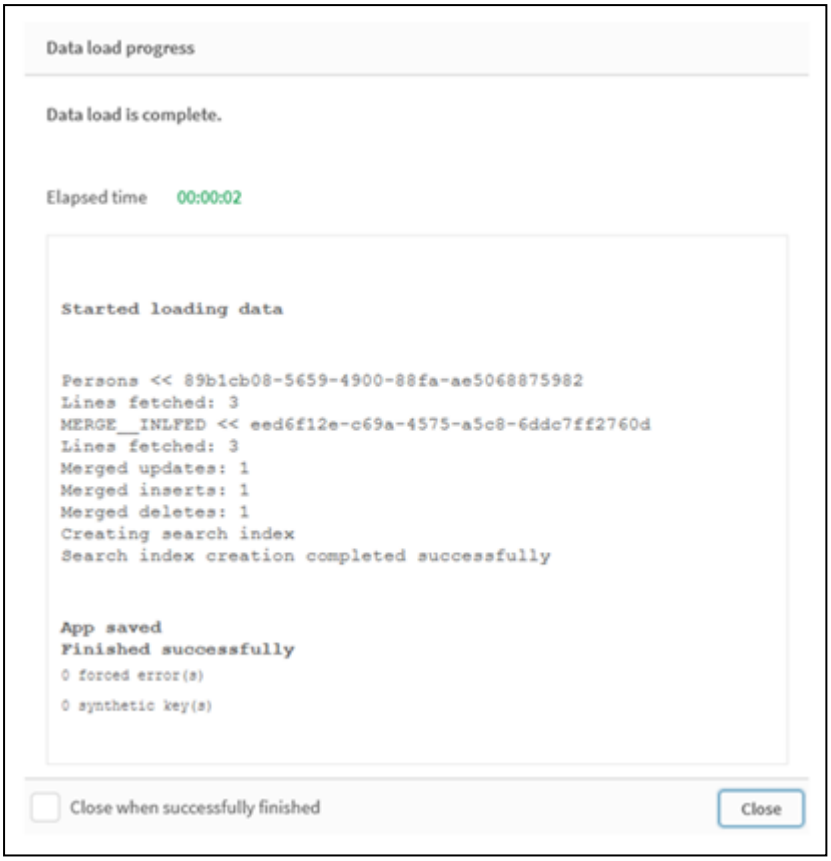
После выполнения **Merge Load** таблица выглядит следующим образом.

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 )
```

Если загружается таблица, содержащая идентичное количество полей с такими же именами, как в таблице, загруженной скриптом ранее, то Qlik Sense по умолчанию объединит эти две таблицы. Это произойдет, даже если вторая таблица имеет другое имя.

Однако если префикс скрипта `NoConcatenate` добавлен перед оператором load или оператором второй таблицы, то эти две таблицы будут загружаться по отдельности.

Типичным примером применения `NoConcatenate` является случай, когда необходимо создать временную копию таблицы, чтобы выполнить некоторые временные преобразования в копии, сохраняя неизмененными первоначальные данные. `NoConcatenate` позволяет создать такую копию, исключив ее неявное добавление в исходную таблицу.

Региональные настройки

Если не указано иное, в примерах, приведенных в данном разделе, используется следующий формат даты: ММ/ДД/ГГГГ. Формат даты указан в операторе `SET DateFormat` скрипта загрузки данных. В вашей системе может быть установлен другой формат даты по умолчанию в зависимости от региональных настроек и других факторов. Можно изменить формат в примерах в соответствии с потребностями. Или можно изменить форматы в скрипте загрузки в соответствии с этими примерами.

Региональные настройки по умолчанию в приложениях основаны на системных региональных настройках компьютера или сервера, где установлено ПО Qlik Sense. Если на сервере Qlik Sense, к которому обращается пользователь, выбран шведский язык, то редактор загрузки данных будет использовать шведские региональные настройки для даты, времени и валюты. Эти параметры регионального формата не связаны с языком, отображаемым в интерфейсе пользователя Qlik Sense. Qlik Sense будет отображаться на языке, который используется в браузере.

Пример функции

Пример	Результат
`Source: LOAD A,B from file1.csv; CopyOfSource: NoConcatenate LOAD A,B resident Source;`	Загружена таблица с мерами `A` и `B`. Вторая таблица с такими же полями загружается отдельно с использованием переменной `NoConcatenate`.

Пример 1. Неявное объединение

Скрипт загрузки и результаты

Обзор

В этом примере будут добавлены два скрипта загрузки, один за другим.

Откройте редактор загрузки данных и добавьте приведенный ниже скрипт загрузки на новую вкладку.

Скрипт загрузки содержит следующее:

- Первоначальный набор данных с датами и суммами, который отправляется в таблицу под именем `Transactions`.

Первый скрипт загрузки

```
Transactions:
LOAD
*
Inline [
id, date, amount
1, 08/30/2018, 23.56
2, 09/07/2018, 556.31
3, 09/16/2018, 5.75
4, 09/22/2018, 125.00
5, 09/22/2018, 484.21
6, 09/22/2018, 59.18
7, 09/23/2018, 177.42
];
```

Результаты

Загрузите данные и откройте лист. Создайте новую таблицу и добавьте эти поля как измерения:

- `id`
- `date`
- `amount`

Первая таблица результатов

id	date	amount
1	08/30/2018	23.56
2	09/07/2018	556.31
3	09/16/2018	5.75
4	09/22/2018	125.00
5	09/22/2018	484.21
6	09/22/2018	59.18
7	09/23/2018	177.42

Второй скрипт загрузки

Откройте редактор загрузки данных и добавьте приведенный ниже скрипт загрузки на новую вкладку.

Скрипт загрузки содержит следующее:

- Второй набор данных с идентичными полями отправляется в таблицу под именем `Sales`.

```
Sales:
LOAD
*
Inline [
id, date, amount
8, 10/01/2018, 164.27
9, 10/03/2018, 384.00
10, 10/06/2018, 25.82
11, 10/09/2018, 312.00
12, 10/15/2018, 4.56
13, 10/16/2018, 90.24
14, 10/18/2018, 19.32
];
```

Результаты

Загрузите данные и откройте таблицу.

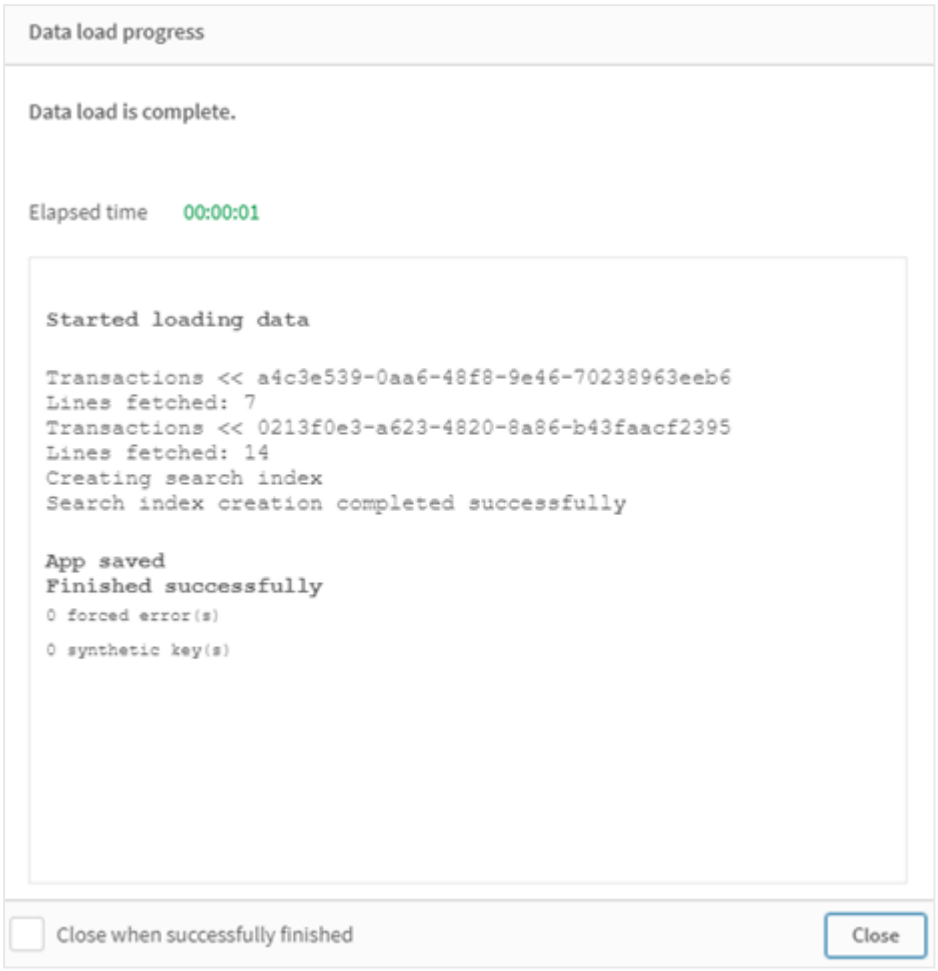
Вторая таблица результатов

id	date	amount
1	08/30/2018	23.56
2	09/07/2018	556.31
3	09/16/2018	5.75
4	09/22/2018	125.00
5	09/22/2018	484.21
6	09/22/2018	59.18
7	09/23/2018	177.42
8	10/01/2018	164.27
9	10/03/2018	384.00
10	10/06/2018	25.82
11	10/09/2018	312.00
12	10/15/2018	4.56
13	10/16/2018	90.24
14	10/18/2018	19.32

Когда выполняется скрипт, таблица `Sales` неявно объединяется с существующей таблицей `Transactions`, так как в двух наборах данных содержится одинаковое количество полей с идентичными именами. Это происходит несмотря на то, что тег имени второй таблицы пытается присвоить результирующему набору имя `'Sales'`.

Чтобы подтвердить неявное объединение набора данных Sales, проверьте записи в журнале **Выполнение загрузки данных**.

Журнал «Выполнение загрузки данных», подтверждающий неявное объединение набора данных Transactions.



Пример 2. Сценарий применения

Скрипт загрузки и результаты

Обзор

В этом сценарии применения имеется следующее:

- Набор данных Transactions, включающий следующие поля:
  - id
  - date
  - amount (in GBP)
- Таблица Currency, включающая:
  - Курс обмена долларов США (USD) на британские фунты (GBP)
- Второй набор данных Transactions, включающий следующие поля:

- id
- date
- amount (в USD)

Будут загружены пять скриптов, один за другим.

- Первый скрипт загрузки содержит первоначальный набор данных с датами и суммами в британских фунтах (GBP), который отправляется в таблицу под именем `Transactions`.
- Второй скрипт загрузки содержит следующее:
  - Второй набор данных с датами и суммами в долларах США (USD), который отправляется в таблицу под именем `Transactions_in_USD`.
  - Префикс `noconcatenate`, добавленный перед оператором load набора данных `Transactions_in_USD` для предотвращения неявного объединения.
- Третий скрипт загрузки содержит префикс `join`, который будет использоваться для создания курса обмена валют между британскими фунтами и долларами США в таблице `Transactions_in_USD`.
- Четвертый скрипт загрузки содержит префикс `concatenate`, который добавляет `Transactions_in_USD` в первоначальную таблицу `Transactions`.
- Пятый скрипт загрузки содержит оператор `drop table`, который удалит таблицу `Transactions_in_USD` после объединения ее данных с таблицей `Transactions`.

Первый скрипт загрузки

```
Transactions:

Load * Inline [
id, date, amount
1, 12/30/2018, 23.56
2, 12/07/2018, 556.31
3, 12/16/2018, 5.75
4, 12/22/2018, 125.00
5, 12/22/2018, 484.21
6, 12/22/2018, 59.18
7, 12/23/2018, 177.42
];
```

Результаты

Загрузите данные и откройте лист. Создайте новую таблицу и добавьте эти поля как измерения:

- `id`
- `date`
- `amount`

Результаты первого скрипта загрузки

id	date	amount
1	12/30/2018	23.56
2	12/07/2018	556.31
3	12/16/2018	5.75
4	12/22/2018	125.00
5	12/22/2018	484.21
6	12/22/2018	59.18
7	12/23/2018	177.42

В таблице отображается первоначальный набор данных с суммами в британских фунтах (GBP).

Второй скрипт загрузки

```
Transactions_in_USD:
NoConcatenate
Load * Inline [
id, date, amount
8, 01/01/2019, 164.27
9, 01/03/2019, 384.00
10, 01/06/2019, 25.82
11, 01/09/2019, 312.00
12, 01/15/2019, 4.56
13, 01/16/2019, 90.24
14, 01/18/2019, 19.32
];
```

Результаты

Загрузите данные и откройте таблицу.

Результаты второго скрипта загрузки

id	date	amount
1	12/30/2018	23.56
2	12/07/2018	556.31
3	12/16/2018	5.75
4	12/22/2018	125.00
5	12/22/2018	484.21
6	12/22/2018	59.18
7	12/23/2018	177.42

id	date	amount
8	01/01/2019	164.27
9	01/03/2019	384.00
10	01/06/2019	25.82
11	01/09/2019	312.00
12	01/15/2019	4.56
13	01/16/2019	90.24
14	01/18/2019	19.32

Обратите внимание, что добавлен второй набора данных из таблицы `Transactions_in_USD`.

Третий скрипт загрузки

Этот скрипт загрузки объединяет курс обмена долларов США (USD) на британские фунты (GBP) в таблицу `Transactions_in_USD`.

```
Join (Transactions_in_USD)
Load * Inline [
rate
0.7
];
```

Результаты

Загрузите данные и откройте просмотр модели данных. Выберите таблицу `Transactions_in_USD`, и обратите внимание, что для каждой существующей записи добавлено значение поля rate, равное 0.7.

Четвертый скрипт загрузки

Используя резидентную загрузку, этот скрипт загрузки объединит таблицы `Transactions_in_USD` и `Transactions` после перевода сумм в доллары США (USD).

```
Concatenate (Transactions)
LOAD
id,
date,
amount * rate as amount
Resident Transactions_in_USD;
```

Результаты

Загрузите данные и откройте таблицу. Появятся новые записи с суммами в британских фунтах (GBP) из строк 8 – 14.

Результаты четвертого скрипта загрузки

id	date	amount
1	12/30/2018	23.56
2	12/07/2018	556.31
3	12/16/2018	5.75
4	12/22/2018	125.00
5	12/22/2018	484.21
6	12/22/2018	59.18
7	12/23/2018	177.42
8	01/01/2019	114.989
8	01/01/2019	164.27
9	01/03/2019	268.80
9	01/03/2019	384.00
10	01/06/2019	18.074
10	01/06/2019	25.82
11	01/09/2019	218.40
11	01/09/2019	312.00
12	01/15/2019	3.192
12	01/15/2019	4.56
13	01/16/2019	63.168
13	01/16/2019	90.24
14	01/18/2019	13.524
14	01/18/2019	19.32

Пятый скрипт загрузки

Этот скрипт загрузки удалит повторяющиеся записи из таблицы результатов четвертого скрипта загрузки, оставляя только записи с суммами в британских фунтах (GBP).

```
drop tables Transactions_in_USD;
```

Результаты

Загрузите данные и откройте таблицу.

Результаты пятого скрипта загрузки

id	date	amount
1	12/30/2018	23.56
2	12/07/2018	556.31
3	12/16/2018	5.75
4	12/22/2018	125.00
5	12/22/2018	484.21
6	12/22/2018	59.18
7	12/23/2018	177.42
8	01/01/2019	114.989
9	01/03/2019	268.80
10	01/06/2019	18.074
11	01/09/2019	218.40
12	01/15/2019	3.192
13	01/16/2019	63.168
14	01/18/2019	13.524

После выполнения пятого скрипта загрузки в таблице результатов отображаются все 14 транзакций, которые существовали в обоих наборах данных транзакций. Однако для транзакций 8-14 суммы переведены в британские фунты (GBP).

Если убрать префикс `NoConcatenate`, который стоял перед `Transactions_in_USD` во втором скрипте загрузки, выполнение скрипта завершится ошибкой «Таблица `Transactions_in_USD` не найдена». Это связано с тем, что в таком случае таблица `Transactions_in_USD` будет автоматически объединена с первоначальной таблицей `Transactions`.

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.

Ограничение

При частичной перезагрузке могут удаляться значения из данных. Однако это не будет отражено в списке уникальных значений, которые хранятся во внутренней таблице. Поэтому даже после частичной перезагрузки список будет содержать все уникальные значения, которые существовали в поле с момента последней полной перезагрузки, то есть их может быть больше, чем существует сейчас после частичной перезагрузки. Это влияет на вывод функций FieldValueCount() и FieldValue (). Функция FieldValueCount() потенциально может вернуть количество, превышающее текущее количество значений поля.

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

ID	Value
5	5
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 1475)

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 )
```

Проверяемое выражение определяет не процент записей из набора данных, которые будут загружены в приложение Qlik Sense, а вероятность загрузки каждой прочитанной записи в приложение. Другими словами, установка `p = 0.5` означает не то, что будет загружено 50% от общего количества записей, а то, что для каждой записи существует 50-процентная вероятность загрузки в приложение Qlik Sense.

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.

Когда это следует использовать

Sample полезно использовать, когда требуется сделать выборку данных из большой таблицы, чтобы оценить природу данных, распределение или содержание полей. Так как выбирается лишь часть данных, загрузка выполняется быстрее, что обеспечивает ускоренное тестирование скриптов. В отличие от `First`, функция `Sample` выбирает данные из всей таблицы, а не только из нескольких первых строк. В некоторых случаях это позволяет обеспечить более точное представление данных.

Следующие примеры демонстрируют два возможных варианта использования префикса скрипта `Sample`:

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

Региональные настройки

Если не указано иное, в примерах, приведенных в данном разделе, используется следующий формат даты: MM/ДД/ГГГГ. Формат даты указан в операторе `SET DateFormat` скрипта загрузки данных. В вашей системе может быть установлен другой формат даты по умолчанию в зависимости от региональных настроек и других факторов. Можно изменить формат в примерах в соответствии с потребностями. Или можно изменить форматы в скрипте загрузки в соответствии с этими примерами.

Региональные настройки по умолчанию в приложениях основаны на системных региональных настройках компьютера или сервера, где установлено ПО Qlik Sense. Если на сервере Qlik Sense, к которому обращается пользователь, выбран шведский язык, то редактор загрузки данных будет использовать шведские региональные настройки для даты, времени и валюты. Эти параметры регионального формата не связаны с языком, отображаемым в интерфейсе пользователя Qlik Sense. Qlik Sense будет отображаться на языке, который используется в браузере.

Пример 1. Выборка из встроенной таблицы

Скрипт загрузки и результаты

Обзор

В этом примере скрипт загружает в таблицу `Transactions` выборку из набора данных, содержащего семь записей, из встроенной таблицы.

Скрипт загрузки

```
Transactions:
SAMPLE 0.3
LOAD
*
Inline [
id, date, amount
1, 08/30/2018, 23.56
2, 09/07/2018, 556.31
```

```
3, 09/16/2018, 5.75
4, 09/22/2018, 125.00
5, 09/22/2018, 484.21
6, 09/22/2018, 59.18
7, 09/23/2018, 177.42
];
```

Результаты

Загрузите данные и откройте лист. Создайте новую таблицу и добавьте эти поля как измерения:

- `id`
- `amount`

Добавьте следующую меру:

```
=sum(amount)8
```

Результирующая таблица

id	date	=Sum(amount)
2	09/07/2018	556.31
4	09/22/2018	125
1	08/30/2018	23.56
3	09/16/2018	5.75

При итерации загрузки, использованной в этом примере, были прочитаны все семь записей, но только четыре из них загружены в таблицу данных. При любом повторном выполнении загрузки число записей может оказаться другим, как и набор записей, загруженных в приложение.

Пример 2. Образец из автоматически созданной таблицы

Скрипт загрузки и результаты

Обзор

В этом примере с помощью `Autogenerate` создается набор данных из 100 записей с полями `date`, `id` и `amount`. Однако используется префикс `Sample` со значением 0.1.

Скрипт загрузки

```
SampleData:
Sample 0.1
LOAD
RecNo() AS id,
MakeDate(2013, Ceil(Rand() * 12), Ceil(Rand() * 29)) as date,
Rand() * 1000 AS amount

Autogenerate(100);
```

Результаты

Загрузите данные и откройте лист. Создайте новую таблицу и добавьте эти поля как измерения:

- `id`
- `amount`

Добавьте следующую меру:

Результирующая таблица

id	date	=Sum(amount)
48	9/28/2013	763
20	5/15/2013	752
19	11/8/2013	657
25	3/24/2013	522
27	8/23/2013	389
81	6/1/2013	53
100	8/15/2013	17

При итерации загрузки, используемой в этом примере, загружаются семь записей из созданного набора данных. Опять же, при любом повторном выполнении загрузки число записей может оказаться другим, как и набор записей, загруженных в приложение.

Semantic

Префикс загрузки `semantic` создает поле особого типа, которое можно использовать в Qlik Sense для подключения реляционных данных и управления ими, включая древовидные структуры, самоссылающиеся структурированные данные «родитель-потомок» и (или) данные, которые можно представить в виде графика.

Обратите внимание, что загрузка `semantic` может работать аналогично префиксам *Hierarchy (page 66)* и *HierarchyBelongsTo (page 68)*. Все три префикса можно использовать в качестве компонентов при создании эффективных интерфейсных решений для просмотра реляционных данных.

Синтаксис:

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

Семантическая нагрузка предполагает входные данные шириной ровно три или четыре поля со строгим определением того, что представляет каждое упорядоченное поле, как показано в таблице ниже:

Поля смысловой нагрузки

Имя поля	Описание поля
1-е поле:	этот тег является представлением первого из двух объектов, между которыми существует связь.
2-е поле:	этот тег будет использоваться для описания «прямой» связи между первым и вторым объектом. Если первый объект является дочерним, а второй объект является родительским, можно создать вкладку связи с указанием parent (родительский) или parent of (родительский объект), как при прослеживании связи от дочернего объекта к родительскому.
3-е поле:	этот тег является представлением второго из двух объектов, между которыми существует связь.
4-е поле:	это поле является необязательным. этот тег будет использоваться для описания «обратной» связи между первым и вторым объектом. Если первый объект является дочерним, а второй объект является родительским, на вкладке связи с может быть указание child (дочерний) или child of (дочерний объект), как при прослеживании связи от родительского объекта к дочернему. Если не добавлять четвертое поле, то тег второго поля будет использоваться для описания отношения в любом направлении. В этом случае символ стрелки автоматически добавляется как часть тега.

Следующий код является примером префикса `semantic`.

```
Semantic
Load
Object,
‘Parent’ AS Relationship,
NeighbouringObject AS Object,
‘Child’ AS Relationship
from graphdata.csv;
```

Допускается и является обычной практикой маркировать третье поле так же, как и первое поле. Это создает самоссылающийся поиск, что позволяет прослеживать объекты к связанным объектам на расстоянии одного шага связи за раз. Если 3-е поле содержит другое имя, то конечным результатом будет простой поиск от объектов к их непосредственным реляционным соседям на расстоянии только одного шага, что не имеет большой практической ценности.

Региональные настройки

Если не указано иное, в примерах, приведенных в данном разделе, используется следующий формат даты: ММ/ДД/ГГГГ. Формат даты указан в операторе `SET DateFormat` скрипта загрузки данных. В вашей системе может быть установлен другой формат даты по умолчанию в зависимости от региональных настроек и других факторов. Можно изменить формат в примерах в соответствии с потребностями. Или можно изменить форматы в скрипте загрузки в соответствии с этими примерами.

Региональные настройки по умолчанию в приложениях основаны на системных региональных настройках компьютера или сервера, где установлено ПО Qlik Sense. Если на сервере Qlik Sense, к которому обращается пользователь, выбран шведский язык, то редактор загрузки данных будет использовать шведские региональные настройки для даты, времени и валюты. Эти параметры регионального формата не связаны с языком, отображаемым в интерфейсе пользователя Qlik Sense. Qlik Sense будет отображаться на языке, который используется в браузере.

Связанные функции

Функции	Взаимодействие
Hierarchy (page 66)	Префикс загрузки Hierarchy используется для разделения и организации узлов в структурах данных «родитель-потомок» и других графоподобных структурах, а также для их преобразования в таблицы.
HierarchyBelongsTo (page 68)	Префикс загрузки HierarchyBelongsTo используется для поиска и организации предков в структурах данных «родитель-потомок» и других графоподобных структурах, а также для их преобразования в таблицы.

Пример. Создание специального поля для соединения связей с помощью семантического префикса

Скрипт загрузки и результаты

Обзор

Откройте редактор загрузки данных и добавьте приведенный ниже скрипт загрузки на новую вкладку.

Скрипт загрузки содержит следующее:

- Набор данных, представляющий записи, связанные с географическим положением, который загружается в таблицу с именем `GeographyTree`.
    - Каждая запись имеет идентификатор ID в начале строки и ParentID в конце строки.
- Префикс `semantic`, который добавит одно специальное поле поведения с именем `Relation`.

Скрипт загрузки

```
GeographyTree:
LOAD
    ID,
    Geography,
    if(ParentID='',null(),ParentID) AS ParentID

INLINE [
ID,Geography,ParentID
1,World
2,Europe,1
3,Asia,1
4,North America,1
5,South America,1
6,UK,2
```

```
7,Germany,2
8,Sweden,2
9,South Korea,3
10,North Korea,3
11,China,3
12,London,6
13,Birmingham,6
];

SemanticTable:
Semantic Load
       ID as ID,
       'Parent' as Relation,
       ParentID as ID,
       'Child' as Relation
resident GeographyTree;
```

Результаты

Загрузите данные и откройте лист. Создайте новую таблицу и добавьте эти поля как измерения:

- `Id`
- `Geography`

Затем создайте фильтр с `Relation` в качестве измерения. Щелкните **Изменение завершено**.

Результирующая таблица

Id	Geography
1	World
2	Europe
3	Asia
4	North America
5	South America
6	UK
7	Germany
8	Sweden
9	South Korea
10	North Korea
11	Китай
12	London
13	Birmingham

Фильтр

Relation

Child

Parent

Нажмите **Europe** в измерении `Geography` в таблице и нажмите **Child** в измерении `Relation` фильтра. Обратите внимание на ожидаемый результат в таблице:

Таблица результатов с «дочерними объектами» Европы

Id	Geography
6	UK
7	Germany
8	Sweden

Если щелкнуть **Child** еще раз, отобразятся места, которые являются «дочерними объектами» Соединенного Королевства (UK), на один шаг ниже.

Таблица результатов с «дочерними объектами» Соединенного Королевства

Id	Geography
12	London
13	Birmingham

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

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**.

Когда это следует использовать

Оператор `Unless` возвращает результат в виде булева значения. Как правило, этот тип функции используется в качестве условия, когда пользователю требуется выполнить условную загрузку или исключить части скрипта.

Ниже приводится три примера использования функции `Unless`.

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

Региональные настройки

Если не указано иное, в примерах, приведенных в данном разделе, используется следующий формат даты: ММ/ДД/ГГГГ. Формат даты указан в операторе `SET DateFormat` скрипта загрузки данных. В вашей системе может быть установлен другой формат даты по умолчанию в зависимости от региональных настроек и других факторов. Можно изменить формат в примерах в соответствии с потребностями. Или можно изменить форматы в скрипте загрузки в соответствии с этими примерами.

Региональные настройки по умолчанию в приложениях основаны на системных региональных настройках компьютера или сервера, где установлено ПО Qlik Sense. Если на сервере Qlik Sense, к которому обращается пользователь, выбран шведский язык, то редактор загрузки данных будет использовать шведские региональные настройки для даты, времени и валюты. Эти параметры регионального формата не связаны с языком, отображаемым в интерфейсе пользователя Qlik Sense. Qlik Sense будет отображаться на языке, который используется в браузере.

Пример 1. Префикс Unless

Скрипт загрузки и результаты

Обзор

Откройте редактор загрузки данных и добавьте приведенный ниже скрипт загрузки на новую вкладку.

Скрипт загрузки содержит следующее:

- Создание переменной A, которой присваивается значение 1.
- Набор данных, загружаемый в таблицу под именем Transactions, если переменная A не имеет значение 2.

Скрипт загрузки

```
LET A = 1;

UNLESS A = 2

Transactions:
LOAD
*
Inline [
id, date, amount
1, 08/30/2018, 23.56
2, 09/07/2018, 556.31
3, 09/16/2018, 5.75
4, 09/22/2018, 125.00
5, 09/22/2018, 484.21
6, 09/22/2018, 59.18
7, 09/23/2018, 177.42
];
```

Результаты

Загрузите данные и откройте лист. Создайте новую таблицу и добавьте эти поля как измерения:

- id
- date
- amount

Результирующая таблица

id	date	amount
1	08/30/2018	23.56
2	09/07/2018	556.31
3	09/16/2018	5.75
4	09/22/2018	125.00
5	09/22/2018	484.21
6	09/22/2018	59.18
7	09/23/2018	177.42

Так как в начале скрипта переменной A задается значение 1, проверяется выполнение условия после префикса `Unless` и возвращается результат `FALSE`. Таким образом, скрипт продолжает выполнение оператора `Load`. В таблице результатов отображаются все записи из таблицы `Transactions`.

Если этой переменной задано значение 2, данные не будут загружаться в модель данных.

Пример 2. Суффикс Unless

Скрипт загрузки и результаты

Обзор

Откройте редактор загрузки данных и добавьте приведенный ниже скрипт загрузки на новую вкладку.

Скрипт загрузки сначала загружает начальный набор данных в таблицу под именем `Transactions`. Затем выполнение скрипта прерывается, если таблица `Transactions` содержит больше 10 записей.

Если это условие не приводит к прекращению выполнения скрипта, следующий набор транзакций добавляется в таблицу `Transactions` в результате объединения, после чего этот процесс повторяется.

Скрипт загрузки

```
Transactions:
LOAD
*
Inline [
id, date, amount
1, 08/30/2018, 23.56
2, 09/07/2018, 556.31
3, 09/16/2018, 5.75
4, 09/22/2018, 125.00
5, 09/22/2018, 484.21
6, 09/22/2018, 59.18
7, 09/23/2018, 177.42
];


exit script unless NoOfRows('Transactions') < 10 ;

Concatenate
LOAD
*
Inline [
id, date, amount
8, 10/01/2018, 164.27
9, 10/03/2018, 384.00
10, 10/06/2018, 25.82
11, 10/09/2018, 312.00
12, 10/15/2018, 4.56
13, 10/16/2018, 90.24
14, 10/18/2018, 19.32
];

exit script unless NoOfRows('Transactions') < 10 ;
```

```
Concatenate
LOAD
*
Inline [
id, date, amount
15, 10/01/2018, 164.27
16, 10/03/2018, 384.00
17, 10/06/2018, 25.82
18, 10/09/2018, 312.00
19, 10/15/2018, 4.56
20, 10/16/2018, 90.24
21, 10/18/2018, 19.32
];

exit script unless NoOfRows('Transactions') < 10 ;
```

Результаты

Загрузите данные и откройте лист. Создайте новую таблицу и добавьте эти поля как измерения:

- id
- date
- amount

Результирующая таблица

id	date	amount
1	08/30/2018	23.56
2	09/07/2018	556.31
3	09/16/2018	5.75
4	09/22/2018	125.00
5	09/22/2018	484.21
6	09/22/2018	59.18
7	09/23/2018	177.42
8	10/01/2018	164.27
9	10/03/2018	384.00
10	10/06/2018	25.82
11	10/09/2018	312.00
12	10/15/2018	4.56
13	10/16/2018	90.24
14	10/18/2018	19.32

В каждом наборе данных из скрипта загрузки имеется семь записей.

Первый набор данных (с id транзакции от 1 до 7) загружается в приложение. Условие Unless проверяет, содержится ли в таблице Transactions меньше 10 строк. Возвращается значение TRUE, поэтому в приложение загружается второй набор данных (с id транзакции от 8 до 14). Второе условие Unless проверяет, содержится ли в таблице Transactions меньше 10 записей. Возвращается значение FALSE, поэтому выполнение скрипта прекращается.

Пример 3. Несколько префиксов Unless

Скрипт загрузки и результаты

Обзор

Откройте редактор загрузки данных и добавьте приведенный ниже скрипт загрузки на новую вкладку.

В этом примере набор данных, содержащий одну транзакцию, создается в таблице под именем Transactions. Затем запускается цикл for, в котором проверяются два вложенных оператора unless:

1. Если в таблице Transactions не больше 100 записей
2. Если количество записей в таблице Transactions не кратно 6

Если эти условия возвращают результат FALSE, еще семь записей создаются и объединяются с существующей таблицей Transactions. Этот процесс повторяется, пока одна из двух транзакций не вернет значение TRUE.

Скрипт загрузки

```
Transactions:
Load
        0 as id
Autogenerate 1;

For i = 1 to 100
        unless NoOfRows('Transactions') > 100 unless mod(NoOfRows('Transactions'),6) = 0
                Concatenate
      Load
if(isnull(Peek(id)),1,peek(id)+1) as id
               Autogenerate 7;
next i
```

Результаты

Загрузите данные и откройте лист. Создайте новую таблицу и добавьте это поле как измерение:id.

Результирующая таблица

id
0
1
2
3
4
5
+30 строк

Вложенные операторы unless в цикле for проверяют следующие условия:

1. Содержит ли таблица `Transactions` больше 100 строк?
2. Является ли количество записей в таблице `Transactions` кратным 6?

Когда оба оператора unless возвращают значение `FALSE`, еще семь записей создаются и объединяются с существующей таблицей `Transactions`.

Эти операторы возвращают значение `FALSE` пять раз, после чего таблица `Transactions` содержит всего 36 строк данных.

После этого второй оператор `Unless` возвращает значение `TRUE`, и поэтому следующий оператор load уже не выполняется.

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 )
```

Возвращаемые типы данных: Булево значение

В Qlik Sense логическое значение «истина» представлено как -1, а «ложь» – как 0.

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`.

Когда это следует использовать

Оператор `When` возвращает результат в виде булева значения. Как правило, этот тип функции используется в качестве условия, когда пользователю требуется выполнить загрузку или исключить части скрипта.

Аргументы

Аргумент	Описание
condition	Логическое выражение, возвращающее 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**.

Региональные настройки

Если не указано иное, в примерах, приведенных в данном разделе, используется следующий формат даты: ММ/ДД/ГГГГ. Формат даты указан в операторе `SET DateFormat` скрипта загрузки данных. В вашей системе может быть установлен другой формат даты по умолчанию в зависимости от региональных настроек и других факторов. Можно изменить формат в примерах в соответствии с потребностями. Или можно изменить форматы в скрипте загрузки в соответствии с этими примерами.

Региональные настройки по умолчанию в приложениях основаны на системных региональных настройках компьютера или сервера, где установлено ПО Qlik Sense. Если на сервере Qlik Sense, к которому обращается пользователь, выбран шведский язык, то редактор загрузки данных будет использовать шведские региональные настройки для даты, времени и валюты. Эти параметры регионального формата не связаны с языком, отображаемым в интерфейсе пользователя Qlik Sense. Qlik Sense будет отображаться на языке, который используется в браузере.

Примеры функции

Пример	Результат
`exit script when A=1;`	Когда оператор `A=1` возвращает TRUE, выполнение скрипта прекращается.
`when A=1 LOAD * from myfile.csv;`	Когда оператор `A=1` возвращает TRUE, выполняется загрузка `myfile.csv`.
`when A=1 unless B=2 drop table Tab1;`	Когда оператор `A=1` возвращает TRUE, а `B=2` возвращает FALSE, то таблица `Tab1` удаляется.

Пример 1. Префикс When

Скрипт загрузки и результаты

Обзор

Откройте редактор загрузки данных и добавьте приведенный ниже скрипт загрузки на новую вкладку.

Скрипт загрузки содержит следующее:

- Набор данных с датами и суммами, который отправляется в таблицу под именем `Transactions`.
- Оператор `Let`, который объявляет создание переменной `A` и имеет значение `1`.
- Условие `When`, согласно которому скрипт продолжает загрузку, если `A = 1`.

Скрипт загрузки

```
LET A = 1;

WHEN A = 1

Transactions:
LOAD
*
Inline [
id, date, amount
1, 08/30/2018, 23.56
2, 09/07/2018, 556.31
3, 09/16/2018, 5.75
4, 09/22/2018, 125.00
5, 09/22/2018, 484.21
6, 09/22/2018, 59.18
7, 09/23/2018, 177.42
];
```

Результаты

Загрузите данные и откройте лист. Создайте новую таблицу и добавьте эти поля как измерения:

- `id`
- `date`
- `amount`

Результирующая таблица

id	date	amount
1	08/30/2018	23.56

id	date	amount
2	09/07/2018	556.31
3	09/16/2018	5.75
4	09/22/2018	125.00
5	09/22/2018	484.21
6	09/22/2018	59.18
7	09/23/2018	177.42

Так как в начале скрипта переменной `A` задается значение `1`, проверяется выполнение условия после префикса `When` и возвращается результат TRUE. Так как возвращен результат TRUE, скрипт продолжает выполнение оператора load. Отображаются все записи из таблицы.

Если этой переменной задано значение, не равное `1`, данные не будут загружаться в модель данных.

Пример 2. Суффикс When

Скрипт загрузки и результаты

Обзор

Откройте редактор загрузки данных и добавьте приведенный ниже скрипт загрузки на новую вкладку.

Скрипт загрузки содержит следующее:

- Три набора данных с датами и суммами, которые отправляются в таблицу под именем `Transactions`.
  - Первый набор данных содержит транзакции 1-7.
  - Второй набор данных содержит транзакции 8-14.
  - Третий набор данных содержит транзакции 15-21.
- Условие `When`, которое определяет, содержит ли таблица `Transactions` больше 10 строк. Если любой из операторов `When` возвращает TRUE, выполнение скрипта загрузки прекращается. Это условие помещается в конце каждого из трех наборов данных.

Скрипт загрузки

```
Transactions:
LOAD
*
Inline [
id, date, amount
1, 08/30/2018, 23.56
2, 09/07/2018, 556.31
3, 09/16/2018, 5.75
4, 09/22/2018, 125.00
```

```
5, 09/22/2018, 484.21
6, 09/22/2018, 59.18
7, 09/23/2018, 177.42
];

exit script when NoOfRows('Transactions') > 10 ;

Concatenate
LOAD
*
Inline [
id, date, amount
8, 10/01/2018, 164.27
9, 10/03/2018, 384.00
10, 10/06/2018, 25.82
11, 10/09/2018, 312.00
12, 10/15/2018, 4.56
13, 10/16/2018, 90.24
14, 10/18/2018, 19.32
];

exit script when NoOfRows('Transactions') > 10 ;

Concatenate
LOAD
*
Inline [
id, date, amount
15, 10/01/2018, 164.27
16, 10/03/2018, 384.00
17, 10/06/2018, 25.82
18, 10/09/2018, 312.00
19, 10/15/2018, 4.56
20, 10/16/2018, 90.24
21, 10/18/2018, 19.32
];

exit script when NoOfRows('Transactions') > 10 ;
```

Результаты

Загрузите данные и откройте лист. Создайте новую таблицу и добавьте эти поля как измерения:

- id
- date
- amount

Результирующая таблица

id	date	amount
1	08/30/2018	23.56

id	date	amount
2	09/07/2018	556.31
3	09/16/2018	5.75
4	09/22/2018	125.00
5	09/22/2018	484.21
6	09/22/2018	59.18
7	09/23/2018	177.42
8	10/01/2018	164.27
9	10/03/2018	384.00
10	10/06/2018	25.82
11	10/09/2018	312.00
12	10/15/2018	4.56
13	10/16/2018	90.24
14	10/18/2018	19.32

Каждый из трех наборов данных содержит семь транзакций. Первый набор данных содержит транзакции 1-7 и загружается в приложение. Условие `When`, которое следует за этим оператором load, возвращает FALSE, так как таблица `Transactions` содержит меньше 10 строк. Скрипт загрузки переходит к следующему набору данных.

Второй набор данных содержит транзакции 8-14 и загружается в приложение. Второе условие `When` возвращает TRUE, так как таблица `Transactions` содержит больше 10 строк. Поэтому выполнение скрипта прекращается.

Пример 3. Несколько префиксов When

Скрипт загрузки и результаты

Обзор

Откройте редактор загрузки данных и добавьте приведенный ниже скрипт загрузки на новую вкладку.

Скрипт загрузки содержит следующее:

- Набор данных, содержащий одну транзакцию, создается как таблица под именем `Transactions`.
- Запущенный цикл `For` содержит два вложенных условия `When`, которые проверяют следующее:
    1. В таблице `Transactions` не больше `100`записей.
    2. Количество записей в таблице `Transactions` не кратно `6`.

Скрипт загрузки

```
Transactions:
Load
       0 as id
Autogenerate 1;

For i = 1 to 100
       when NoOfRows('Transactions') < 100 when mod(NoOfRows('Transactions'),6) <> 0
               Concatenate
    Load
       if(isnull(Peek(id)),1,peek(id)+1) as id
      Autogenerate 7;
next i
```

Результаты

Загрузите данные и откройте лист. Создайте новую таблицу и добавьте это поле как измерение:

- id

В таблице результатов отображаются только первые пять идентификаторов транзакций, но скрипт загрузки создает 36 строк, а затем завершает работу, как только выполняется условие `When`.

Результирующая таблица

id
0
1
2
3
4
5
+30 строк

Вложенные условия `When` в цикле `For` проверяют ответы на следующие вопросы:

- Содержит ли таблица `Transactions` меньше 100 строк?
- Не является ли количество записей в таблице `Transactions` кратным 6?

Когда оба оператора `When` возвращают значение TRUE, еще семь записей создаются и объединяются с существующей таблицей `Transactions`.

Условия `When` возвращают TRUE пять раз. На этом этапе таблица `Transactions` содержит всего 36 строк данных.

После того как в таблице `Transactions` создано 36 строк, второй оператор `When` возвращает значение FALSE и поэтому следующий оператор load больше не выполняется.

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
```

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 585)* 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.

Аргумент	Описание
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

KeyToOtherTable	Region	Year	Month	Sales
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
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:`

Аргумент	Описание
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;`

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:

p *Load (page 154)*

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:

p *Load (page 154)*

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:

p *Load (page 154)*

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:

p *Load (page 154)*

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:

p *Load (page 154)*

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:

p *Load (page 154)*

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:

p *Load (page 154)*

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:

p *Load (page 154)*

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:

p *Load (page 154)*

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 274)*
- *Переменные интерпретации числа (page 208)*
- *Системные переменные (page 199)*
- *Переменные обработки значений (page 206)*

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 определяются системой. Переменные указываются в верхней части скрипта загрузки и применяют настройки форматирования чисел время выполнения скрипта. Эти переменные можно удалять, редактировать или копировать.

Переменные интерпретации числа создаются автоматически при создании нового приложения в соответствии с текущими региональными настройками операционной системы. В Qlik Sense Desktop они соответствуют настройкам операционной системы компьютера. В Qlik Sense эти настройки соответствуют операционной системе сервера, на котором установлена программа Qlik Sense. Если на сервере Qlik Sense, к которому обращается пользователь, выбран шведский язык, то редактор загрузки данных будет использовать шведские региональные настройки для даты, времени и валюты. Эти параметры регионального формата не связаны с языком, отображаемым в интерфейсе пользователя Qlik Sense. Qlik Sense будет отображаться на языке, который используется в браузере.

Форматирование валюты

MoneyDecimalSep

Указанный десятичный разделитель заменяет символ десятичного знака для денежных сумм, заданный в региональных настройках.

```
MoneyDecimalSep
```

MoneyFormat

Указанный символ заменяет обозначение валюты, заданное в региональных настройках.

```
MoneyFormat
```

MoneyThousandSep

Указанный разделитель тысяч заменяет группирующий символ знаков для денежных сумм, заданный в региональных настройках.

```
MoneyThousandSep
```

Форматирование чисел

DecimalSep

Указанный десятичный разделитель заменяет символ десятичного знака, заданный в региональных настройках.

```
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 имена месяцев, заданные в региональных настройках.

MonthNames

LongMonthNames

Указанный формат заменяет полные имена месяцев, заданные в региональных настройках.

LongMonthNames

DayNames

Указанный формат заменяет имена дней недели, заданные в региональных настройках.

DayNames

LongDayNames

Указанный формат заменяет полные имена дней недели, заданные в региональных настройках.

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)

Региональные настройки

Если не указано иное, в примерах, приведенных в данном разделе, используется следующий формат даты: ММ/ДД/ГГГГ. Формат даты указан в операторе `SET DateFormat` скрипта загрузки данных. В вашей системе может быть установлен другой формат даты по умолчанию в зависимости от региональных настроек и других факторов. Можно изменить формат в примерах в соответствии с потребностями. Или можно изменить форматы в скрипте загрузки в соответствии с этими примерами.

Региональные настройки по умолчанию в приложениях основаны на системных региональных настройках компьютера или сервера, где установлено ПО Qlik Sense. Если на сервере Qlik Sense, к которому обращается пользователь, выбран шведский язык, то редактор загрузки данных будет использовать шведские региональные настройки для даты, времени и валюты. Эти параметры регионального формата не связаны с языком, отображаемым в интерфейсе пользователя Qlik Sense. Qlik Sense будет отображаться на языке, который используется в браузере.

Примеры:

```
Set BrokenWeeks=0; //(use unbroken weeks)
Set BrokenWeeks=1; //(use broken weeks)
```

DateFormat

Эта переменная среды определяет формат даты, используемый по умолчанию в приложении, а также используемый функциями возврата даты, такими как `date()` и

`date#()`. Формат используется для интерпретации и форматирования дат. Если переменная не определена, при выполнении скрипта извлекается формат даты, заданный региональными настройками.

Синтаксис:

```
DateFormat
```

Примеры функции DateFormat

Пример	Результат
`Set DateFormat='M/D/YY'; //(US format)`	Это использование функции `DateFormat` определяет дату в формате США – «месяц/дата/год».
`Set DateFormat='DD/MM/YY'; //(UK date format)`	Это использование функции `DateFormat` определяет дату в формате Соединенного Королевства – «дата/месяц/год».
`Set DateFormat='YYYY/MM/DD'; //(ISO date format)`	Это использование функции `DateFormat` определяет дату в формате ISO – «год/месяц/дата».

Региональные настройки

Если не указано иное, в примерах, приведенных в данном разделе, используется следующий формат даты: ММ/ДД/ГГГГ. Формат даты указан в операторе `SET DateFormat` скрипта загрузки данных. В вашей системе может быть установлен другой формат даты по умолчанию в зависимости от региональных настроек и других факторов. Можно изменить формат в примерах в соответствии с потребностями. Или можно изменить форматы в скрипте загрузки в соответствии с этими примерами.

Региональные настройки по умолчанию в приложениях основаны на системных региональных настройках компьютера или сервера, где установлено ПО Qlik Sense. Если на сервере Qlik Sense, к которому обращается пользователь, выбран шведский язык, то редактор загрузки данных будет использовать шведские региональные настройки для даты, времени и валюты. Эти параметры регионального формата не связаны с языком, отображаемым в интерфейсе пользователя Qlik Sense. Qlik Sense будет отображаться на языке, который используется в браузере.

Пример 1. Системные переменные по умолчанию

Скрипт загрузки и результаты

Обзор

Откройте редактор загрузки данных и добавьте приведенный ниже скрипт загрузки на новую вкладку.

Скрипт загрузки содержит следующее:

- Набор данных с датами.
- Функция `DateFormat`, которая будет использовать формат даты США.

В этом примере набор данных загружается в таблицу под именем `Transactions`. Она включает поле `date`. Используется определение `DateFormat` США. Этот шаблон будет использоваться для неявного преобразования текста в дату при загрузке текстовых дат.

Скрипт загрузки

```
Set DateFormat='MM/DD/YYYY';

Transactions:
LOAD
date,
month(date) as month,
id,
amount
INLINE
[
date,id,amount
01/01/2022,1,1000
02/01/2022,2,2123
03/01/2022,3,4124
04/01/2022,4,2431
];
```

Результаты

Загрузите данные и откройте лист. Создайте новую таблицу и добавьте эти поля как измерения:

- `date`
- `month`

Создайте это измерение:

```
=sum(amount)
```

Результирующая таблица

date	month	=sum(amount)
01/01/2022	Jan	1000
02/01/2022	Feb	2123
03/01/2022	Mar	4124
04/01/2022	Apr	2431

Определение `DateFormat` MM/DD/YYYY используется для неявного преобразования текста в даты. Именно поэтому поле `date` правильно интерпретируется как дата. Тот же формат используется для отображения даты, как показано в результатах таблицы.

Пример 2. Изменение системной переменной

Скрипт загрузки и результаты

Обзор

Откройте редактор загрузки данных и добавьте приведенный ниже скрипт загрузки на новую вкладку.

Скрипт загрузки содержит следующее:

- Тот же набор данных из предыдущего примера.
- Функция `DateFormat`, которая будет использовать формат DD/MM/YYYY.

Скрипт загрузки

```
SET DateFormat='DD/MM/YYYY';
Transactions:
LOAD
date,
month(date) as month,
id,
amount
INLINE
[
date,id,amount
01/01/2022,1,1000
02/01/2022,2,2123
03/01/2022,3,4124
04/01/2022,4,2431
];
```

Результаты

Загрузите данные и откройте лист. Создайте новую таблицу и добавьте эти поля как измерения:

- `date`
- `month`

Создайте это измерение:

```
=sum(amount)
```

Результирующая таблица

date	month	=sum(amount)
01/01/2022	Jan	1000
02/01/2022	Jan	2123
03/01/2022	Jan	4124
04/01/2022	Jan	2431

Так как для `DateFormat` задано определение DD/MM/YYYY, мы видим, что две цифры после первой косой черты (/) интерпретированы как месяц, таким образом, все записи относятся к январю.

Пример 3. Интерпретация данных

Скрипт загрузки и результаты

Обзор

Откройте редактор загрузки данных и добавьте приведенный ниже скрипт загрузки на новую вкладку.

Скрипт загрузки содержит следующее:

- Набор данных с датами в числовом формате.
- Переменная `DateFormat`, которая будет использовать формат DD/MM/YYYY.
- Переменная `date()`.

Скрипт загрузки

```
SET DateFormat='MM/DD/YYYY';

Transactions:
Load
date(numerical_date),
month(date(numerical_date)) as month,
id,
amount
Inline
[
numerical_date,id,amount
43254,1,1000
43255,2,2123
43256,3,4124
43258,4,2431
];
```

Результаты

Загрузите данные и откройте лист. Создайте новую таблицу и добавьте эти поля как измерения:

- `date`
- `month`

Создайте это измерение:

```
=sum(amount)
```

Результирующая таблица

date	month	=sum(amount)
06/03/2022	Jun	1000
06/04/2022	Jun	2123

date	month	=sum(amount)
06/05/2022	Jun	4124
06/07/2022	Jun	2431

В скрипте загрузки используется функция `date()` для преобразования числовой даты в формат даты. Так как не предоставлен определенный формат для второго аргумента функции, используется `DateFormat`. В результате этого для поля даты используется формат MM/DD/YYYY.

Пример 4. Иностранный формат даты

Скрипт загрузки и результаты

Обзор

Откройте редактор загрузки данных и добавьте приведенный ниже скрипт загрузки на новую вкладку.

Скрипт загрузки содержит следующее:

- Набор данных с датами.
- Переменная `DateFormat`, которая использует формат DD/MM/YYYY , но раскомментирована косой чертой.

Скрипт загрузки

```
// SET DateFormat='DD/MM/YYYY';


Transactions:
Load
date,
month(date) as month,
id,
amount
Inline
[
date,id,amount
22-05-2022,1,1000
23-05-2022,2,2123
24-05-2022,3,4124
25-05-2022,4,2431
];
```

Результаты

Загрузите данные и откройте лист. Создайте новую таблицу и добавьте эти поля как измерения:

- `date`
- `month`

Создайте это измерение:

```
=sum(amount)
```

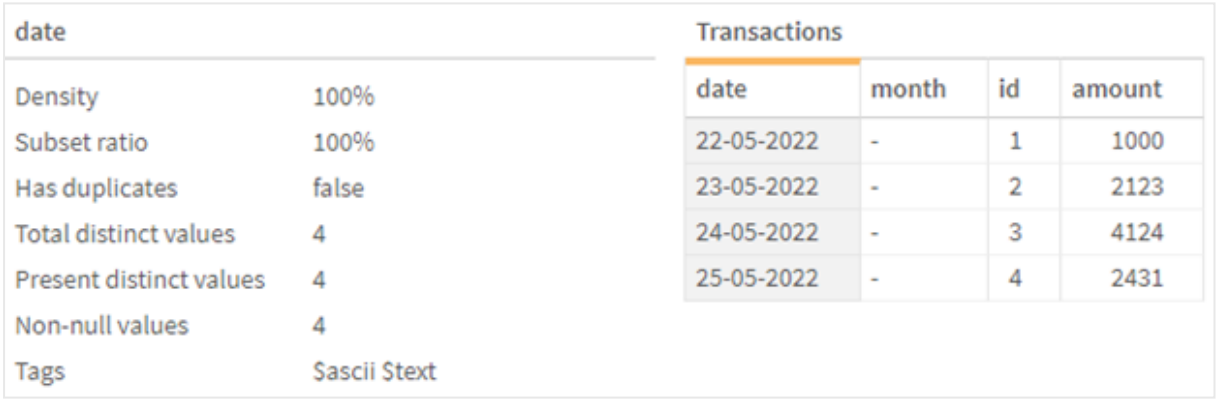
Результирующая таблица

date	month	=sum(amount)
22-05-2022	-	1000
23-05-2022	-	2123
24-05-2022	-	4124
25-05-2022	-	2431

В начальном скрипте загрузки для `DateFormat` по умолчанию используется MM/DD/YYYY. Так как поле `date` в наборе данных transactions имеет другой формат, оно не интерпретируется как дата. Это видно в таблице результатов, где в поле `month` отображаются значения null.

Можно проверить интерпретируемые типы данных в окне просмотра модели данных: обратите внимание на свойства Tags поля `date`:

Предварительный просмотр таблицы `Transactions`. Обратите внимание на свойства Tags для поля `date`, указывающие на то, что текстовые входные данные не были неявно преобразованы в дату/метку времени.



Это можно решить, включив системную переменную `DateFormat`:

```
// SET DateFormat='DD/MM/YYYY';
```

Удалите двойную косую черту и перезагрузите данные.

Предварительный просмотр таблицы `Transactions`. Обратите внимание на свойства Tags для поля `date`, указывающие на то, что текстовые входные данные были неявно преобразованы в дату/метку времени.

date

Density	100%
Subset ratio	100%
Has duplicates	false
Total distinct values	4
Present distinct values	4
Non-null values	4
Tags	$numeric $integer $timestamp $date

Transactions

date	month	id	amount
22-05-2022	May	1	1000
23-05-2022	May	2	2123
24-05-2022	May	3	4124
25-05-2022	May	4	2431

DayNames

Указанный формат заменяет имена дней недели, заданные в региональных настройках.

Синтаксис:

```
DayNames
```

При модификации переменной необходимо использовать точку с запятой ; для разделения отдельных значений.

Примеры функции DayName

Пример функции	Определение результата
`Set DayNames='Mon;Tue;Wed;Thu;Fri;Sat;Sun';`	Это использование функции DayNames определяет названия дней в сокращенной форме.
`Set DayNames='M;Tu;W;Th;F;Sa;Su';`	Это использование функции DayNames определяет названия дней по первым буквам.

Функция `DayNames` часто используется в сочетании со следующими функциями:

Связанные функции

Функция	Взаимодействие
weekday (page 1088)	Функция скрипта для возврата `DayNames` в качестве значений поля.
Date (page 1248)	Функция Script для возврата `DayNames` в качестве значений полей.
LongDayNames (page 229)	Значения длинной формы для `DayNames`.

Региональные настройки

Если не указано иное, в примерах, приведенных в данном разделе, используется следующий формат даты: ММ/ДД/ГГГГ. Формат даты указан в операторе `SET DateFormat` скрипта загрузки данных. В вашей системе может быть установлен другой формат даты по умолчанию в

зависимости от региональных настроек и других факторов. Можно изменить формат в примерах в соответствии с потребностями. Или можно изменить форматы в скрипте загрузки в соответствии с этими примерами.

Региональные настройки по умолчанию в приложениях основаны на системных региональных настройках компьютера или сервера, где установлено ПО Qlik Sense. Если на сервере Qlik Sense, к которому обращается пользователь, выбран шведский язык, то редактор загрузки данных будет использовать шведские региональные настройки для даты, времени и валюты. Эти параметры регионального формата не связаны с языком, отображаемым в интерфейсе пользователя Qlik Sense. Qlik Sense будет отображаться на языке, который используется в браузере.

Пример 1. Системные переменные по умолчанию

Скрипт загрузки и результаты

Обзор

В этом примере даты в наборе данных заданы в формате MM/DD/YYYY.

Откройте редактор загрузки данных и добавьте приведенный ниже скрипт загрузки на новую вкладку.

Скрипт загрузки содержит следующее:

- Набор данных с датами, который будет загружен в таблицу с именем `Transactions`.
- Поле `date`.
- Определение `DayNames` по умолчанию.

Скрипт загрузки

```
SET DayNames='Mon;Tue;Wed;Thu;Fri;Sat;Sun';

Transactions:
LOAD
date,
WeekDay(date) as dayname,
id,
amount
INLINE
[
date,id,amount
01/01/2022,1,1000
02/01/2022,2,2123
03/01/2022,3,4124
04/01/2022,4,2431
];
```

Результаты

Загрузите данные и откройте лист. Создайте новую таблицу и добавьте эти поля как измерения:

- date
- dayname

Создайте эту меру:

```
sum(amount)
```

Результирующая таблица

date	dayname	sum(amount)
01/01/2022	Sat	1000
02/01/2022	Tue	2123
03/01/2022	Tue	4124
04/01/2022	Fri	2431

В скрипте загрузки функция WeekDay используется с полем date в качестве предоставленного аргумента. В таблице результатов выходные данные этой функции WeekDay отображают дни недели в формате определения DayNames.

Пример 2. Изменение системной переменной

Скрипт загрузки и результаты

Обзор

Откройте редактор загрузки данных и добавьте приведенный ниже скрипт загрузки на новую вкладку. Используется тот же набор данных и сценарий, что в первом примере.

Однако в начале скрипта определение DayNames изменено для использования сокращенных названий дней недели на языке африкаанс.

Скрипт загрузки

```
SET DayNames='Ma;Di;Wo;Do;Vr;Sa;So';

Transactions:
Load
date,
WeekDay(date) as dayname,
id,
amount
Inline
[
date,id,amount
01/01/2022,1,1000
02/01/2022,2,2123
03/01/2022,3,4124
04/01/2022,4,2431
];
```

Результаты

Загрузите данные и откройте лист. Создайте новую таблицу и добавьте эти поля как измерения:

- `date`
- `dayname`

Создайте это измерение:

```
sum(amount)
```

Результирующая таблица

date	dayname	sum(amount)
01/01/2022	Sa	1000
02/01/2022	Di	2123
03/01/2022	Di	4124
04/01/2022	Vr	2431

В таблице результатов выходные данные этой функции `WeekDay` отображают дни недели в формате определения `DayNames`.

Важно помнить, что если язык для `DayNames` изменяется так, как в этом примере, `LongDayNames` все равно будет содержать дни недели на английском языке. Это также необходимо изменить, если в приложении используются обе переменные.

Пример 3. Функция даты

Скрипт загрузки и результаты

Обзор

Откройте редактор загрузки данных и добавьте приведенный ниже скрипт загрузки на новую вкладку.

Скрипт загрузки содержит следующее:

- Набор данных с датами, который будет загружен в таблицу с именем `Transactions`.
- Поле `date`.
- Определение `DayNames` по умолчанию.

Скрипт загрузки

```
SET DayNames='Mon;Tue;Wed;Thu;Fri;Sat;Sun';

Transactions:
Load
date,
Date(date,'WWW') as dayname,
```

```
id,
amount
Inline
[
date,id,amount
01/01/2022,1,1000
02/01/2022,2,2123
03/01/2022,3,4124
04/01/2022,4,2431
];
```

Результаты

Загрузите данные и откройте лист. Создайте новую таблицу и добавьте эти поля как измерения:

- `date`
- `dayname`

Создайте это измерение:

```
sum(amount)
```

Результирующая таблица

date	dayname	sum(amount)
01/01/2022	Sat	1000
02/01/2022	Tue	2123
03/01/2022	Tue	4124
04/01/2022	Fri	2431

По умолчанию используется определение `DayNames`. В скрипте загрузки функция `Date` используется вместе с полем `date`, предоставленном в качестве первого аргумента. Второй аргумент — `WWW`. Это форматирование преобразует результат в значения, сохраненные в определении `DayNames`. Это отображается в выводе таблицы результатов.

DecimalSep

Указанный десятичный разделитель заменяет символ десятичного знака, заданный в региональных настройках.

Qlik Sense автоматически интерпретирует текст как числа каждый раз, когда встречается распознаваемый числовой шаблон. Системные переменные `ThousandSep` и `DecimalSep` определяют макет шаблонов, применяемых при синтаксическом анализе текста как чисел. Переменные `ThousandSep` и `DecimalSep` задают шаблон числового формата по умолчанию при визуализации числового содержимого в интерфейсных диаграммах и таблицах. То есть это напрямую влияет на параметры **форматирования чисел** для любого интерфейсного выражения.

При использовании запятой «`,`» в качестве разделителя тысяч и «`.`» в качестве десятичного разделителя это примеры шаблонов, которые будут неявно преобразованы в числовые эквивалентные значения:

0,000.00

0000.00

0,000

Это примеры шаблонов, которые останутся текстом в неизмененном виде; то есть не будут преобразованы в числовой формат:

0.000,00

0,00

Синтаксис:

DecimalSep

Примеры функции

Пример	Результат
Set DecimalSep='.';	Задает «.» в качестве десятичного разделителя.
Set DecimalSep=',';	Задает «.» в качестве десятичного разделителя.

Региональные настройки

Если не указано иное, в примерах, приведенных в данном разделе, используется следующий формат даты: ММ/ДД/ГГГГ. Формат даты указан в операторе SET DateFormat скрипта загрузки данных. В вашей системе может быть установлен другой формат даты по умолчанию в зависимости от региональных настроек и других факторов. Можно изменить формат в примерах в соответствии с потребностями. Или можно изменить форматы в скрипте загрузки в соответствии с этими примерами.

Региональные настройки по умолчанию в приложениях основаны на системных региональных настройках компьютера или сервера, где установлено ПО Qlik Sense. Если на сервере Qlik Sense, к которому обращается пользователь, выбран шведский язык, то редактор загрузки данных будет использовать шведские региональные настройки для даты, времени и валюты. Эти параметры регионального формата не связаны с языком, отображаемым в интерфейсе пользователя Qlik Sense. Qlik Sense будет отображаться на языке, который используется в браузере.

Пример. Влияние настройки переменных-разделителей чисел на различные входные данные

Скрипт загрузки и результаты

Обзор

Откройте редактор загрузки данных и добавьте приведенный ниже скрипт загрузки на новую вкладку.

Скрипт загрузки содержит следующее:

- Набор данных сумм и дат с суммами, настроенными с использованием различных шаблонов формата.
- Таблица под именем `Transactions`.
- Переменная `DecimalSep`, для которой задан формат `.`.
- Переменная `ThousandSep`, для которой задан формат `,`.
- Переменная `delimiter`, заданная как символ «|» для разделения различных полей в строке.

Скрипт загрузки

```
Set ThousandSep=',';
Set DecimalSep='.';

Transactions:
Load date,
id,
amount as amount
Inline
[
date|id|amount
01/01/2022|1|1.000-45
01/02/2022|2|23.344
01/03/2022|3|4124,35
01/04/2022|4|2431.36
01/05/2022|5|4,787
01/06/2022|6|2431.84
01/07/2022|7|4132.5246
01/08/2022|8|3554.284
01/09/2022|9|3.756,178
01/10/2022|10|3,454.356
] (delimiter is '|');
```

Результаты

Загрузите данные и откройте лист. Создайте новую таблицу и добавьте это поле как измерение:`amount`

Создайте это измерение:

```
=sum(amount)
```

Результирующая таблица

Amount	=Sum(amount)
Итоги	20814.7086
1.000-45	
3.756,178	
4124,35	
23.344	23.344

Amount	=Sum(amount)
2431.36	2431.36
2431.84	2431.84
3,454.356	3454.356
3554.284	3554.284
4132.5246	4132.5246
4,787	4787

Любое значение, не интерпретированное как число, остается текстом и по умолчанию выравнивается по левому краю. Любые успешно преобразованные значения выравниваются по правому краю, сохраняя исходный формат ввода.

Столбец выражения показывает числовой эквивалент, который по умолчанию отформатирован только с десятичным разделителем «.». Это можно переопределить с помощью раскрывающегося списка **Числовое форматирование** в конфигурации выражения.

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

Региональные настройки

Если не указано иное, в примерах, приведенных в данном разделе, используется следующий формат даты: ММ/ДД/ГГГГ. Формат даты указан в операторе `SET DateFormat` скрипта загрузки данных. В вашей системе может быть установлен другой формат даты по умолчанию в

зависимости от региональных настроек и других факторов. Можно изменить формат в примерах в соответствии с потребностями. Или можно изменить форматы в скрипте загрузки в соответствии с этими примерами.

Региональные настройки по умолчанию в приложениях основаны на системных региональных настройках компьютера или сервера, где установлено ПО Qlik Sense. Если на сервере Qlik Sense, к которому обращается пользователь, выбран шведский язык, то редактор загрузки данных будет использовать шведские региональные настройки для даты, времени и валюты. Эти параметры регионального формата не связаны с языком, отображаемым в интерфейсе пользователя Qlik Sense. Qlik Sense будет отображаться на языке, который используется в браузере.

Пример 1. Использование значения по умолчанию (скрипт)

Скрипт загрузки и результаты

Обзор

Откройте Редактор загрузки данных и добавьте приведенный ниже скрипт загрузки на новую вкладку.

В этом примере скрипт загрузки использует значение системной переменной Qlik Sense по умолчанию, `FirstWeekDay=6`. Этот набор данных содержит данные для первых 14 дней в 2020 году.

Скрипт загрузки

```
// Example 1: Load Script using the default value of FirstWeekDay=6, i.e. Sunday

SET FirstWeekDay = 6;

Sales:
LOAD
        date,
        sales,
        week(date) as week,
        weekday(date) as weekday
Inline [
date,sales
01/01/2021,6000
01/02/2021,3000
01/03/2021,6000
01/04/2021,8000
01/05/2021,5000
01/06/2020,7000
01/07/2020,3000
01/08/2020,5000
01/09/2020,9000
01/10/2020,5000
01/11/2020,7000
01/12/2020,7000
01/13/2020,7000
01/14/2020,7000
];
```

Результаты

Загрузите данные и откройте лист. Создайте новую таблицу и добавьте эти поля как измерения:

- `date`
- `week`
- `weekday`

Результирующая таблица

Date	week	weekday
01/01/2021	1	Wed
01/02/2021	1	Thu
01/03/2021	1	Fri
01/04/2021	1	Sat
01/05/2021	2	Sun
01/06/2020	2	Mon
01/07/2020	2	Tue
01/08/2020	2	Wed
01/09/2020	2	Thu
01/10/2020	2	Fri
01/11/2020	2	Sat
01/12/2020	3	Sun
01/13/2020	3	Mon
01/14/2020	3	Tue

Так как используются настройки по умолчанию, системной переменной `FirstWeekDay` задается значение 6. В таблице results видно, что каждая новая неделя начинается с воскресенья (5-е и 12-е января).

Пример 2. Изменение переменной FirstWeekDay (скрипт)

Скрипт загрузки и результаты

Обзор

Откройте Редактор загрузки данных и добавьте приведенный ниже скрипт загрузки на новую вкладку.

В этом примере набор данных содержит первые 14 дней в 2020 году. В начале скрипта мы задали переменной `FirstWeekDay` значение 3.

Скрипт загрузки

```
// Example 2: Load Script setting the value of FirstWeekDay=3, i.e. Thursday

SET FirstWeekDay = 3;

Sales:
LOAD
       date,
       sales,
       week(date) as week,
       weekday(date) as weekday
Inline [
date,sales
01/01/2021,6000
01/02/2021,3000
01/03/2021,6000
01/04/2021,8000
01/05/2021,5000
01/06/2020,7000
01/07/2020,3000
01/08/2020,5000
01/09/2020,9000
01/10/2020,5000
01/11/2020,7000
01/12/2020,7000
01/13/2020,7000
01/14/2020,7000
];
```

Результаты

Загрузите данные и откройте лист. Создайте новую таблицу и добавьте эти поля как измерения:

- `date`
- `week`
- `weekday`

Результирующая таблица

Date	week	weekday
01/01/2021	52	Wed
01/02/2021	1	Thu
01/03/2021	1	Fri
01/04/2021	1	Sat
01/05/2021	1	Sun
01/06/2020	1	Mon

Date	week	weekday
01/07/2020	1	Tue
01/08/2020	1	Wed
01/09/2020	2	Thu
01/10/2020	2	Fri
01/11/2020	2	Sat
01/12/2020	2	Sun
01/13/2020	2	Mon
01/14/2020	2	Tue

Так как системной переменной `FirstWeekDay` задано значение 3, первым днем каждой недели будет четверг. В таблице results видно, что каждая новая неделя начинается в четверг (2-е и 9-е января).

LongDayNames

Указанный формат заменяет полные имена дней недели, заданные в региональных настройках.

Синтаксис:

```
LongDayNames
```

В следующем примере функция `LongDayNames` определяет полные названия дней недели:

```
Set LongDayNames='Monday;Tuesday;Wednesday;Thursday;Friday;Saturday;Sunday';
```

При модификации переменной необходимо использовать точку с запятой `;` для разделения отдельных значений.

Функцию `LongDayNames` можно использовать в сочетании с функцией *Date (page 1248)*, которая возвращает `DayNames` в качестве значений поля.

Региональные настройки

Если не указано иное, в примерах, приведенных в данном разделе, используется следующий формат даты: ММ/ДД/ГГГГ. Формат даты указан в операторе `SET DateFormat` скрипта загрузки данных. В вашей системе может быть установлен другой формат даты по умолчанию в зависимости от региональных настроек и других факторов. Можно изменить формат в примерах в соответствии с потребностями. Или можно изменить форматы в скрипте загрузки в соответствии с этими примерами.

Региональные настройки по умолчанию в приложениях основаны на системных региональных настройках компьютера или сервера, где установлено ПО Qlik Sense. Если на сервере Qlik Sense, к которому обращается пользователь, выбран шведский язык, то редактор загрузки данных будет использовать шведские региональные настройки для даты, времени и валюты. Эти параметры регионального формата не связаны с языком, отображаемым в интерфейсе пользователя Qlik Sense. Qlik Sense будет отображаться на языке, который используется в браузере.

Пример 1. Системная переменная по умолчанию

Скрипт загрузки и результаты

Обзор

Откройте редактор загрузки данных и добавьте приведенный ниже скрипт загрузки на новую вкладку.

Скрипт загрузки содержит следующее:

- Набор данных с датами, который будет загружен в таблицу с именем `Transactions`.
- Поле `date`.
- Определение `LongDayNames` по умолчанию.

Скрипт загрузки

```
SET LongDayNames='Monday;Tuesday;Wednesday;Thursday;Friday;Saturday;Sunday';

Transactions:
LOAD
date,
Date(date,'WWWW') as dayname,
id,
amount
INLINE
[
date,id,amount
01/01/2022,1,1000
02/01/2022,2,2123
03/01/2022,3,4124
04/01/2022,4,2431
];
```

Результаты

Загрузите данные и откройте лист. Создайте новую таблицу и добавьте эти поля как измерения:

- `date`
- `dayname`

Создайте это измерение:

```
=sum(amount)
```

Результирующая таблица

date	dayname	=sum(amount)
01/01/2022	Суббота	1000
02/01/2022	Вторник	2123

date	dayname	=sum(amount)
03/01/2022	Вторник	4124
04/01/2022	Пятница	2431

В скрипте загрузки для создания поля `dayname` функция `Date` используется вместе с полем `date`, предоставленном в качестве первого аргумента. Второй аргумент функции – форматирование `WWWW`.

Использование этого форматирования преобразует значения из первого аргумента в соответствующее полное название дня, заданного в переменной `LongDayNames`. В таблице результатов это демонстрируют значения созданного нами поля `dayname`.

Пример 2. Изменение системной переменной

Скрипт загрузки и результаты

Обзор

Откройте редактор загрузки данных и добавьте приведенный ниже скрипт загрузки на новую вкладку.

Используется тот же набор данных и сценарий, что в первом примере. Однако в начале скрипта определение `LongDayNames` модифицируется для использования названий дней недели на испанском языке.

Скрипт загрузки

```
SET LongDayNames='Lunes;Martes;Miércoles;Jueves;Viernes;Sábado;Domingo';

Transactions:
LOAD
date,
Date(date,'WWWW') as dayname,
id,
amount
INLINE
[
date,id,amount
01/01/2022,1,1000
02/01/2022,2,2123
03/01/2022,3,4124
04/01/2022,4,2431
];
```

Результаты

Загрузите данные и откройте лист. Создайте новую таблицу и добавьте эти поля как измерения:

- date
- dayname

Создайте это измерение:

```
=sum(amount)
```

Результирующая таблица

date	dayname	=sum(amount)
01/01/2022	Sábado	1000
02/01/2022	Martes	2123
03/01/2022	Martes	4124
04/01/2022	Viernes	2431

В скрипте загрузки переменная LongDayNames изменена, чтобы отображать дни недели на испанском языке.

Затем вы создаете поле dayname, которое представляет собой функцию Date, используемую вместе с полем date в качестве первого аргумента.

Второй аргумент функции – форматирование WWWW. Использование этого форматирования Qlik Sense преобразует значения из первого аргумента в соответствующее полное название дня, заданного в переменной LongDayNames.

В таблице результатов значения полей созданного поля dayname отображают полные названия дней недели на испанском языке.

LongMonthNames

Указанный формат заменяет полные имена месяцев, заданные в региональных настройках.

Синтаксис:

```
LongMonthNames
```

При модификации переменной необходимо использовать ; для разделения отдельных значений.

В следующем примере функция LongMonthNames определяет полные названия месяцев:

```
Set
LongMonthNames='January;February;March;April;May;June;July;August;September;October;November;D
ecember';
```

Функция LongMonthNames часто используется в сочетании со следующими функциями:

Связанные функции

Функция	Взаимодействие
Date (page 1248)	Функция Script для возврата DayNames в качестве значений полей.
LongDayNames (page 229)	Значения длинной формы для DayNames.

Региональные настройки

Если не указано иное, в примерах, приведенных в данном разделе, используется следующий формат даты: ММ/ДД/ГГГГ. Формат даты указан в операторе `SET DateFormat` скрипта загрузки данных. В вашей системе может быть установлен другой формат даты по умолчанию в зависимости от региональных настроек и других факторов. Можно изменить формат в примерах в соответствии с потребностями. Или можно изменить форматы в скрипте загрузки в соответствии с этими примерами.

Региональные настройки по умолчанию в приложениях основаны на системных региональных настройках компьютера или сервера, где установлено ПО Qlik Sense. Если на сервере Qlik Sense, к которому обращается пользователь, выбран шведский язык, то редактор загрузки данных будет использовать шведские региональные настройки для даты, времени и валюты. Эти параметры регионального формата не связаны с языком, отображаемым в интерфейсе пользователя Qlik Sense. Qlik Sense будет отображаться на языке, который используется в браузере.

Пример 1. Системные переменные по умолчанию

Скрипт загрузки и результаты

Обзор

Откройте редактор загрузки данных и добавьте приведенный ниже скрипт загрузки на новую вкладку.

Скрипт загрузки содержит следующее:

- Набор данных, загруженный в таблицу под именем `Transactions`.
- Поле `date`.
- Определение `LongMonthNames` по умолчанию.

Скрипт загрузки

```
SET
LongMonthNames='January;February;March;April;May;June;July;August;September;October;November;D
ecember';

Transactions:
Load
date,
Date(date,'MMMM') as monthname,
id,
amount
Inline
[
date,id,amount
01/01/2022,1,1000.45
01/02/2022,2,2123.34
01/03/2022,3,4124.35
01/04/2022,4,2431.36
01/05/2022,5,4787.78
```

```
01/06/2022,6,2431.84
01/07/2022,7,2854.83
01/08/2022,8,3554.28
01/09/2022,9,3756.17
01/10/2022,10,3454.35
];
```

Результаты

Загрузите данные и откройте лист. Создайте новую таблицу и добавьте эти поля как измерения:

- `date`
- `monthname`

Создайте это измерение:

```
=sum(amount)
```

Результирующая таблица

date	monthname	sum(amount)
01/01/2022	Январь	1000.45
01/02/2022	Январь	2123.34
01/03/2022	Январь	4124.35
01/04/2022	Январь	2431.36
01/05/2022	Январь	4787.78
01/06/2022	Январь	2431.84
01/07/2022	Январь	2854.83
01/08/2022	Январь	3554.28
01/09/2022	Январь	3756.17
01/10/2022	Январь	3454.35

По умолчанию используется определение `LongMonthNames`. В скрипте загрузки для создания поля `month` функция `Date` используется вместе с полем `date`, предоставленном в качестве первого аргумента. Второй аргумент функции – форматирование `MMMM`.

При использовании этого форматирования Qlik Sense преобразует значения из первого аргумента в соответствующее полное название месяца, заданного в переменной `LongMonthNames`. В таблице результатов это демонстрируют значения созданного нами поля `month`.

Пример 2. Изменение системной переменной

Скрипт загрузки и результаты

Обзор

Откройте редактор загрузки данных и добавьте приведенный ниже скрипт загрузки на новую вкладку.

Скрипт загрузки содержит следующее:

- Набор данных, загруженный в таблицу под именем `Transactions`.
- Поле `date`.
- Переменная `LongMonthNames`, модифицированная для использования сокращенных названий дней недели на испанском языке.

Скрипт загрузки

```
SET
LongMonthNames='Enero;Febrero;Marzo;Abril;Mayo;Junio;Julio;Agosto;Septiembre;OctubreNoviembre;
Diciembre';

Transactions:
LOAD
date,
Date(date,'MMMM') as monthname,
id,
amount
INLINE
[
date,id,amount
01/01/2022,1,1000
02/01/2022,2,2123
03/01/2022,3,4124
04/01/2022,4,2431
];
```

Результаты

Загрузите данные и откройте лист. Создайте новую таблицу и добавьте `sum(amount)` в качестве меры и эти поля в качестве измерений:

- `date`
- `monthname`

Создайте это измерение:

```
=sum(amount)
```

Результирующая таблица

date	monthname	sum(amount)
01/01/2022	Enero	1000.45
01/02/2022	Enero	2123.34

date	monthname	sum(amount)
01/03/2022	Enero	4124.35
01/04/2022	Enero	2431.36
01/05/2022	Enero	4787.78
01/06/2022	Enero	2431.84
01/07/2022	Enero	2854.83
01/08/2022	Enero	3554.28
01/09/2022	Enero	3756.17
01/10/2022	Enero	3454.35

В скрипте загрузки переменная `LongMonthNames` модифицируется для перечисления месяцев года на испанском языке. Затем для создания поля `monthname` функция `Date` используется вместе с полем `date` в качестве первого аргумента. Второй аргумент функции – форматирование `MMMM`.

При использовании этого форматирования Qlik Sense преобразует значения из первого аргумента в соответствующее полное название месяца, заданного в переменной `LongMonthNames`. В таблице результатов значения созданного нами поля `monthname` отображают название месяца на испанском языке.

MoneyDecimalSep

Указанный десятичный разделитель заменяет символ десятичного знака для денежных сумм, заданный в региональных настройках.

По умолчанию Qlik Sense по-разному отображает числа и текст в диаграммах таблицы. Числа выравниваются по правому краю, а текст по левому. Это позволяет легко выявлять проблемы преобразования текста в числа. Все таблицы на этой странице, в которых отображаются результаты Qlik Sense, будут использовать это форматирование.

Синтаксис:

```
MoneyDecimalSep
```

Приложения Qlik Sense будут интерпретировать текстовые поля, соответствующие этому формату, как денежные значения. Текстовое поле должно содержать символ валюты, определенный в системной переменной `MoneyFormat`. `MoneyDecimalSep` особенно полезно при работе с источниками данных, полученными из систем с разными региональными настройками.

Следующий пример демонстрирует возможное использование системной переменной `MoneyDecimalSep`:

```
Set MoneyDecimalSep='.';
```

Эта функция часто используется вместе со следующими функциями:

Связанные функции

Функция	Взаимодействие
MoneyFormat	В случаях интерпретации текстового поля символ MoneyFormat будет использоваться как часть такой интерпретации. В качестве формата чисел будет использоваться MoneyFormat для Qlik Sense в объектах диаграммы.
MoneyThousandSep	В случаях интерпретации текстового поля также необходимо использовать функцию MoneyThousandSep.

Региональные настройки

Если не указано иное, в примерах, приведенных в данном разделе, используется следующий формат даты: ММ/ДД/ГГГГ. Формат даты указан в операторе SET DateFormat скрипта загрузки данных. В вашей системе может быть установлен другой формат даты по умолчанию в зависимости от региональных настроек и других факторов. Можно изменить формат в примерах в соответствии с потребностями. Или можно изменить форматы в скрипте загрузки в соответствии с этими примерами.

Региональные настройки по умолчанию в приложениях основаны на системных региональных настройках компьютера или сервера, где установлено ПО Qlik Sense. Если на сервере Qlik Sense, к которому обращается пользователь, выбран шведский язык, то редактор загрузки данных будет использовать шведские региональные настройки для даты, времени и валюты. Эти параметры регионального формата не связаны с языком, отображаемым в интерфейсе пользователя Qlik Sense. Qlik Sense будет отображаться на языке, который используется в браузере.

Пример 1. Использование точки (.) в MoneyDecimalSep

Скрипт загрузки и результаты

Обзор

Откройте редактор загрузки данных и добавьте приведенный ниже скрипт загрузки на новую вкладку.

Скрипт загрузки содержит следующее:

- Набор данных, загруженный в таблицу под именем Transactions.
- Предоставленные данные, содержащие денежное поле в текстовом формате с использованием точки (.) в качестве десятичного разделителя. Каждая запись также имеет префикс в виде символа «$», кроме последней записи, в которой используется префикс «£».

Помните, что системная переменная MoneyFormat определяет доллар ($) в качестве валюты по умолчанию.

Скрипт загрузки

```
SET MoneyThousandSep=',';
SET MoneyDecimalSep='.';
SET MoneyFormat='$###0.00;-$###0.00';
```

```
Transactions:
Load
date,
id,
amount
Inline
[
date,id,amount
01/01/2022,1,'$14.41'
01/02/2022,2,'$2,814.32'
01/03/2022,3,'$249.36'
01/04/2022,4,'$24.37'
01/05/2022,5,'$7.54'
01/06/2022,6,'$243.63'
01/07/2022,7,'$545.36'
01/08/2022,8,'$3.55'
01/09/2022,9,'$3.436'
01/10/2022,10,'£345.66'
];
```

Результаты

Загрузите данные и откройте лист. Создайте новую таблицу и добавьте это поле как измерение:`amount`.

Добавьте следующие меры:

- `isNum(amount)`
- `sum(amount)`

Ниже приводятся результаты, демонстрирующие правильную интерпретацию только всех значений с символом доллара ($).

Результирующая таблица

amount	=isNum(amount)	=Sum(amount)
Итоги	0	$3905.98
£345.66	0	$0.00
$3.436	-1	$3.44
$3.55	-1	$3.55
$7.54	-1	$7.54
$14.41	-1	$14.41
$24.37	-1	$24.37
243.63	-1	$243.63
$249.36	-1	$249.36

amount	=isNum(amount)	=Sum(amount)
$545.36	-1	$545.36
$2,814.32	-1	$2814.32

Приведенные выше результаты показывают, что поле `amount` правильно интерпретируется для всех значений с символом доллара ($), в то время как при использовании символа фунта (£) `amount` не преобразуется в денежное значение.

Пример 2. Использование запятой (,) в MoneyDecimalSep

Скрипт загрузки и результаты

Обзор

Откройте редактор загрузки данных и добавьте приведенный ниже скрипт загрузки на новую вкладку.

Скрипт загрузки содержит следующее:

- Набор данных, загруженный в таблицу под именем `Transactions`.
- Предоставленные данные, содержащие денежное поле в текстовом формате с использованием запятой (,) в качестве десятичного разделителя. Каждая запись также имеет префикс в виде символа «$», кроме последней записи, в которой по ошибке используется десятичный разделитель «.».

Помните, что системная переменная `MoneyFormat` определяет доллар ($) в качестве валюты по умолчанию.

Скрипт загрузки

```
SET MoneyThousandSep='.';
SET MoneyDecimalSep=',';
SET MoneyFormat='$###0.00;-$###0.00';

Transactions:
Load
date,
id,
amount
Inline
[
date,id,amount
01/01/2022,1,'$14,41'
01/02/2022,2,'$2.814,32'
01/03/2022,3,'$249,36'
01/04/2022,4,'$24,37'
01/05/2022,5,'$7,54'
01/06/2022,6,'$243,63'
01/07/2022,7,'$545,36'
```

```
01/08/2022,8,'$3,55'
01/09/2022,9,'$3,436'
01/10/2022,10,'$345.66'
];
```

Результаты

Текст абзаца для результатов.

Загрузите данные и откройте лист. Создайте новую таблицу и добавьте это поле как измерение:`amount`.

Добавьте следующие меры:

- `isNum(amount)`
- `sum(amount)`

Ниже приводятся результаты, демонстрирующие правильную интерпретацию всех значений, кроме суммы, в которой использовалась точка в качестве десятичного разделителя. В этом случае должна была использоваться запятая.

Результирующая таблица

amount	=isNum(amount)	=Sum(amount)
Итоги	0	$3905.98
$345.66	0	$0.00
$3,436	-1	$3.44
$3,55	-1	$3.55
$7,54	-1	$7.54
$14,41	-1	$14.41
$24,37	-1	$24.37
$243,63	-1	$243.63
$249,36	-1	$249.36
$545,36	-1	$545.36
$2.814,32	-1	$2814.32

MoneyFormat

Эта системная переменная определяет образец формата, используемый Qlik для автоматического преобразования текста в число, где перед числом стоит символ денежной единицы. Она также определяет, как меры, для которых свойству «Формат числа» задано значение «Денежный», будут отображаться в объектах диаграммы.

Символ, определенный как часть образца формата в системной переменной `MoneyFormat`, заменяет обозначение валюты, заданное в региональных настройках.

По умолчанию Qlik Sense по-разному отображает числа и текст в диаграммах таблицы. Числа выравниваются по правому краю, а текст по левому. Это позволяет легко выявлять проблемы преобразования текста в числа. Все таблицы на этой странице, в которых отображаются результаты Qlik Sense, будут использовать это форматирование.

Синтаксис:

MoneyFormat

```
Set MoneyFormat='$ #,##0.00; ($ #,##0.00)';
```

Это форматирование будет использоваться в объектах диаграммы, когда свойству `Number Formatting` числового поля задано значение `Money`. Затем при интерпретации числовых текстовых полей в Qlik Sense, если символ валюты текстового поля соответствует символу, определенному в переменной `MoneyFormat`, Qlik Sense будет интерпретировать это поле как денежное значение.

Эта функция часто используется вместе со следующими функциями:

Связанные функции

Функция	Взаимодействие
MoneyDecimalSep (page 236)	В качестве формата чисел будет использоваться `MoneyDecimalSep` для форматирования полей объектов.
MoneyThousandSep (page 245)	В качестве формата чисел будет использоваться `MoneyThousandSep` для форматирования полей объектов.

Региональные настройки

Если не указано иное, в примерах, приведенных в данном разделе, используется следующий формат даты: ММ/ДД/ГГГГ. Формат даты указан в операторе `SET DateFormat` скрипта загрузки данных. В вашей системе может быть установлен другой формат даты по умолчанию в зависимости от региональных настроек и других факторов. Можно изменить формат в примерах в соответствии с потребностями. Или можно изменить форматы в скрипте загрузки в соответствии с этими примерами.

Региональные настройки по умолчанию в приложениях основаны на системных региональных настройках компьютера или сервера, где установлено ПО Qlik Sense. Если на сервере Qlik Sense, к которому обращается пользователь, выбран шведский язык, то редактор загрузки данных будет использовать шведские региональные настройки для даты, времени и валюты. Эти параметры регионального формата не связаны с языком, отображаемым в интерфейсе пользователя Qlik Sense. Qlik Sense будет отображаться на языке, который используется в браузере.

Пример 1. MoneyFormat

Скрипт загрузки и результаты

Обзор

Откройте редактор загрузки данных и добавьте приведенный ниже скрипт загрузки на новую вкладку.

Скрипт загрузки содержит набор данных, который загружается в таблицу с именем `Transactions`. По умолчанию используется определение переменной `MoneyFormat`.

Скрипт загрузки

```
SET MoneyThousandSep=',';
SET MoneyDecimalSep='.';
SET MoneyFormat='$###0.00;-$###0.00';

Transactions:
Load
date,
id,
amount
Inline
[
date,id,amount
01/01/2022,1,$10000000441
01/02/2022,2,$21237492432
01/03/2022,3,$249475336
01/04/2022,4,$24313369837
01/05/2022,5,$7873578754
01/06/2022,6,$24313884663
01/07/2022,7,$545883436
01/08/2022,8,$35545828255
01/09/2022,9,$37565817436
01/10/2022,10,$3454343566
];
```

Результаты

Загрузите данные и откройте лист. Создайте новую таблицу и добавьте эти поля как измерения:

- `date`
- `amount`

Добавьте эту меру:

```
=Sum(amount)
```

В области **Формат чисел** выберите **Денежный**, чтобы настроить `Sum(amount)` в качестве денежной единицы.

Результирующая таблица

date	Amount	=Sum(amount)
Итоги		$165099674156.00
01/01/2022	$10000000441	$10000000441.00
01/02/2022	$21237492432	$21237492432.00
01/03/2022	$249475336	$249475336.00
01/04/2022	$24313369837	$24313369837.00
01/05/2022	$7873578754	$7873578754.00
01/06/2022	$24313884663	$24313884663.00
01/07/2022	$545883436	$545883436.00
01/08/2022	$35545828255	$35545828255.00
01/09/2022	$37565817436	$37565817436.00
01/10/2022	$3454343566	$3454343566.00

По умолчанию используется определение `MoneyFormat`. Это выглядит следующим образом: `$###0.00;-$###0.00`. В таблице результатов формат поля `amount` отображает символ валюты и десятичную точку, также задано количество десятичных знаков.

Пример 2. MoneyFormat с разделителем тысяч и смешанными форматами ввода

Скрипт загрузки и результаты

Обзор

Откройте редактор загрузки данных и добавьте приведенный ниже скрипт загрузки на новую вкладку.

Скрипт загрузки содержит следующее:

- Набор данных со смешанными форматами ввода, загруженный в таблицу под именем `Transactions`, где перемешаны разделители тысяч и десятичные разделители.
- Модифицированное определение `MoneyFormat` включает запятую в качестве разделителя тысяч.
- В одной из строк данных разделители тысяч (,) по ошибке стоят в неправильном месте. Обратите внимание, что эта сумма осталась в виде текста и не преобразована в число.

Скрипт загрузки

```
SET MoneyThousandSep=',';
SET MoneyDecimalSep='.';
SET MoneyFormat = '$#,##0.00;-$#,##0.00';
```

```
Transactions:
Load
date,
id,
amount
Inline
[
date,id,amount
01/01/2022,1,'$10,000,000,441.45'
01/02/2022,2,'$212,3749,24,32.23'
01/03/2022,3,$249475336.45
01/04/2022,4,$24,313,369,837
01/05/2022,5,$7873578754
01/06/2022,6,$24313884663
01/07/2022,7,$545883436
01/08/2022,8,$35545828255
01/09/2022,9,$37565817436
01/10/2022,10,$3454343566
];
```

Результаты

Загрузите данные и откройте лист. Создайте новую таблицу и добавьте эти поля как измерения:

- `date`
- `amount`

Добавьте эту меру:

```
=Sum(amount)
```

В области **Формат чисел** выберите **Денежный**, чтобы настроить `Sum(amount)` в качестве денежной единицы.

Результирующая таблица

date	Amount	=Sum(amount)
Итоги		$119,548,811,911.90
01/01/2022	$10,000,000,441.45	$10,000,000,441.45
01/02/2022	$212,3749,24,32.23	$0.00
01/03/2022	$249475336.45	$249,475,336.45
01/04/2022	$24	$24.00
01/05/2022	$7873578754	$7,873,578,754.00
01/06/2022	$24313884663	$24,313,884,663.00
01/07/2022	$545883436	$545,883,436.00
01/08/2022	$35545828255	$35,545,828,255.00

date	Amount	=Sum(amount)
01/09/2022	$37565817436	$37,565,817,436.00
01/10/2022	$3454343566	$3,454,343,566.00

В начале скрипта системная переменная `MoneyFormat` модифицируется для использования запятой в качестве разделителя тысяч. В таблице Qlik Sense видно, что форматирование включает этот разделитель. Более того, строку с ошибочным разделителем не удалось правильно интерпретировать, и она осталась в виде текста. Именно поэтому эта строка не включена в итоговую сумму.

MoneyThousandSep

Указанный разделитель тысяч заменяет группирующий символ знаков для денежных сумм, заданный в региональных настройках.

По умолчанию Qlik Sense по-разному отображает числа и текст в диаграммах таблицы. Числа выравниваются по правому краю, а текст по левому. Это позволяет легко выявлять проблемы преобразования текста в числа. Все таблицы на этой странице, в которых отображаются результаты Qlik Sense, будут использовать это форматирование.

Синтаксис:

`MoneyThousandSep`

Приложения Qlik Sense будут интерпретировать текстовые поля, соответствующие этому формату, как денежные значения. Текстовое поле должно содержать символ валюты, определенный в системной переменной `MoneyFormat`. `MoneyThousandSep` особенно полезно при работе с источниками данных, полученными из систем с разными региональными настройками.

Следующий пример демонстрирует возможное использование системной переменной `MoneyThousandSep`:

```
Set MoneyDecimalSep=',';
```

Эта функция часто используется вместе со следующими функциями:

Связанные функции

Функция	Взаимодействие
`MoneyFormat`	В случаях интерпретации текстового поля символ `MoneyFormat` будет использоваться как часть такой интерпретации. В качестве формата чисел будет использоваться `MoneyFormat` для Qlik Sense в объектах диаграммы.
`MoneyDecimalSep`	В случаях интерпретации текстового поля также необходимо использовать функцию `MoneyDecimalSep`.

Региональные настройки

Если не указано иное, в примерах, приведенных в данном разделе, используется следующий формат даты: ММ/ДД/ГГГГ. Формат даты указан в операторе `SET DateFormat` скрипта загрузки данных. В вашей системе может быть установлен другой формат даты по умолчанию в зависимости от региональных настроек и других факторов. Можно изменить формат в примерах в соответствии с потребностями. Или можно изменить форматы в скрипте загрузки в соответствии с этими примерами.

Региональные настройки по умолчанию в приложениях основаны на системных региональных настройках компьютера или сервера, где установлено ПО Qlik Sense. Если на сервере Qlik Sense, к которому обращается пользователь, выбран шведский язык, то редактор загрузки данных будет использовать шведские региональные настройки для даты, времени и валюты. Эти параметры регионального формата не связаны с языком, отображаемым в интерфейсе пользователя Qlik Sense. Qlik Sense будет отображаться на языке, который используется в браузере.

Пример 1. Использование запятой (,) в MoneyThousandSep

Скрипт загрузки и результаты

Обзор

Откройте редактор загрузки данных и добавьте приведенный ниже скрипт загрузки на новую вкладку.

Скрипт загрузки содержит следующее:

- Набор данных, загруженный в таблицу под именем `Transactions`.
- Предоставленные данные, содержащие денежное поле в текстовом формате с использованием запятой (,) в качестве разделителя тысяч. Кроме того, каждая запись имеет префикс «$».

Помните, что системная переменная `MoneyFormat` определяет доллар ($) в качестве валюты по умолчанию.

Скрипт загрузки

```
SET MoneyThousandSep=',';
SET MoneyDecimalSep='.';
SET MoneyFormat='$###0.00;-$###0.00';

Transactions:
Load
date,
id,
amount
Inline
[
date,id,amount
01/01/2022,1,'$10,000,000,441'
01/02/2022,2,'$21,237,492,432'
```

```
01/03/2022,3,'$249,475,336'
01/04/2022,4,'$24,313,369,837'
01/05/2022,5,'$7,873,578,754'
01/06/2022,6,'$24,313,884,663'
01/07/2022,7,'$545,883,436'
01/08/2022,8,'$35,545,828,255'
01/09/2022,9,'$37,565,817,436'
01/10/2022,10,'$3.454.343.566'
];
```

Результаты

Загрузите данные и откройте лист. Создайте новую таблицу и добавьте это поле как измерение:`amount`.

Добавьте следующие меры:

- `isNum(amount)`
- `sum(amount)`

Результаты приводятся ниже. Таблица демонстрирует правильную интерпретацию всех значений с использованием запятой (,) в качестве разделителей тысяч.

Поле `amount` правильно интерпретировано для всех значений, за исключением одного, где в качестве разделителя тысяч используется точка (.).

Результирующая таблица

amount	=isNum(amount)	=Sum(amount)
Итоги	0	$161645330590.00
$3.454.343.566	0	$0.00
$249,475,336	-1	$249475336.00
$545,883,436	-1	$545883436.00
$7,873,578,754	-1	$7873578754.00
$10,000,000,441	-1	$10000000441.00
$21,237,492,432	-1	$21237492432.00
$24,313,369,837	-1	$24313369837.00
$24,33,884,663	-1	$24313884663.00
$35,545,828,255	-1	$35545828255.00
$37,565,817,436	-1	$37565817436.00

Пример 2. Использование точки (.) в MoneyThousandSep

Скрипт загрузки и результаты

Обзор

Откройте редактор загрузки данных и добавьте приведенный ниже скрипт загрузки на новую вкладку.

Скрипт загрузки содержит следующее:

- Набор данных, загруженный в таблицу под именем `Transactions`.
- Предоставленные данные, содержащие денежное поле в текстовом формате с использованием точки (.) в качестве разделителя тысяч. Кроме того, каждая запись имеет префикс «$».

Помните, что системная переменная `MoneyFormat` определяет доллар ($) в качестве валюты по умолчанию.

Скрипт загрузки

```
SET MoneyThousandSep='.';
SET MoneyDecimalSep=',';
SET MoneyFormat='$###0.00;-$###0.00';

Transactions:
Load
date,
id,
amount
Inline
[
date,id,amount
01/01/2022,1,'$10.000.000.441'
01/02/2022,2,'$21.237.492.432'
01/03/2022,3,'$249.475.336'
01/04/2022,4,'$24.313.369.837'
01/05/2022,5,'$7.873.578.754'
01/06/2022,6,'$24.313.884.663'
01/07/2022,7,'$545.883.436'
01/08/2022,8,'$35.545.828.255'
01/09/2022,9,'$37.565.817.436'
01/10/2022,10,'$3,454,343,566'
];
```

Результаты

Загрузите данные и откройте лист. Создайте новую таблицу и добавьте это поле как измерение:`amount`.

Добавьте следующие меры:

- `isNum(amount)`
- `sum(amount)`

Ниже приводится таблица результатов, которая демонстрирует правильную интерпретацию всех значений с использованием точки (.) в качестве разделителей тысяч.

Поле `amount` правильно интерпретировано для всех значений, за исключением одного, где в качестве разделителя тысяч используется запятая (,).

Результирующая таблица

amount	=isNum(amount)	=Sum(amount)
Итоги	0	$161645330590.00
$3,545,343,566	0	$0.00
$249.475.336	-1	$249475336.00
$545.883.436	-1	545883436.00
$7.873.578.754	-1	$7873578754.00
$10.000.000.441	-1	$10000000441.00
$21.237.492.432	-1	$21237492432.00
$24.313.884.663	-1	$24313884663.00
$24.313.884.663	-1	$24313884663.00
$35.545.828.255	-1	$35545828255.00
$37.565.817.436	-1	$37565817436.00

MonthNames

Указанный формат заменяет имена месяцев, заданные в региональных настройках.

Синтаксис:

```
MonthNames
```

При модификации переменной необходимо использовать ; для разделения отдельных значений.

Примеры функции

Пример	Результаты
`Set MonthNames='Jan;Feb;Mar;Apr;May;Jun;Jul;Aug;Sep;Oct;Nov;Dec';`	Это использование функции `MonthNames` определяет названия месяцев на английском языке в

Пример	Результаты
	сокращенной форме.
`Set MonthNames='Enero;Feb;Marzo;Abr;Mayo;Jun;Jul;Agosto;Set;Oct;Nov;Dic';`	Это использование функции `MonthNames` определяет названия месяцев на испанском языке в сокращенной форме.

Функция `MonthNames` часто используется в сочетании со следующими функциями:

Связанные функции

Функция	Взаимодействие
month (page 924)	Функция скрипта для возврата значений, определенных в `MonthNames` в качестве значений поля
Date (page 1248)	Функция скрипта для возврата значений, определенных в `MonthNames` в качестве значений поля на основе предоставленного аргумента форматирования
LongMonthNames (page 232)	Значения длинной формы `MonthNames`.

Региональные настройки

Если не указано иное, в примерах, приведенных в данном разделе, используется следующий формат даты: ММ/ДД/ГГГГ. Формат даты указан в операторе `SET DateFormat` скрипта загрузки данных. В вашей системе может быть установлен другой формат даты по умолчанию в зависимости от региональных настроек и других факторов. Можно изменить формат в примерах в соответствии с потребностями. Или можно изменить форматы в скрипте загрузки в соответствии с этими примерами.

Региональные настройки по умолчанию в приложениях основаны на системных региональных настройках компьютера или сервера, где установлено ПО Qlik Sense. Если на сервере Qlik Sense, к которому обращается пользователь, выбран шведский язык, то редактор загрузки данных будет использовать шведские региональные настройки для даты, времени и валюты. Эти параметры регионального формата не связаны с языком, отображаемым в интерфейсе пользователя Qlik Sense. Qlik Sense будет отображаться на языке, который используется в браузере.

Пример 1. Системные переменные по умолчанию

Скрипт загрузки и результаты

Обзор

Откройте редактор загрузки данных и добавьте приведенный ниже скрипт загрузки на новую вкладку.

Скрипт загрузки содержит следующее:

- Набор данных, загруженный в таблицу под именем `Transactions`.
- Поле `date`.
- Определение `MonthNames` по умолчанию.

Скрипт загрузки

```
SET MonthNames='Jan;Feb;Mar;Apr;May;Jun;Jul;Aug;Sep;Oct;Nov;Dec';

Transactions:
LOAD
date,
Month(date) as monthname,
id,
amount
INLINE
[
date,id,amount
01/01/2022,1,1000.45
01/02/2022,2,2123.34
01/03/2022,3,4124.35
01/04/2022,4,2431.36
01/05/2022,5,4787.78
01/06/2022,6,2431.84
01/07/2022,7,2854.83
01/08/2022,8,3554.28
01/09/2022,9,3756.17
01/10/2022,10,3454.35
];
```

Результаты

Загрузите данные и откройте лист. Создайте новую таблицу и добавьте эти поля как измерения:

- `date`
- `monthname`

Создайте эту меру:

```
=sum(amount)
```

Результирующая таблица

date	monthname	sum(amount)
01/01/2022	Jan	1000.45

date	monthname	sum(amount)
01/02/2022	Jan	2123.34
01/03/2022	Jan	4124.35
01/04/2022	Jan	2431.36
01/05/2022	Jan	4787.78
01/06/2022	Jan	2431.84
01/07/2022	Jan	2854.83
01/08/2022	Jan	3554.28
01/09/2022	Jan	3756.17
01/10/2022	Jan	3454.35

По умолчанию используется определение `MonthNames`. В скрипте загрузки функция `Month` используется вместе с полем `date`, предоставленном в качестве аргумента.

В таблице результатов выход этой функции `Month` показывает месяцы года в формате определения `MonthNames`.

Пример 2. Изменение системной переменной

Скрипт загрузки и результаты

Обзор

Откройте редактор загрузки данных и добавьте приведенный ниже скрипт загрузки на новую вкладку.

Скрипт загрузки содержит следующее:

- Набор данных, загруженный в таблицу под именем `Transactions`.
- Поле `date`.
- Переменная `MonthNames` , модифицированная для использования сокращенных названий месяцев на испанском языке.

Скрипт загрузки

```
Set MonthNames='Enero;Feb;Marzo;Abr;Mayo;Jun;Jul;Agosto;Set;Oct;Nov;Dic';

Transactions:
LOAD
date,
month(date) as month,
id,
amount
INLINE
[
date,id,amount
```

```
01/01/2022,1,1000
02/01/2022,2,2123
03/01/2022,3,4124
04/01/2022,4,2431
];
```

Результаты

Загрузите данные и откройте лист. Создайте новую таблицу и добавьте эти поля как измерения:

- `date`
- `monthname`

Создайте эту меру:

```
=sum(amount)
```

Результирующая таблица

date	monthname	sum(amount)
01/01/2022	Enero	1000.45
01/02/2022	Enero	2123.34
01/03/2022	Enero	4124.35
01/04/2022	Enero	2431.36
01/05/2022	Enero	4787.78
01/06/2022	Enero	2431.84
01/07/2022	Enero	2854.83
01/08/2022	Enero	3554.28
01/09/2022	Enero	3756.17
01/10/2022	Enero	3454.35

В скрипте загрузки первая переменная `MonthNames` модифицируется для перечисления месяцев года с использованием сокращенных названий на испанском языке. Функция `Month` используется вместе с полем `date`, предоставленном в качестве аргумента.

В таблице результатов выход этой функции `Month` показывает месяцы года в формате определения `MonthNames`.

Важно помнить, что если язык для переменной `MonthNames` изменяется так, как в этом примере, переменная `LongMonthNames` все равно будет содержать месяцы года на английском языке. Переменную `LongMonthNames` потребуется изменить, если в приложении используются обе эти переменные.

Пример 3. Функция даты

Скрипт загрузки и результаты

Обзор

Откройте редактор загрузки данных и добавьте приведенный ниже скрипт загрузки на новую вкладку.

Скрипт загрузки содержит следующее:

- Набор данных, загруженный в таблицу под именем `Transactions`.
- Поле `date`.
- Определение `MonthNames` по умолчанию.

Скрипт загрузки

```
SET MonthNames='Jan;Feb;Mar;Apr;May;Jun;Jul;Aug;Sep;Oct;Nov;Dec';

Transactions:
LOAD
date,
Month(date, 'MMM') as monthname,
id,
amount
INLINE
[
date,id,amount
01/01/2022,1,1000.45
01/02/2022,2,2123.34
01/03/2022,3,4124.35
01/04/2022,4,2431.36
01/05/2022,5,4787.78
01/06/2022,6,2431.84
01/07/2022,7,2854.83
01/08/2022,8,3554.28
01/09/2022,9,3756.17
01/10/2022,10,3454.35
];
```

Результаты

Загрузите данные и откройте лист. Создайте новую таблицу и добавьте эти поля как измерения:

- `date`
- `monthname`

Создайте эту меру:

```
=sum(amount)
```

Результирующая таблица

date	monthname	sum(amount)
01/01/2022	Jan	1000.45

date	monthname	sum(amount)
01/02/2022	Jan	2123.34
01/03/2022	Jan	4124.35
01/04/2022	Jan	2431.36
01/05/2022	Jan	4787.78
01/06/2022	Jan	2431.84
01/07/2022	Jan	2854.83
01/08/2022	Jan	3554.28
01/09/2022	Jan	3756.17
01/10/2022	Jan	3454.35

По умолчанию используется определение `MonthNames`. В скрипте загрузки функция `Date` используется вместе с полем `date`, предоставленном в качестве первого аргумента. Второй аргумент — `MMM`.

При использовании этого форматирования Qlik Sense преобразует значения из первого аргумента в соответствующее название месяца, заданного в переменной `MonthNames`. В таблице результатов это демонстрируют значения созданного нами поля `month`.

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

Параметр определяет, какой день в январе необходимо задать в качестве исходного дня для определения недели 1. Другими словами, этот параметр определяет, сколько дней в неделе 1 должны быть датами в январе.

Синтаксис:

```
ReferenceDay
```

ReferenceDay устанавливает, сколько дней включается в первую неделю года. Для ReferenceDay можно задать любое значение между 1 и 7. Любое значение за пределами диапазона 1-7 интерпретируется как середина недели (4), что эквивалентно установке ReferenceDay = 4.

Если не задано значение для параметра ReferenceDay, то значение по умолчанию будет отображать ReferenceDay=0, что будет интерпретироваться как середина недели (4), как показано в таблице значений ReferenceDay ниже.

Функция ReferenceDay часто используется в сочетании со следующими функциями:

Связанные функции

Переменная	Взаимодействие
BrokenWeeks (page 211)	Если приложение Qlik Sense работает с полными неделями, будет принудительно применен параметр переменной ReferenceDay. Однако если используются неполные недели, неделя 1 начнется 1 января и завершится в сочетании с настройкой переменной FirstWeekDay, флаг ReferenceDay игнорируется.
FirstWeekDay (page 225)	Целое число, которое определяет, какой день использовать в качестве первого дня недели.

Qlik Sense позволяет задать следующие значения для ReferenceDay:

Значения ReferenceDay

Значение	Reference day
0 (по умолчанию)	January 4
1	January 1
2	Январь 2 г.
3	January 3
4	January 4
5	January 5
6	January 6
7	January 7

В следующем примере ReferenceDay = 3 определяет 3 января как исходный день (ReferenceDay):

```
SET ReferenceDay=3; //(set January 3 as the reference day)
```

Региональные настройки

Если не указано иное, в примерах, приведенных в данном разделе, используется следующий формат даты: ММ/ДД/ГГГГ. Формат даты указан в операторе `SET DateFormat` скрипта загрузки данных. В вашей системе может быть установлен другой формат даты по умолчанию в зависимости от региональных настроек и других факторов. Можно изменить формат в примерах в соответствии с потребностями. Или можно изменить форматы в скрипте загрузки в соответствии с этими примерами.

Региональные настройки по умолчанию в приложениях основаны на системных региональных настройках компьютера или сервера, где установлено ПО Qlik Sense. Если на сервере Qlik Sense, к которому обращается пользователь, выбран шведский язык, то редактор загрузки данных будет использовать шведские региональные настройки для даты, времени и валюты. Эти параметры регионального формата не связаны с языком, отображаемым в интерфейсе пользователя Qlik Sense. Qlik Sense будет отображаться на языке, который используется в браузере.

Пример 1. Скрипт загрузки с использованием значения по умолчанию, ReferenceDay=0

Скрипт загрузки и результаты

Обзор

Откройте редактор загрузки данных и добавьте приведенный ниже скрипт загрузки на новую вкладку.

Скрипт загрузки содержит следующее:

- Переменная `ReferenceDay`, для которой задано значение `0`.
- Переменная `BrokenWeeks` со значением `0`, которая заставляет приложение использовать полные недели.
- Набор данных, содержащий даты с конца 2019 по начало 2020 года.

Скрипт загрузки

```
SET BrokenWeeks = 0;
SET ReferenceDay = 0;

Sales:
LOAD
date,
sales,
week(date) as week,
weekday(date) as weekday
Inline [
date,sales
12/27/2019,5000
12/28/2019,6000
```

```
12/29/2019,7000
12/30/2019,4000
12/31/2019,3000
01/01/2020,6000
01/02/2020,3000
01/03/2020,6000
01/04/2020,8000
01/05/2020,5000
01/06/2020,7000
01/07/2020,3000
01/08/2020,5000
01/09/2020,9000
01/10/2020,5000
01/11/2020,7000
];
```

Результаты

Загрузите данные и откройте лист. Создайте новую таблицу и добавьте эти поля как измерения:

- `date`
- `week`
- `weekday`

Результирующая таблица

date	week	weekday
12/27/2019	52	Fri
12/28/2019	52	Sat
12/29/2019	1	Sun
12/30/2019	1	Mon
12/31/2019	1	Tue
01/01/2020	1	Wed
01/02/2020	1	Thu
01/03/2020	1	Fri
01/04/2020	1	Sat
01/05/2020	2	Sun
01/06/2020	2	Mon
01/07/2020	2	Tue
01/08/2020	2	Wed
01/09/2020	2	Thu
01/10/2020	2	Fri
01/11/2020	2	Sat

Неделя 52 заканчивается в субботу, 28 декабря. Поскольку `ReferenceDay` требует, чтобы 4 января было включено в неделю 1, неделя 1 начинается 29 декабря и заканчивается в субботу, 4 января.

Пример. Переменная ReferenceDay со значением 5

Скрипт загрузки и результаты

Обзор

Откройте редактор загрузки данных и добавьте приведенный ниже скрипт загрузки на новую вкладку.

Скрипт загрузки содержит следующее:

- Переменная `ReferenceDay`, для которой задано значение `5`.
- Переменная `BrokenWeeks` со значением `0`, которая заставляет приложение использовать полные недели.
- Набор данных, содержащий даты с конца 2019 по начало 2020 года.

Скрипт загрузки

```
SET BrokenWeeks = 0;
SET ReferenceDay = 5;

Sales:
LOAD
date,
sales,
week(date) as week,
weekday(date) as weekday
Inline [
date,sales
12/27/2019,5000
12/28/2019,6000
12/29/2019,7000
12/30/2019,4000
12/31/2019,3000
01/01/2020,6000
01/02/2020,3000
01/03/2020,6000
01/04/2020,8000
01/05/2020,5000
01/06/2020,7000
01/07/2020,3000
01/08/2020,5000
01/09/2020,9000
01/10/2020,5000
01/11/2020,7000
];
```

Результаты

Загрузите данные и откройте лист. Создайте новую таблицу и добавьте эти поля как измерения:

- `date`
- `week`
- `weekday`

Результирующая таблица

date	week	weekday
12/27/2019	52	Fri
12/28/2019	52	Sat
12/29/2019	53	Sun
12/30/2019	53	Mon
12/31/2019	53	Tue
01/01/2020	53	Wed
01/02/2020	53	Thu
01/03/2020	53	Fri
01/04/2020	53	Sat
01/05/2020	1	Sun
01/06/2020	1	Mon
01/07/2020	1	Tue
01/08/2020	1	Wed
01/09/2020	1	Thu
01/10/2020	1	Fri
01/11/2020	1	Sat

Неделя 52 заканчивается в субботу, 28 декабря. Переменная `BrokenWeeks`, которая заставляет приложение использовать полные недели. Значение исходного дня `5` требует, чтобы 5 января было включено в неделю 1.

Однако эта дата наступает через восемь дней после завершения недели 52 предыдущего года. Таким образом, неделя 53 начинается 29 декабря и заканчивается 4 января. Неделя 1 начинается в воскресенье, 5 января.

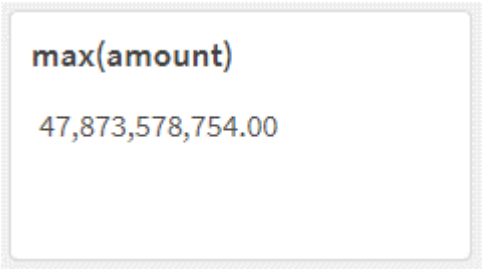
ThousandSep

Указанный разделитель тысяч заменяет группирующий символ знаков, используемый в операционной системе (региональные настройки).

Синтаксис:

```
ThousandSep
```

Объект Qlik Sense с использованием переменной ThousandSep (с разделителем тысяч)



Приложения Qlik Sense интерпретируют текстовые поля, соответствующие этому формату, как числа. Это форматирование будет использоваться в объектах диаграммы, когда свойству **Формат числа** числового поля задано значение **Число**.

`ThousandSep` помогает при работе с источниками данных, полученными из систем с разными региональными настройками.

> *Если в переменную `ThousandSep` вносятся изменения уже после создания и форматирования объектов в приложении, пользователю потребуется переформатировать каждое релевантное поле, отменив выбор значения **Формат чисел** и снова выбрав **Числовой**.*

Следующие примеры демонстрируют возможное использование системной переменной `ThousandSep`:

```
Set ThousandSep=','; //(for example, seven billion will be displayed as: 7,000,000,000)
```

```
Set ThousandSep=' '; //(for example, seven billion will be displayed as: 7 000 000 000)
```

Эти темы помогут вам в работе с этой функцией:

Связанные темы

Тема	Описание
DecimalSep (page 222)	В случаях интерпретации текстовых полей также должны учитываться настройки десятичного разделителя, предоставленные этой функцией. Где это необходимо Qlik Sense будет использовать формат чисел **DecimalSep**.

Региональные настройки

Если не указано иное, в примерах, приведенных в данном разделе, используется следующий формат даты: ММ/ДД/ГГГГ. Формат даты указан в операторе `SET DateFormat` скрипта загрузки данных. В вашей системе может быть установлен другой формат даты по умолчанию в зависимости от региональных настроек и других факторов. Можно изменить формат в примерах в соответствии с потребностями. Или можно изменить форматы в скрипте загрузки в соответствии с этими примерами.

Региональные настройки по умолчанию в приложениях основаны на системных региональных настройках компьютера или сервера, где установлено ПО Qlik Sense. Если на сервере Qlik Sense, к которому обращается пользователь, выбран шведский язык, то редактор загрузки данных будет использовать шведские региональные настройки для даты, времени и валюты. Эти параметры регионального формата не связаны с языком, отображаемым в интерфейсе пользователя Qlik Sense. Qlik Sense будет отображаться на языке, который используется в браузере.

Пример 1. Системные переменные по умолчанию

Скрипт загрузки и результаты

Обзор

Откройте редактор загрузки данных и добавьте приведенный ниже скрипт загрузки на новую вкладку.

Скрипт загрузки содержит следующее:

- Набор данных, загруженный в таблицу под именем `Transactions`.
- Использование определения переменной `ThousandSep` по умолчанию.

Скрипт загрузки

```
Transactions:
Load
date,
id,
amount
Inline
[
date,id,amount
01/01/2022,1,10000000441
01/02/2022,2,21237492432
01/03/2022,3,41249475336
01/04/2022,4,24313369837
01/05/2022,5,47873578754
01/06/2022,6,24313884663
01/07/2022,7,28545883436
01/08/2022,8,35545828255
01/09/2022,9,37565817436
01/10/2022,10,3454343566
];
```

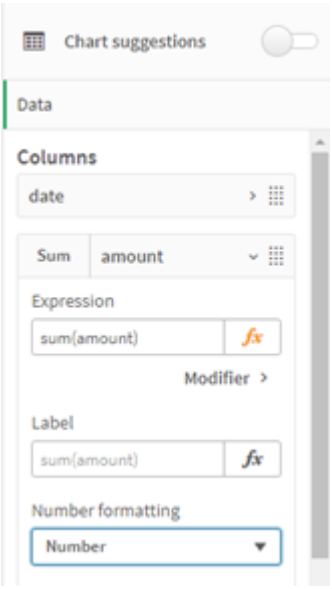
Результаты

Выполните следующие действия.

1. Загрузите данные и откройте лист. Создайте новую таблицу и добавьте это поле как измерение:`date`.

2. Добавьте следующую меру:
   `=sum(amount)`
3. На панели свойств выберите меру в области **Данные**.
4. В поле **Формат чисел** выберите **Числовой**.

Коррекция формата чисел для меры диаграммы



Результирующая таблица

date	=sum(amount)
01/01/2022	10,000,000,441.00
01/02/2022	21,237,492,432.00
01/03/2022	41,249,475,336.00
01/04/2022	24,313,369,837.00
01/05/2022	47,873,578,754.00
01/06/2022	24,313,884,663.00
01/07/2022	28,545,883,436.00
01/08/2022	35,545,828,255.00
01/09/2022	37,565,817,436.00
01/10/2022	3,454,343,566.00

В этом примере используется определение `ThousandSep` по умолчанию, которое задает запятую (,) в качестве разделителя. В таблице результатов формат поля суммы использует отображает запятую в качестве разделителя тысяч.

Пример 2. Изменение системной переменной

Скрипт загрузки и результаты

Обзор

Откройте редактор загрузки данных и добавьте приведенный ниже скрипт загрузки на новую вкладку.

Скрипт загрузки содержит следующее:

- Тот же набор данных, что в первом примере, загруженный в таблицу под именем `Transactions`.
- Модификация определения `ThousandSep` в начале скрипта для отображения символа «*» в качестве разделителя тысяч. Этот радикальный пример используется исключительно для демонстрации функциональных возможностей переменной.

Модификация, используемая в этом примере, является чрезмерной, но она используется здесь для демонстрации функциональных возможностей переменной.

Скрипт загрузки

```
SET ThousandSep='*';

Transactions:
Load
date,
id,
amount
Inline
[
date,id,amount
01/01/2022,1,10000000441
01/02/2022,2,21237492432
01/03/2022,3,41249475336
01/04/2022,4,24313369837
01/05/2022,5,47873578754
01/06/2022,6,24313884663
01/07/2022,7,28545883436
01/08/2022,8,35545828255
01/09/2022,9,37565817436
01/10/2022,10,3454343566
];
```

Результаты

Выполните следующие действия.

1. Загрузите данные и откройте лист. Создайте новую таблицу и добавьте это поле как измерение:`date`.

2. Добавьте следующую меру:
   `=sum(amount)`
3. На панели свойств выберите меру в области **Данные**.
4. В поле **Формат чисел** выберите **Пользовательский**.

Результирующая таблица

date	=sum(amount)
01/01/2022	10*000*000*441.00
01/02/2022	21*237*492*432.00
01/03/2022	41*249*475*336.00
01/04/2022	24*313*369*837.00
01/05/2022	47*873*578*754.00
01/06/2022	24*313*884*663.00
01/07/2022	28*545*883*436.00
01/08/2022	35*545*828*255.00
01/09/2022	37*565*817*436.00
01/10/2022	3*454*343*566.00

В начале скрипта значение системной переменной `ThousandSep` меняется на «*». В таблице результатов формат поля суммы отображает «*» в качестве разделителя тысяч.

Пример 3. Интерпретация текста

Скрипт загрузки и результаты

Обзор

Откройте редактор загрузки данных и добавьте приведенный ниже скрипт загрузки на новую вкладку.

Скрипт загрузки содержит следующее:

- Набор данных, загруженный в таблицу под именем `Transactions`.
- Данные, содержащие числовое поле в текстовом формате с использованием запятой (,) в качестве разделителя тысяч.
- Использование системной переменной `ThousandSep` по умолчанию.

Скрипт загрузки

```
Transactions:
Load
date,
id,
```

```
amount
Inline
[
date,id,amount
01/01/2022,1,'10,000,000,441'
01/02/2022,2,'21,492,432'
01/03/2022,3,'4,249,475,336'
01/04/2022,4,'24,313,369,837'
01/05/2022,5,'4,873,578,754'
01/06/2022,6,'313,884,663'
01/07/2022,7,'2,545,883,436'
01/08/2022,8,'545,828,255'
01/09/2022,9,'37,565,817,436'
01/10/2022,10,'3,454,343,566'
];
```

Результаты

Выполните следующие действия.

1. Загрузите данные и откройте лист. Создайте новую таблицу и добавьте это поле как измерение:`date`.
2. Добавьте следующую меру:
   `=sum(amount)`
3. На панели свойств выберите меру в области **Данные**.
4. В поле **Формат чисел** выберите **Число**.
5. Добавьте следующую меру, чтобы проверить, содержит ли поле суммы числовое значение:
   `=isnum(amount)`

Результирующая таблица

date	=sum(amount)	=isnum(amount)
01/01/2022	10,000,000,441.00	-1
01/02/2022	21,492,432.00	-1
01/03/2022	4,249,475,336.00	-1
01/04/2022	24,313,369,837.00	-1
01/05/2022	4,873,578,754.00	-1
01/06/2022	313,884,663.00	-1
01/07/2022	2,545,883,436.00	-1
01/08/2022	545,828,255.00	-1
01/09/2022	37,565,817,436.00	-1
01/10/2022	3*454*343*566.00	-1

После загрузки данных видно, что приложение Qlik Sense интерпретировало сумму как числовое поле, так как данные соответствуют переменной ThousandSep. Это демонстрирует функция isnum(), которая проверяет каждую запись, возвращая -1 или TRUE.

В Qlik Sense логическое значение «истина» представлено как -1, а «ложь» – как 0.

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 могут использоваться во внутренних и внешних функциях агрегирования и заключаются в фигурные скобки.

Пример: Внутреннее выражение множества

```
Sum( {$<Year={2021}>} Sales )
```

Пример: Внешнее выражение множества

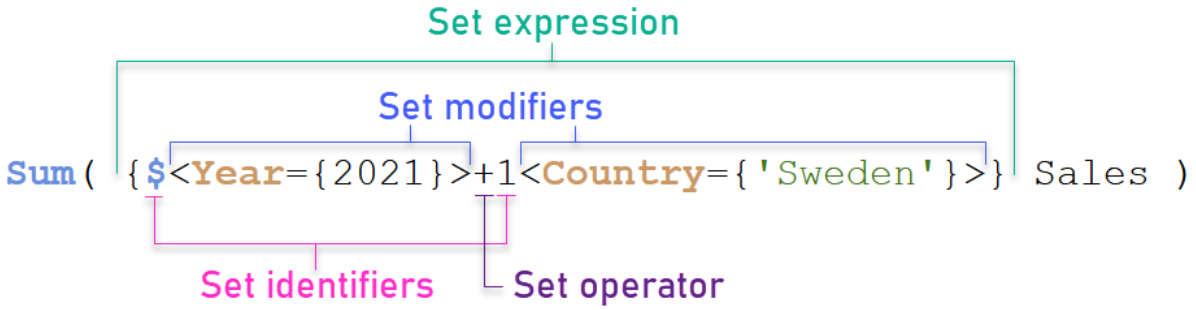
```
{<Year={2021}>} Sum(Sales) / Count(distinct Customer)
```

Выражение множества состоит из комбинации следующих элементов.

- **Identifiers**. 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`
- **Operators**. 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: `+`, `-`, `*`, `/`
- **Modifiers**. 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.

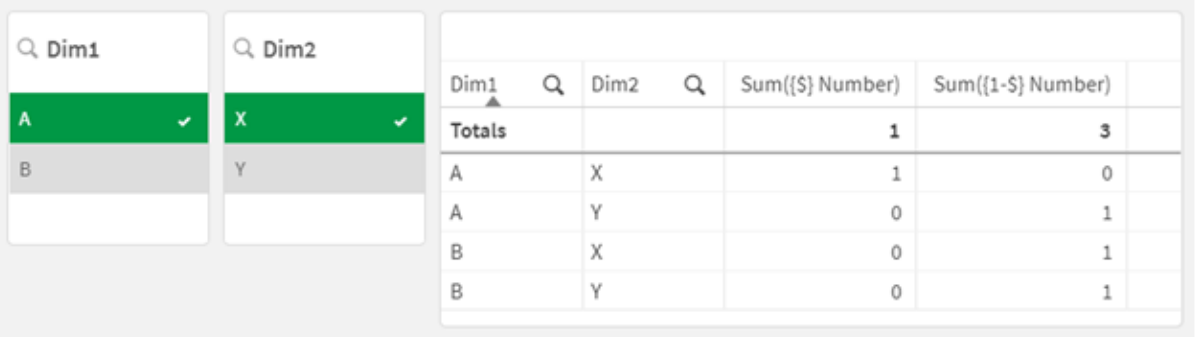
Однако выражение множества, использующее операторы множества, также представляет подмножество записей, но, как правило, уже не может быть описано как использующее выборку значений поля. Такое выражение является ненатуральным множеством.

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



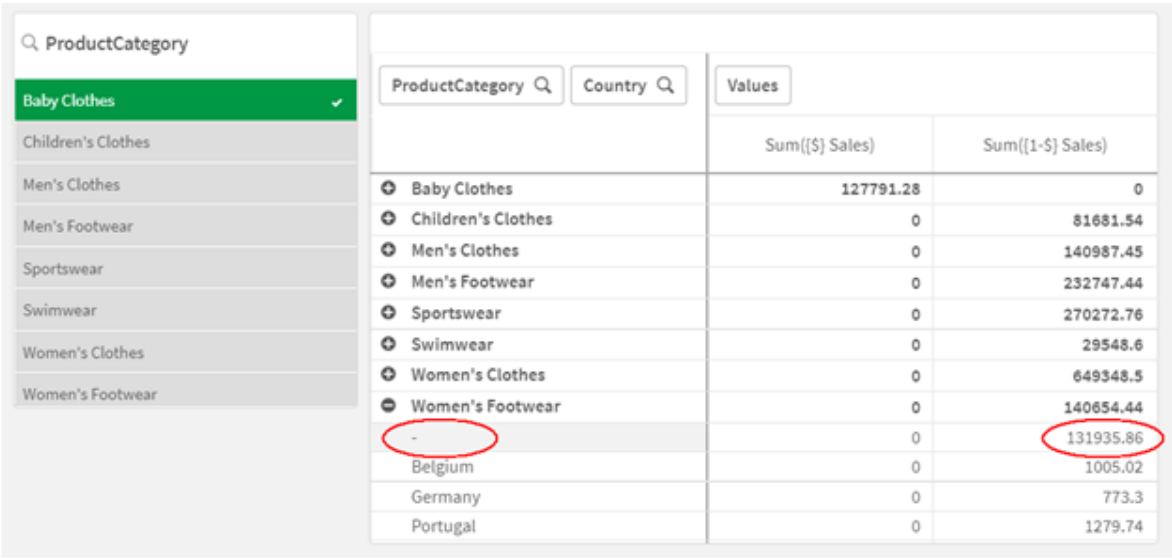
Выражение множества в первой мере использует натуральное множество: оно соответствует выборке, созданной в `{$}`.

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.

Идентификатор	Описание
1	Представляет полное множество всех записей в приложении, независимо от выборок.
$	Представляет записи текущей выборки. Выражение множества **{$}**, таким образом, эквивалентно неутверждению выражения множества.
$1	Представляет предыдущую выборку. $2 представляет предыдущую предпоследнюю выборку и т. д.
$_1	Представляет следующую (стоящую впереди) выборку. $_2 представляет следующую предпоследнюю выборку и т. д.
BM01	Можно использовать любой ID закладки или имя закладки.
MyAltState	Можно ссылаться на выборки, сделанные в другом состоянии, указав имя этого состояния.

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.

Пример	Результат
sum ({BM01} Sales)	Возвращает продажи для закладки с именем *BM01*.

Пример	Результат
sum({$<OrderDate = DeliveryDate>} Sales)	Возвращает продажи для текущей выборки, где OrderDate = DeliveryDate.
sum({1<Region = {US}>} Sales)	Возвращает продажи в регионе US, игнорируя текущую выборку.
sum({$<Region = >} Sales)	Возвращает продажи для выборки, но выборка в элементе *Region* удаляется.
sum({<Region = >} Sales)	Возвращает то же, что и в примере выше. Если множество для изменения отсутствует, используется знак $.
sum({$<Year={2000}, Region={“U*”}>} Sales)	Возвращает продажи для текущей выборки, но с новыми выборками в элементах *Year* и *Region*.

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.

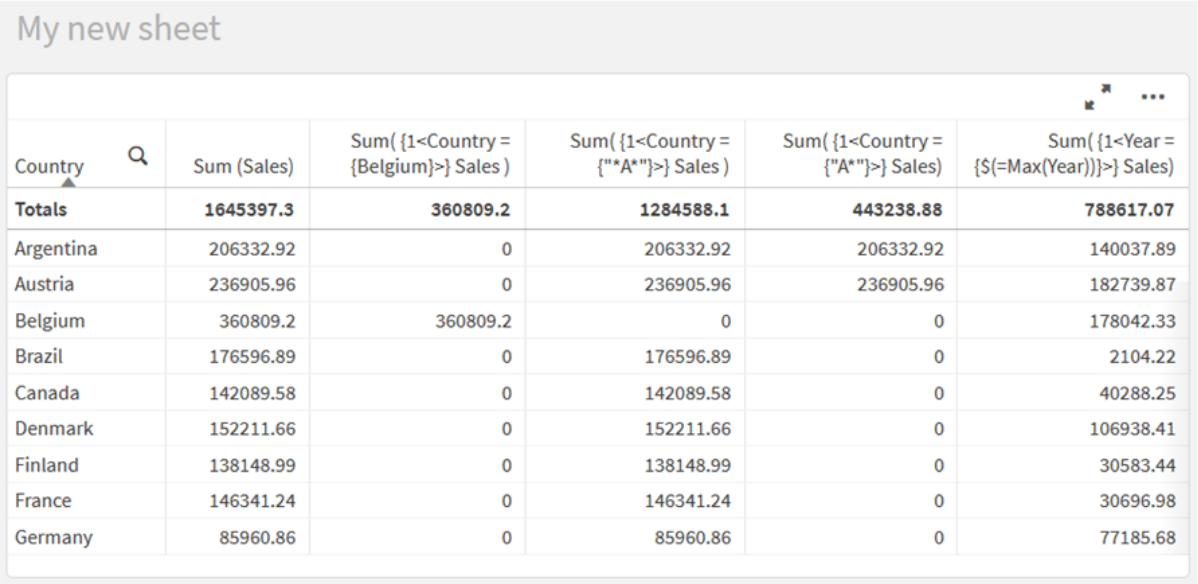
Таблица. Модификаторы множества на основе множеств элементов

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
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.

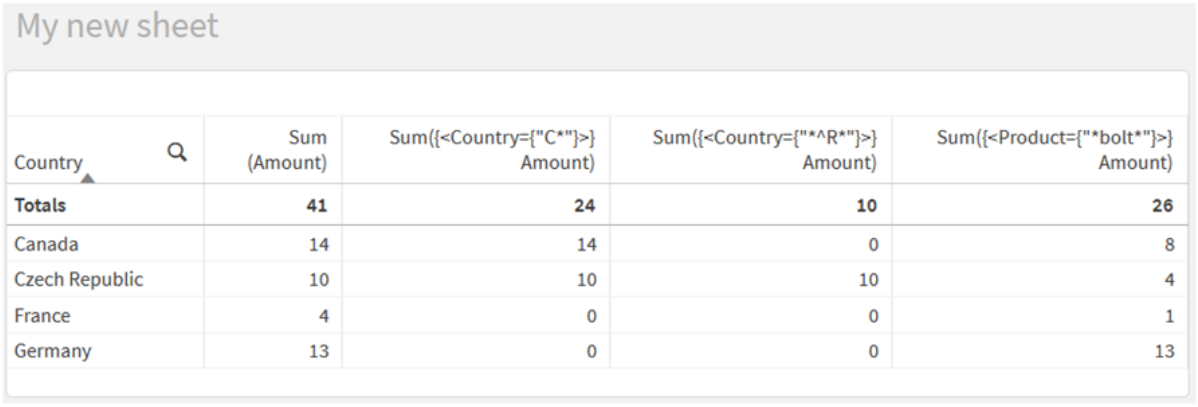
Таблица. Модификаторы множества с текстовыми поисками

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

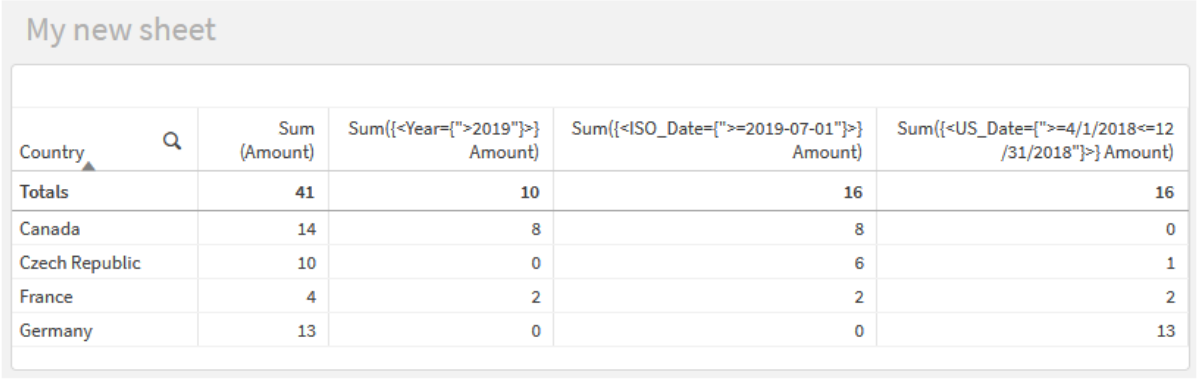
Country	Sum (Amount)	Sum({<Year={">2019"}>} Amount)	Sum({<ISO_Date={">=2019-07-01"}>} Amount)	Sum({<US_Date={">=4/1/2018<=12/31/2018"}>} Amount)
Итоги	**41**	**10**	**16**	**16**
Canada	14	8	8	0
Czech Republic	10	0	6	1
France	4	2	2	2
Germany	13	0	0	13

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



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

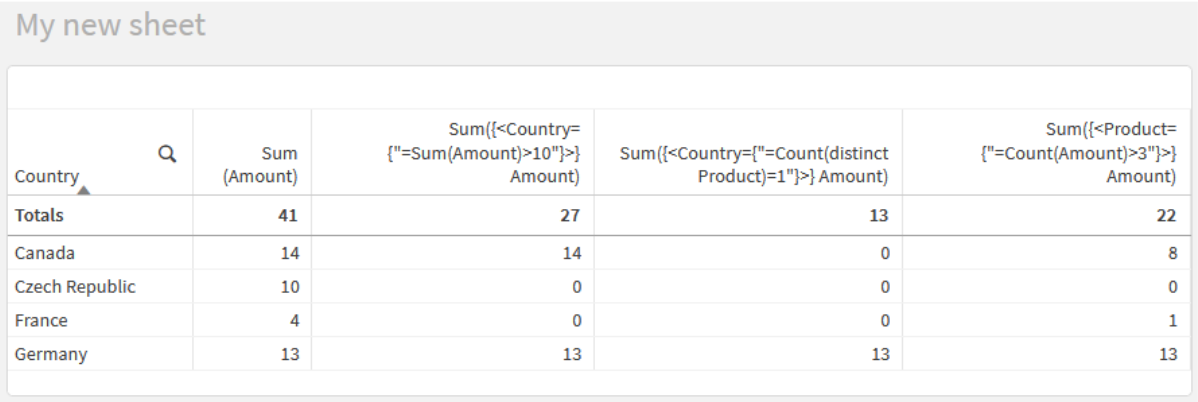
Объяснение

- 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

Примеры	Результаты
sum({$-1<Product = {“*Internal*”, “*Domestic*”}>} Sales)	Возвращает продажи для текущей выборки за исключением транзакций, относящихся к продуктам со строкой «Internal» или «Domestic» в имени продукта.
sum({$<Customer = {“=Sum ({1<Year = {2007}>} Sales) > 1000000”}>} Sales)	Возвращает продажи для текущей выборки, но с новой выборкой в поле «Customer»: только клиенты, общая сумма продаж которых в 2007 г. составила больше 1 000 000.

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.

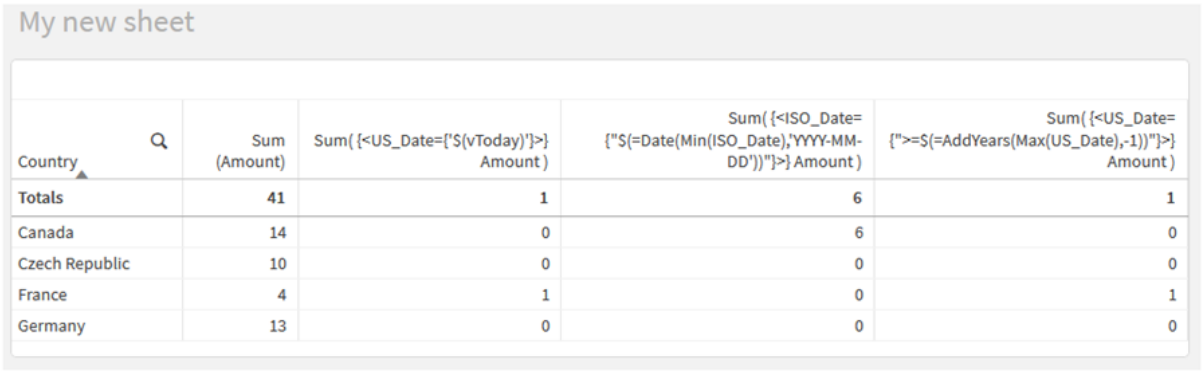
Таблица. Модификаторы множества с расширениями со знаком доллара

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
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


My new sheet

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)
Totals	41	1	6	1
Canada	14	0	6	0
Czech Republic	10	0	0	0
France	4	1	0	1
Germany	13	0	0	0

Примеры	Результаты
sum({$<Year = {$(#vLastYear)}>} Sales)	Возвращает продажи за предыдущий год в отношении текущей выборки. Здесь переменная vLastYear, содержащая соответствующий год, используется в расширении со знаком доллара.
sum({$<Year = {$(#=Only(Year)-1)}>} Sales)	Возвращает продажи за предыдущий год в отношении текущей выборки. Здесь расширение со знаком доллара используется для расчета предыдущего года.

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.

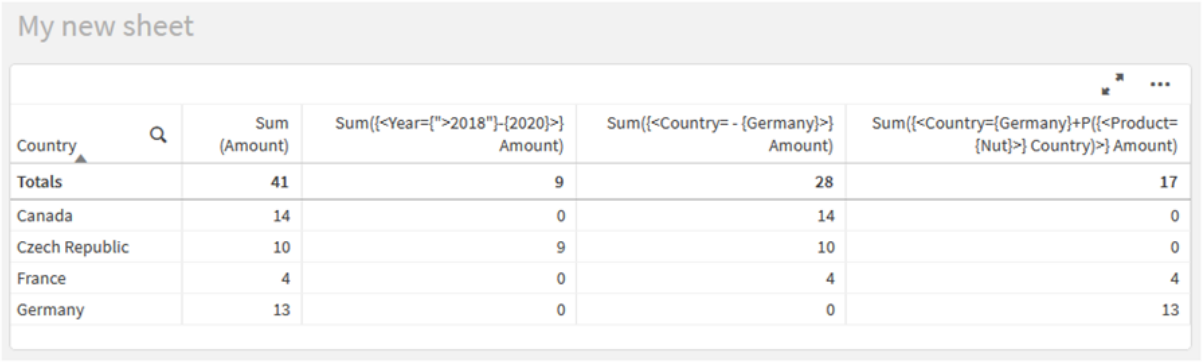
Таблица. Модификаторы множества с операторами множества

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



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

Примеры	Результаты
sum({$<Product = Product + {OurProduct1} - {OurProduct2} >} Sales)	Возвращает продажи для текущей выборки, но в список выбранных продуктов добавляется продукт «OurProduct1» и удаляется продукт «OurProduct2».
sum({$<Year = Year + ({“20*”,1997} - {2000}) >} Sales)	Возвращает продажи для текущей выборки, но с дополнительными выборками в поле «Year»: 1997 и все года, начинающиеся с «20», за исключением 2000. Обратите внимание, что в случае включения значения 2000 в текущую выборку, оно останется включенным и после изменения.
sum({$<Year = (Year + {“20*”,1997}) - {2000} >} Sales)	Возвращает практически все то же самое, что и выше, однако здесь значение 2000 будет исключено, даже если изначально оно было включено в текущую выборку. Пример демонстрирует важность использования в некоторых случаях скобок для определения очередности.

Примеры	Результаты
sum({$<Year = {“*”} - {2000}, Product = {“*bearing*”} >} Sales)	Возвращает продажи для текущей выборки, но с новой выборкой в поле «Year»: все года, кроме 2000; и только для продуктов, содержащих строку «bearing».

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

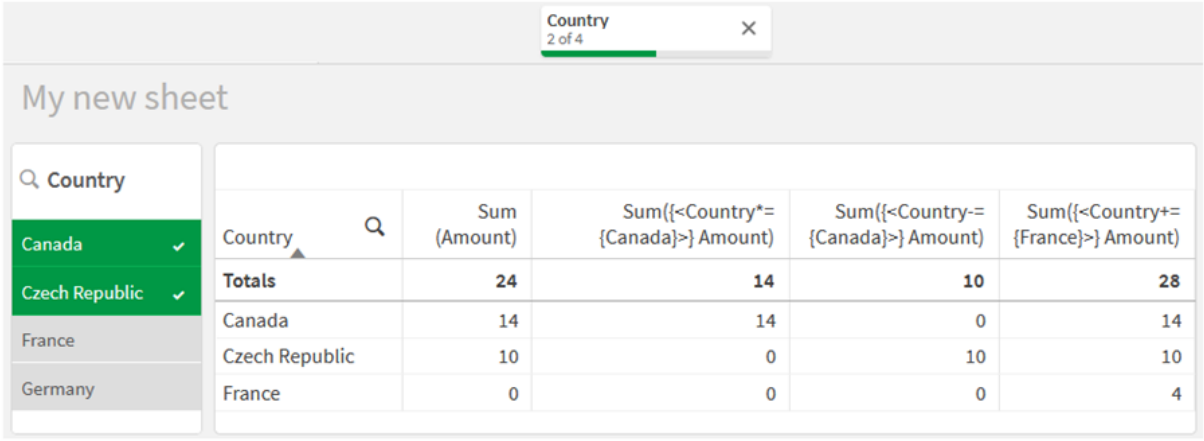
Объяснение

- 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



Примеры	Результаты
sum({$<Product += {OurProduct1, OurProduct2} >} Sales)	Возвращает продажи для текущей выборки, но с использованием неявного объединения для добавления продуктов «OurProduct1» и «OurProduct2» в список выбранных продуктов.
sum({$<Year += {“20*”,1997} - {2000} >} Sales)	Возвращает продажи для текущей выборки, но с использованием неявного объединения для добавления нескольких годов в выборку: 1997 и все годы, начинающиеся с «20», за исключением 2000. Обратите внимание, что в случае включения значения 2000 в текущую выборку, оно останется включенным и после изменения. То же, что и `<Year=Year + ({“20*”,1997}-{2000})>`.
sum({$<Product *= {OurProduct1} >} Sales)	Возвращает продажи для текущей выборки, но только для пересечения выбранных на данный момент продуктов и продукта «OurProduct1».

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.

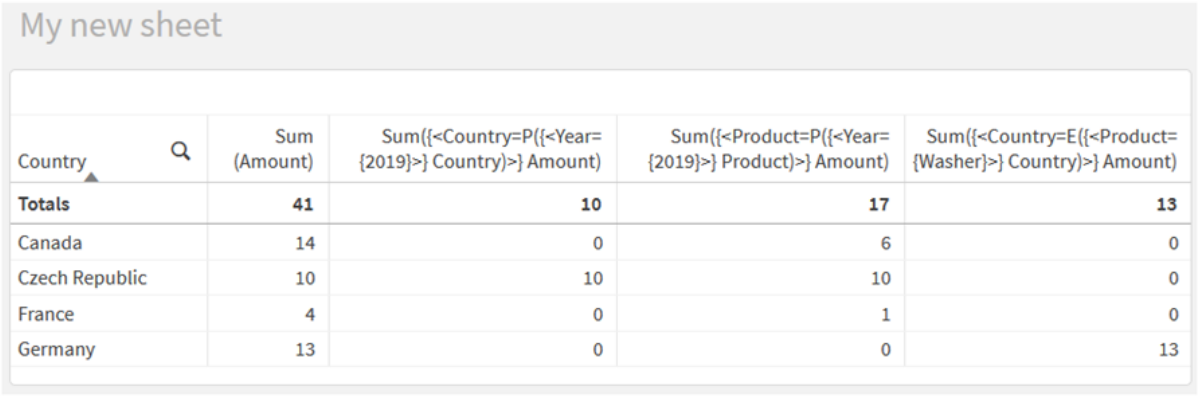
Таблица. Модификаторы множества с использованием функций множества

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



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

Примеры	Результаты
sum({$<Customer = P({1<Product= {'Shoe'}>} Customer)>} Sales)	Возвращает продажи для текущей выборки, но только для тех клиентов, которые когда-либо покупали продукт «Shoe». Здесь функция элемента P( ) возвращает список возможных клиентов, подразумеваемых выборкой «Shoe» в поле Product.
sum({$<Customer = P({1<Product= {'Shoe'}>})>} Sales)	Так же, как выше. Если в функции элемента поле опущено, функция вернет возможные значения для поля, указанного во внешнем назначении.
sum({$<Customer = P({1<Product= {'Shoe'}>} Supplier)>} Sales)	Возвращает продажи для текущей выборки, но только для тех клиентов, которые когда-либо поставляли продукт «Shoe», то есть клиент также является поставщиком. Здесь функция элемента P( ) возвращает список возможных поставщиков, подразумеваемых выборкой «Shoe» в поле Product. Список поставщиков затем используется в качестве выборки в поле Customer.
sum({$<Customer = E({1<Product= {'Shoe'}>})>} Sales)	Возвращает продажи для текущей выборки, но только для тех клиентов, которые никогда не покупали продукт «Shoe». Здесь функция элемента E( ) возвращает список исключенных клиентов, которые исключены выборкой «Shoe» в поле Product.

Внутренние и внешние выражения множества

Выражения множества могут использоваться во внутренних и внешних функциях агрегирования и заключаются в фигурные скобки.

Когда выражение множества используется внутри функции агрегирования, оно может выглядеть следующим образом:

Пример: Внутреннее выражение множества

```
Sum( {$<Year={2021}>} Sales )
```

Используйте выражение множества вне функции агрегирования, если имеются выражения с несколькими агрегированиями и нужно избежать написания одного и того же выражения множества в каждой функции агрегирования.

Если используется внешнее выражение множества, оно должно быть помещено в начало определения области.

Пример: Внешнее выражение множества

```
{<Year={2021}>} Sum(Sales) / Count(distinct Customer)
```

Если используется выражение множества вне функции агрегирования, его также можно применить к существующим основным мерам.

Пример: Внешнее выражение множества применяется к основной мере

```
{<Year={2021}>} [Master Measure]
```

Выражение множества, используемое вне функций агрегирования, влияет на все выражение, если оно не заключено в скобки, тогда скобки определяют область. В приведенном ниже примере определения лексической области выражение множества применяется только к агрегированию внутри скобок.

Пример: Определение лексической области

```
( {<Year={2021}>} Sum(Amount) / Count(distinct Customer) ) - Avg(CustomerSales)
```

Правила

Лексическая область

Выражение множества влияет на все выражение, если оно не заключено в скобки. В этом случае скобки определяют лексическую область.

Позиция

Выражение множества должно быть помещено в начало определения лексической области.

Контекст

Контекст — это выборка, имеющая отношение к выражению. Традиционно контекст всегда был состоянием по умолчанию для текущей выборки. Но если для объекта установлено альтернативное состояние, контекстом будет альтернативное состояние текущей выборки.

Контекст также можно определить в виде внешнего выражения множества.

Наследование

Внутренние выражения множества имеют приоритет над внешними. Если внутреннее выражение множества содержит идентификатор множества, он заменяет контекст. В противном случае контекст и выражение множества будут объединены.

- {$<SetExpression>} – переопределяет внешнее выражение множества
- {<SetExpression>} – объединяется с внешним выражением множества

Назначение множества элементов

Назначение множества элементов определяет способ объединения двух выборок. Если используется обычный знак равенства, выборка во внутреннем выражении множества имеет приоритет. В противном случае будет использоваться неявный оператор множества.

- {<Field={value}>} – эта внутренняя выборка заменяет любую внешнюю выборку в "Field".
- {<Field+={value}>} – эта внутренняя выборка объединяется с внешней выборкой в "Field" с помощью оператора объединения.
- {<Field*={value}>} – эта внутренняя выборка объединяется с внешней выборкой в "Field" с помощью оператора пересечения.

Наследование в несколько этапов

Наследование может происходить в несколько этапов. Примеры:

- Текущая выборка → Sum(Amount)
  Функция агрегирования будет использовать контекст, который здесь является текущей выборкой.
- Текущая выборка → {<Set1>} Sum(Amount)
  Set1 будет наследоваться от текущей выборки, а результат будет контекстом для функции агрегирования.
- Текущая выборка → {<Set1>} ({<Set2>} Sum(Amount))
  Set2 будет наследоваться от множества Set1, которое в свою очередь будет наследоваться от текущей выборки, а результат будет контекстом для функции агрегирования.

Функция Aggr()

Функция Aggr() создает вложенное агрегирование, которое имеет два независимых агрегирования. В приведенном ниже примере Count() вычисляется для каждого значения Dim и полученный массив агрегируется с помощью функции Sum().

Пример:

```
Sum(Aggr(Count(X),Dim))
```

Count() – внутреннее агрегирование, а Sum() – внешнее агрегирование.

- Внутреннее агрегирование не наследует никакого контекста от внешнего агрегирования.
- Внутреннее агрегирование наследует контекст от функции Aggr(), которая может содержать выражение множества.
- И функция Aggr(), и функция внешнего агрегирования наследуют контекст от внешнего выражения множества.

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.

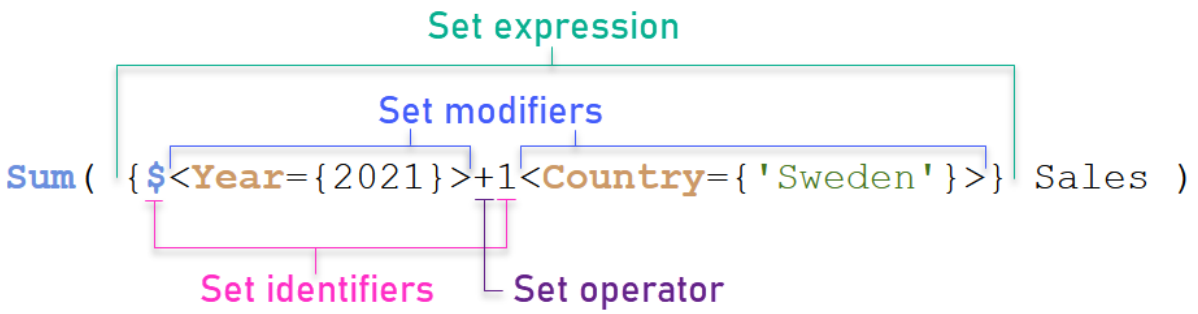
Что требуется сделать перед началом работы

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 322)*
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 )
```

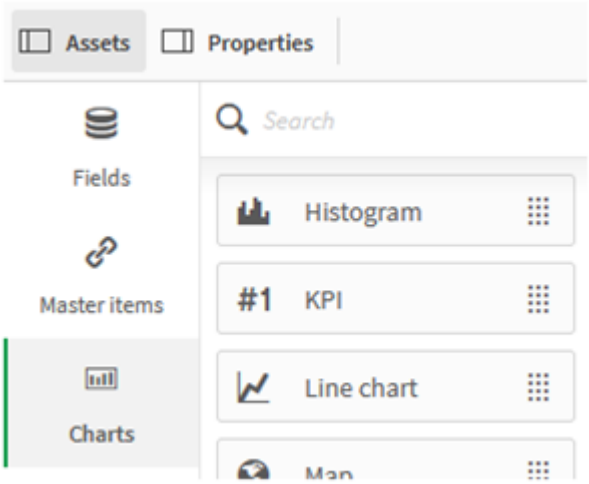
Модификатор:

```
<Year = {2015}>
```

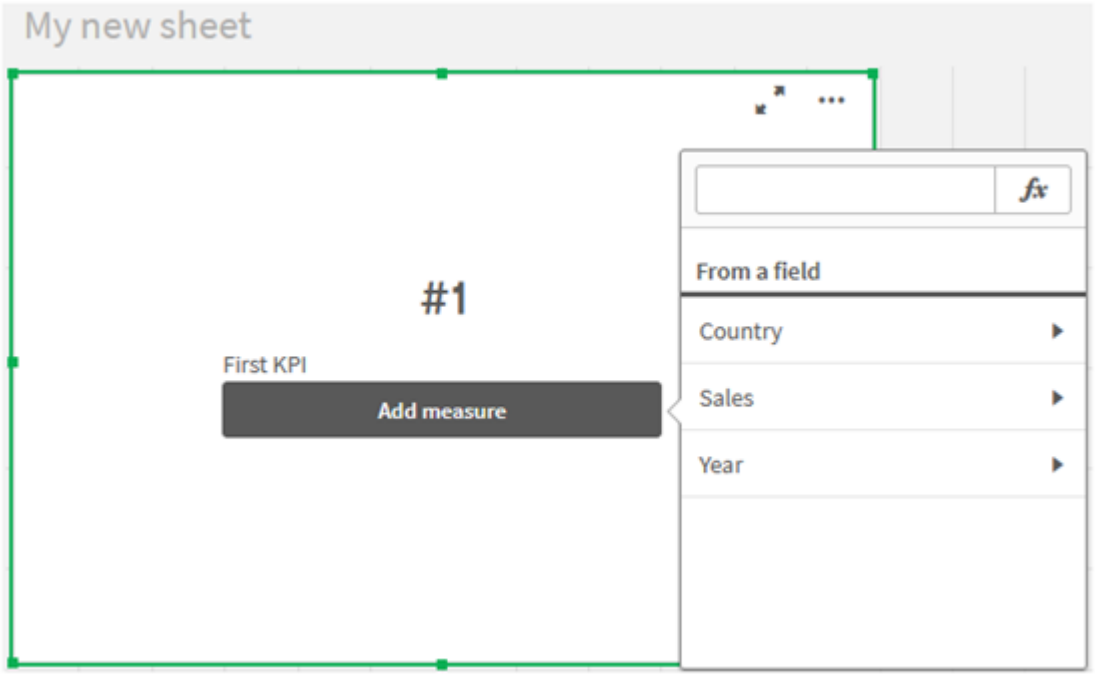
Этот модификатор задает выборку данных из 2015 года. Фигурные скобки, в которые заключен модификатор, обозначают выражение множества.

Выполните следующие действия.

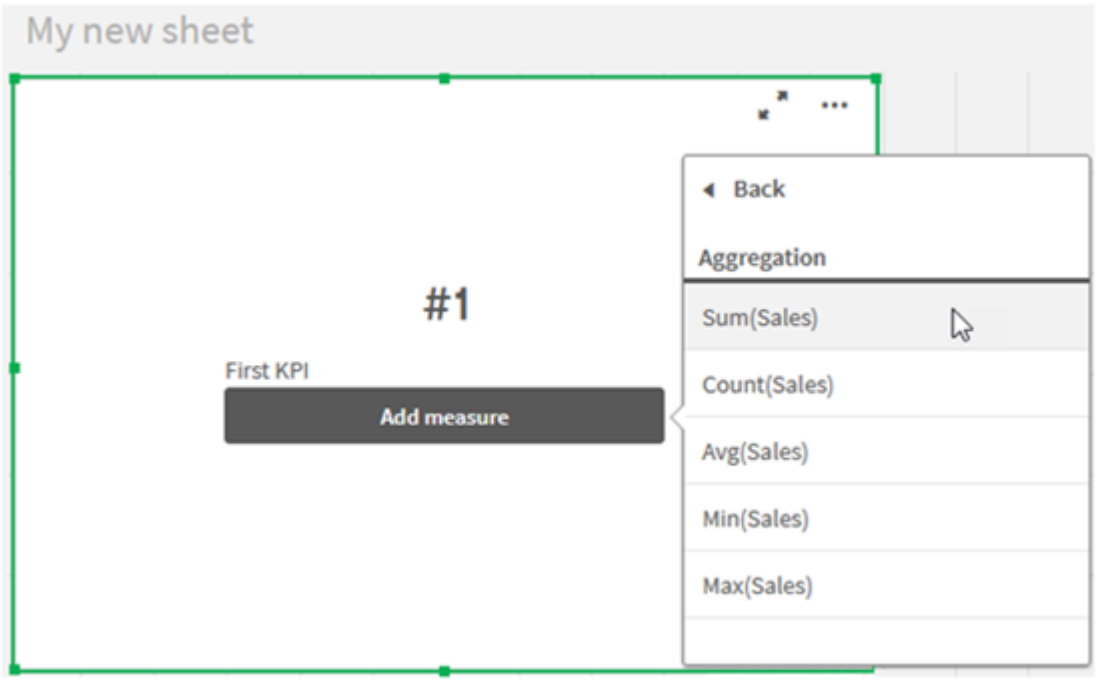
1. На листе откройте панель **Ресурсы** с панели навигации, затем выберите **Диаграммы**.



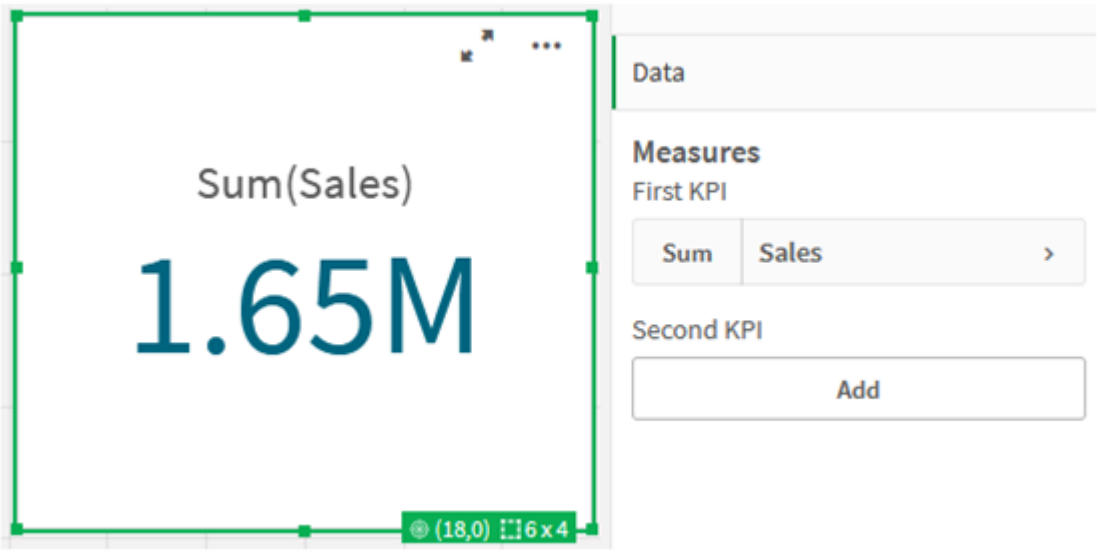
2. Перетащите **ключевой показатель эффективности** на лист, затем щелкните **Добавить меру**.



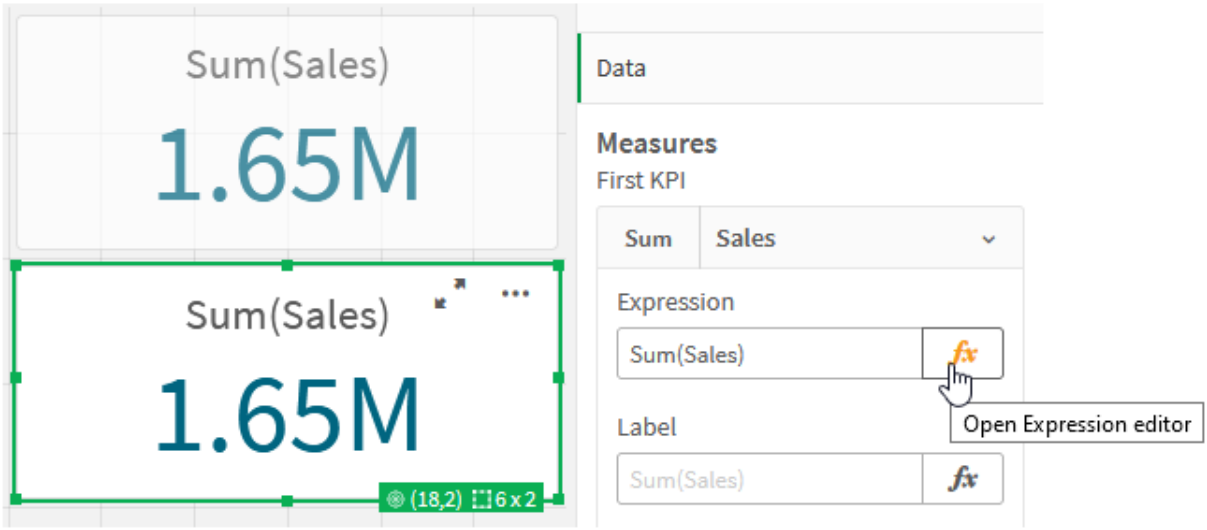
3. Щелкните `Sales`, затем выберите `Sum(Sales)` для агрегирования.



KPI показывает сумму объемов продаж за все годы.



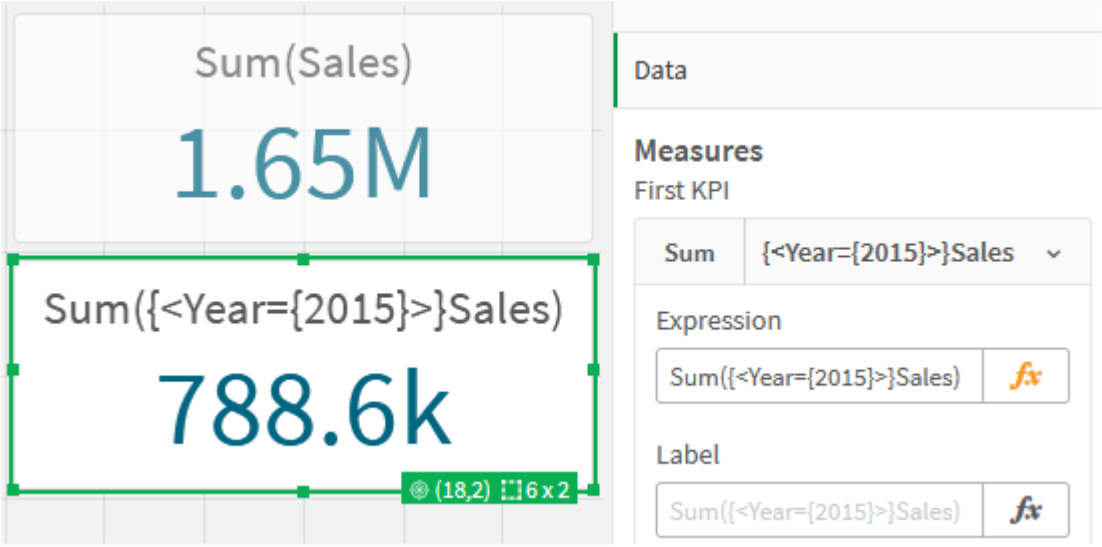
4. Скопируйте и вставьте KPI, чтобы создать новый объект KPI.
5. Щелкните созданный KPI выберите **Продажи** в области **Меры** и затем щелкните **Открыть редактор выражения**.



Редактор выражения открывается с агрегацией `Sum(Sales)`.

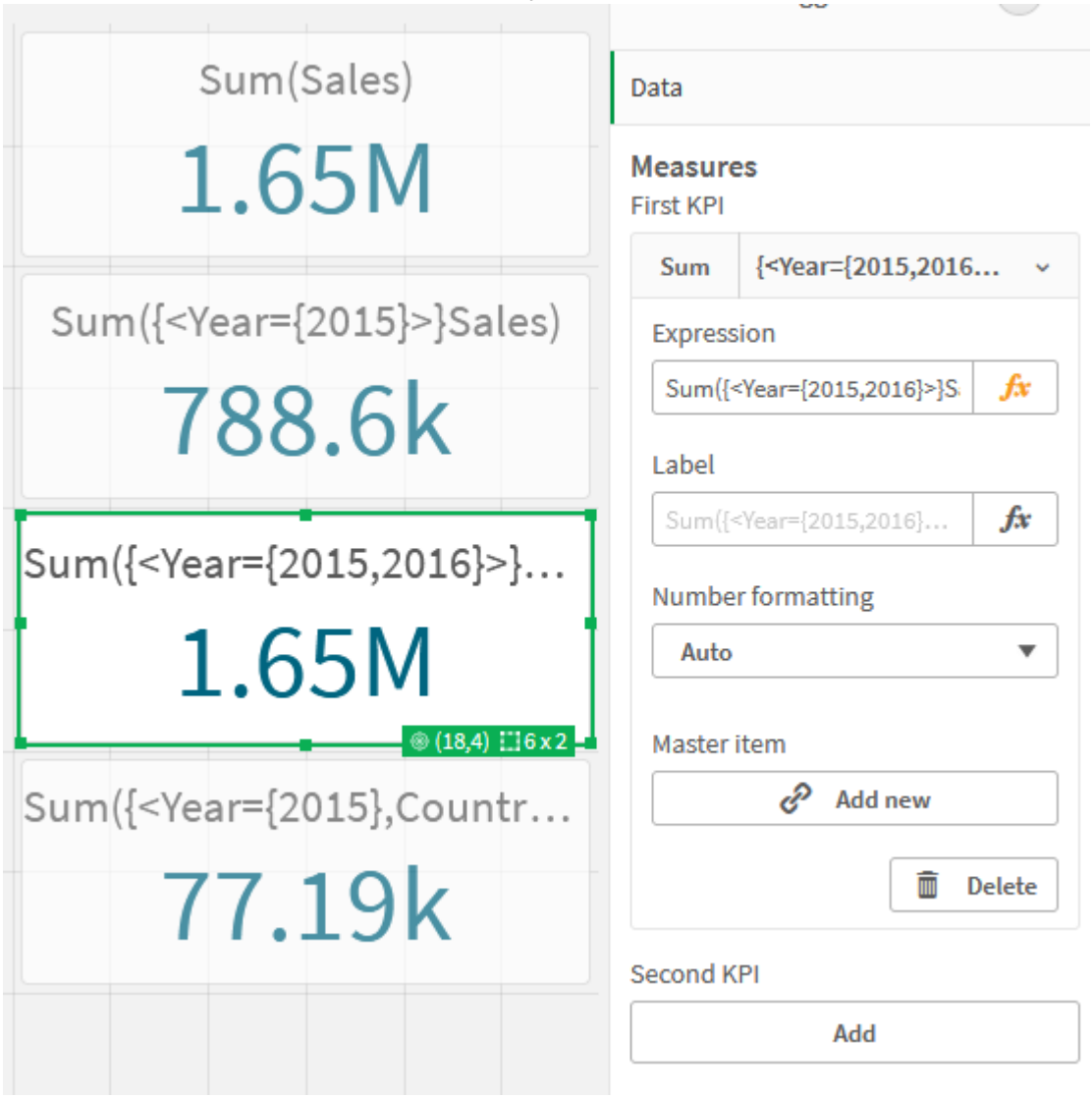
6. В редакторе выражения создайте выражение, чтобы суммировать Sales только за 2015 год.
   i. Добавьте фигурные скобки, чтобы обозначить выражение множества: `Sum({}Sales)`
   i. Добавьте угловые скобки, чтобы обозначить модификатор множества: `Sum({<>}Sales)`
   ii. В угловых скобках добавьте поле, 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:

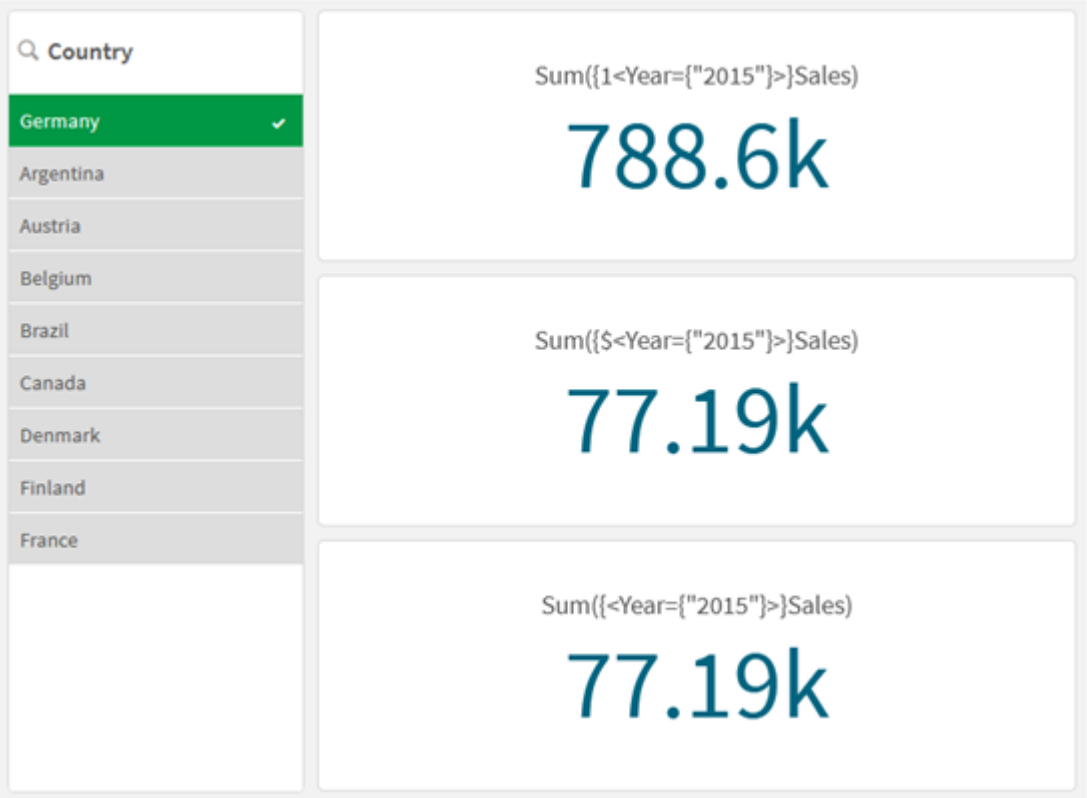
```
Sum({$<Year={"2015"}>}Sales
```

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.

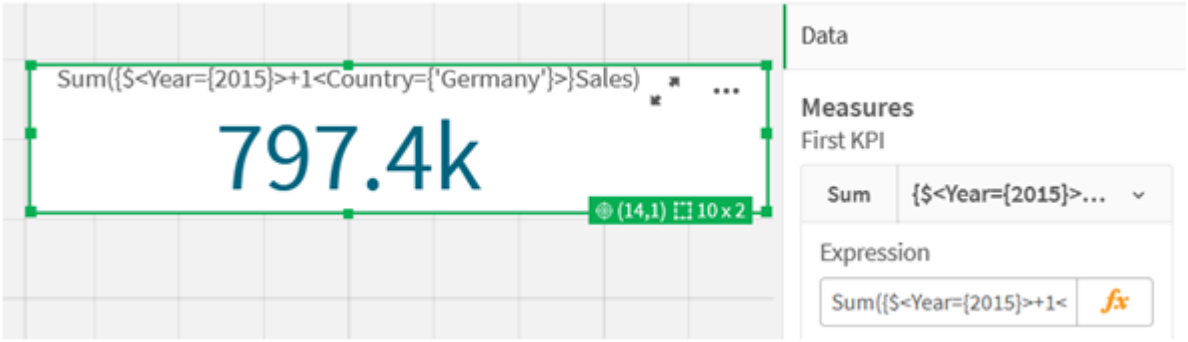
Выполните следующие действия.

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» (+)

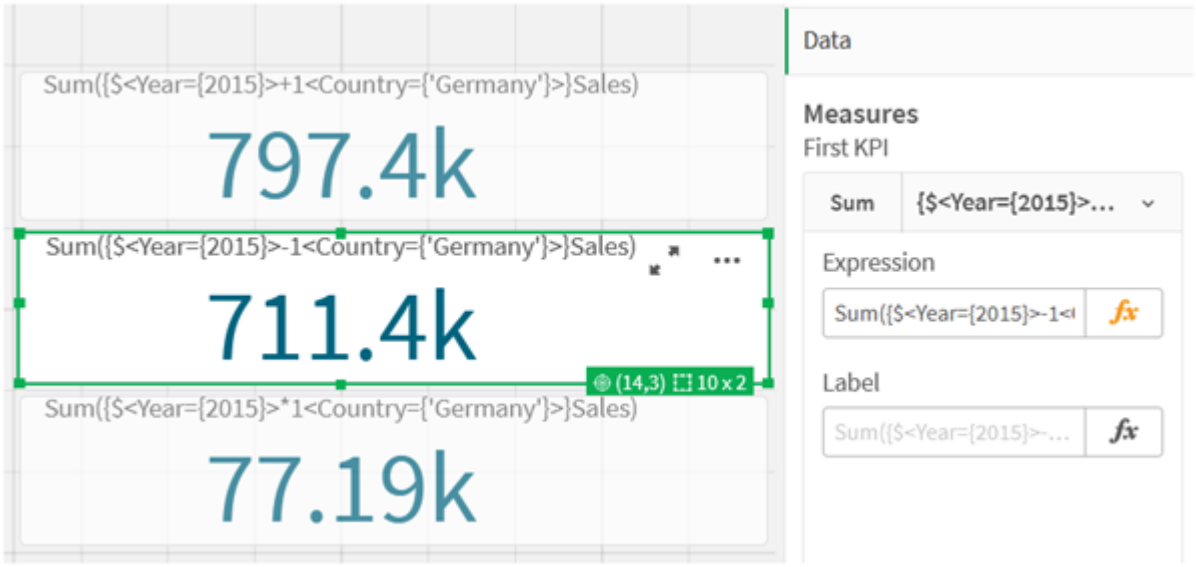


Также можно использовать знак «минус» (`(-)`), чтобы вернуть набор данных, включающий записи, которые относятся к 2015 году, но не к Germany. Или используйте звездочку (`(*)`), чтобы вернуть множество, содержащее записи, относящиеся к обоим множествам.

```
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.

Присваивая имя сущности, старайтесь не использовать одно и то же имя для нескольких полей, переменных или мер. Существует строгий порядок очередности при разрешении конфликтов между сущностями с идентичными именами. Этот порядок отражается во всех объектах и контекстах, в которых используются такие сущности. Этот порядок приоритета выглядит следующим образом:

- Внутри агрегирования у поля есть приоритет над переменной. Метки мер не являются релевантными в агрегированиях и не приоритизируются.
- Вне агрегирования у метки меры есть приоритет над переменной, у которой в свою очередь есть приоритет над полем.
- Кроме того, вне агрегирования меру можно использовать повторно путем ссылки на его метку, если метка по сути не является вычисляемой. В такой ситуации мера теряет часть значимости, чтобы сократить риск создания самоссылки, и в данном случае имя всегда будет интерпретироваться, во-первых, как метка меры, во-вторых, как имя поля и, в-третьих, как имя переменной.

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

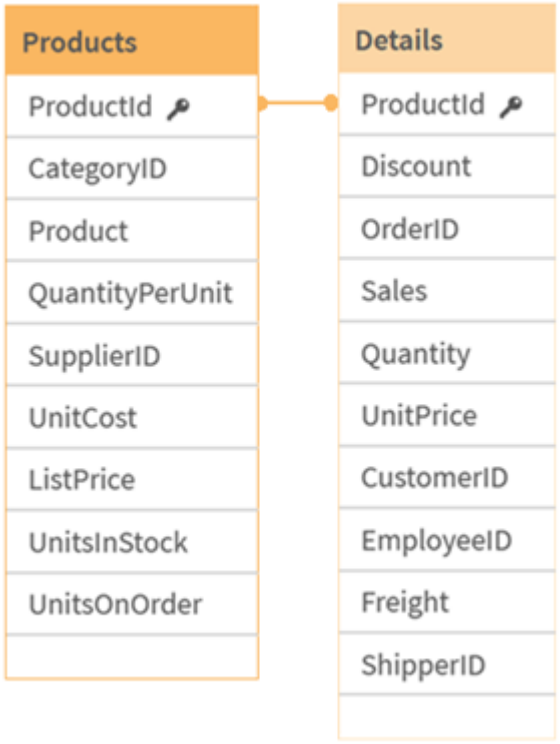
Способ вычисления агрегирований определяет, что нельзя агрегировать ключевые поля, потому что не ясно, какая таблица должна использоваться для агрегирования. Например, если поле <Key> связывает две таблицы, не ясно, возвратит ли **Count**(<Key>) количество записей в первой или второй таблице.

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:

p *FirstSortedValue – функция диаграммы (page 338)*
p *RangeMax (page 1365)*

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:

p *FirstSortedValue – функция диаграммы (page 338)*
p *RangeMin (page 1369)*

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:

p *Avg – функция диаграммы (page 407)*
p *Median – функция диаграммы (page 447)*

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

Функция скрипта **NPV()** принимает процентную скидку и несколько значений, упорядоченных по периоду. Поступления (доходы) имеют положительное значение, а расходы (будущие платежи) имеют отрицательное значение в рамках этих вычислений. Они производятся в конце каждого периода.

```
NPV (rate, expression)
```

XNPV

Функция скрипта **XNPV()** принимает конкретные даты, соответствующие каждому движению денежных средств с применением скидки, отличной от указанной процентной скидки. Она отличается от функции **NPV()**, так как **NPV()** рассматривает все периоды времени как равные. По этой причине **XNPV()** дает более точный результат, чем **NPV()**.

```
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 390)([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:

p *XIRR – функция диаграммы (page 390)*
p *Aggr – функция диаграммы (page 555)*

NPV

Функция скрипта **NPV()** принимает процентную скидку и несколько значений, упорядоченных по периоду. Поступления (доходы) имеют положительное значение, а расходы (будущие платежи) имеют отрицательное значение в рамках этих вычислений. Они производятся в конце каждого периода.

Чистая текущая стоимость (Net Present Value, NPV) используется для расчета текущей общей стоимости будущих потоков движения денежных средств. Чтобы рассчитать NPV, необходимо оценить будущие денежные потоки для каждого периода и определить правильный процент скидки. Функция скрипта **NPV()** принимает процентную скидку и несколько значений, упорядоченных по периоду. Поступления (доходы) имеют положительное значение, а расходы (будущие платежи) имеют отрицательное значение в рамках этих вычислений. Они производятся в конце каждого периода.

Синтаксис:

```
NPV(discount_rate, value)
```

Возвращаемые типы данных: числовой. По умолчанию результат будет отформатирован как валюта.

Используется следующая форму расчета чистой текущей стоимости:

$$NPV = \sum_{t=1}^{n} \frac{R_t}{(1+i)^t}$$

где:

- `R t` = чистый приход-расход денежных средств в течение одного периода`t`
- `i` = процент скидки или возврата, который можно получить при альтернативных инвестициях
- `t` = количество периодов времени

Аргументы

Аргумент	Описание
discount_rate	**discount_rate** – это процентная ставка примененной скидки. Значение 0,1 обозначает скидку 10%.
value	Это поле содержит значения для нескольких периодов, перечисленные через запятую. Первое значение представляет денежный поток на конец периода 1 и т. д.

Ограничения:

Функция `NPV()` имеет следующие ограничения:

- Текстовые значения, значения NULL и отсутствующие значения игнорируются.
- Значения денежного потока должны перечисляться по возрастанию периодов.

Когда это следует использовать

`NPV()` – это финансовая функция, предназначенная для проверки рентабельности проекта и для получения производных мер. Эта функция полезна, когда сведения о денежных потоках доступны в формате необработанных данных.

Региональные настройки

Если не указано иное, в примерах, приведенных в данном разделе, используется следующий формат даты: ММ/ДД/ГГГГ. Формат даты указан в операторе `SET DateFormat` скрипта загрузки данных. В вашей системе может быть установлен другой формат даты по умолчанию в зависимости от региональных настроек и других факторов. Можно изменить формат в примерах в соответствии с потребностями. Или можно изменить форматы в скрипте загрузки в соответствии с этими примерами.

Региональные настройки по умолчанию в приложениях основаны на системных региональных настройках компьютера или сервера, где установлено ПО Qlik Sense. Если на сервере Qlik Sense, к которому обращается пользователь, выбран шведский язык, то редактор загрузки данных будет использовать шведские региональные настройки для даты, времени и валюты. Эти параметры регионального формата не связаны с языком, отображаемым в интерфейсе пользователя Qlik Sense. Qlik Sense будет отображаться на языке, который используется в браузере.

Пример 1. Один платеж (скрипт)

Скрипт загрузки и результаты

Обзор

Откройте редактор загрузки данных и добавьте приведенный ниже скрипт загрузки на новую вкладку.

Скрипт загрузки содержит следующее:

- Набор данных одного проекта и его денежного потока за один период, загруженный в таблицу под именем `CashFlow`.
- Резидентная загрузка из таблицы `CashFlow`, используемая при расчете поля NPV для проекта в таблице под именем `NPV`.
- Жестко запрограммированный процент скидки 10%, используемый при расчете NPV.
- Оператор `Group By`, используемый для группировки всех платежей в проекте.

Скрипт загрузки

```
CashFlow:
Load
*
Inline
[
PrjId,PeriodId,Values
1,1,1000
];

NPV:
Load
       PrjId,
       NPV(0.1,Values) as NPV //Discount Rate of 10%
Resident CashFlow
Group By PrjId;
```

Результаты

Загрузите данные и откройте лист. Создайте новую таблицу и добавьте эти поля как измерения:

- `PrjId`
- `NPV`

Результирующая таблица

PrjId	NPV
1	$909.09

Если имеется один платеж $1000 к получению в конце одного периода со скидкой 10% за период, значение NPV = $1000 / (1 + процент скидки). Эффективное значение NPV равно $909.09

Пример 2. Несколько платежей (скрипт)

Скрипт загрузки и результаты

Обзор

Откройте редактор загрузки данных и добавьте приведенный ниже скрипт загрузки на новую вкладку.

Скрипт загрузки содержит следующее:

- Набор данных одного проекта и его денежного потока за несколько периодов, загруженный в таблицу под именем `CashFlow`.
- Резидентная загрузка из таблицы `CashFlow`, используемая при расчете поля NPV для проекта в таблице под именем `NPV`.
- Жестко запрограммированный процент скидки 10% (0,1), используемый при расчете NPV.
- Оператор `Group By`, используемый для группировки всех платежей в проекте.

Скрипт загрузки

```
CashFlow:
Load
*
Inline
[
PrjId,PeriodId,Values
1,1,1000
1,2,1000
];

NPV:
Load
       PrjId,
       NPV(0.1,Values) as NPV //Discount Rate of 10%
Resident CashFlow
Group By PrjId;
```

Результаты

Загрузите данные и откройте лист. Создайте новую таблицу и добавьте эти поля как измерения:

- `PrjId`
- `NPV`

Результирующая таблица

PrjId	NPV
1	$1735.54

Если имеются платежи по $1000 к получению в конце двух периодов со скидкой 10 % за период, эффективное значение NPV = $1735.54.

Пример 3. Несколько платежей (скрипт)

Скрипт загрузки и результаты

Обзор

Откройте редактор загрузки данных и добавьте приведенный ниже скрипт загрузки на новую вкладку.

Скрипт загрузки содержит следующее:

- Процентные ставки скидки для двух проектов, загруженных в таблицу под именем `Project`.
- Денежные потоки за несколько периодов для каждого проекта по идентификаторам проекта и периода. Этот идентификатор периода можно использовать для упорядочения записей, если данные не упорядочены сразу.
- Комбинация `NoConcatenate`, резидентных загрузок и функции `Left Join` для создания временной таблицы `tmpNPV`. В таблице объединены записи таблиц `Project` и `CashFlow` в одну плоскую таблицу. В этой таблице процентные ставки скидки будут повторяться для каждого периода.
- Резидентная загрузка из таблицы `tmpNPV`, используемая при расчете поля NPV для каждого проекта в таблице под именем `NPV`.
- С каждым проектом связана одна процентная ставка скидки стоимости. Она извлекается с помощью функции `only()` и используется при расчете NPV для каждого проекта.
- Оператор `Group By`, используемый для группировки всех платежей в каждом проекте по идентификатору проекта.

Для предотвращения загрузки синтетических или избыточных данных в модель данных, таблица `tmpNPV` удаляется в конце скрипта.

Скрипт загрузки

```
Project:
Load * inline [
PrjId,Discount_Rate
1,0.1
2,0.15
];

CashFlow:
Load
*
Inline
[
PrjId,PeriodId,Values
1,1,1000
1,2,1000
1,3,1000
2,1,500
2,2,500
2,3,1000
```

```
2,4,1000
];

tmpNPV:
NoConcatenate Load *
Resident Project;
Left Join
Load *
Resident CashFlow;

NPV:
Load
       PrjId,
       NPV(Only(Discount_Rate),Values) as NPV //Discount Rate will be 10% for Project 1 and 15% for
Project 2
Resident tmpNPV
Group By PrjId;

Drop table tmpNPV;
```

Результаты

Загрузите данные и откройте лист. Создайте новую таблицу и добавьте эти поля как измерения:

- PrjId
- NPV

Результирующая таблица

PrjId	NPV
1	$2486.85
2	$2042.12

Для идентификатора проекта 1 ожидаются платежи по $1000 к получению в конце трех периодов со скидкой 10% за период. Таким образом, эффективное значение NPV = $2486.85.

Для идентификатора проекта 2 ожидаются два платежа по $500 и еще два платежа по $1000 за четыре периода со скидкой 15%. Таким образом, эффективное значение NPV = $2042.12.

Пример 4. Пример рентабельности проекта (скрипт)

Скрипт загрузки и результаты

Обзор

Откройте редактор загрузки данных и добавьте приведенный ниже скрипт загрузки на новую вкладку.

Скрипт загрузки содержит следующее:

- Процентные ставки скидки и начальные капиталовложения (период 0) для двух проектов, загруженных в таблицу под именем `Project`.
- Денежные потоки за несколько периодов для каждого проекта по идентификаторам проекта и периода. Этот идентификатор периода можно использовать для упорядочения записей, если данные не упорядочены сразу.
- Комбинация `NoConcatenate`, резидентных загрузок и функции `Left Join` для создания временной таблицы `tmpNPV`. В таблице объединены записи таблиц `Project` и `CashFlow` в одну плоскую таблицу. В этой таблице процентные ставки скидки будут повторяться для каждого периода.
- С каждым проектом связана одна процентная ставка скидки, которая извлекается с помощью функции `only()` и используется при расчете NPV для каждого проекта.
- Резидентная загрузка из таблицы tmpNPV, используемая при расчете поля NPV для каждого проекта в таблице под именем NPV.
- Дополнительное поле, которое делит NPV на начальное капиталовложение каждого проекта, создается для расчета индекса рентабельности проекта.
- Оператор group by, используемый для группировки по идентификатору проекта, применяется с целью группировки всех платежей для каждого проекта.

Для предотвращения загрузки синтетических или избыточных данных в модель данных, таблица `tmpNPV` удаляется в конце скрипта.

Скрипт загрузки

```
Project:
Load * inline [
PrjId,Discount_Rate, Initial_Investment
1,0.1,100000
2,0.15,100000
];

CashFlow:
Load
*
Inline
[
PrjId,PeriodId,Values,
1,1,35000
1,2,35000
1,3,35000
2,1,30000
2,2,40000
2,3,50000
2,4,60000
];

tmpNPV:
NoConcatenate Load *
Resident Project;
Left Join
Load *
```

```
Resident CashFlow;

NPV:
Load
        PrjId,
        NPV(Only(Discount_Rate),Values) as NPV, //Discount Rate will be 10% for Project 1 and
15% for Project 2
    NPV(Only(Discount_Rate),Values)/ Only(Initial_Investment) as Profitability_Index
Resident tmpNPV
Group By PrjId;

Drop table tmpNPV;
```

Результаты

Загрузите данные и откройте лист. Создайте новую таблицу и добавьте эти поля как измерения:

- PrjId
- NPV

Создайте следующую меру:

```
=only(Profitability_Index)
```

Результирующая таблица

PrjId	NPV	=only(Profitability_Index)
1	$87039.82	0.87
2	$123513.71	1.24

Для идентификатора проекта 1 эффективное значение NPV равно $87039.82, а начальное капиталовложение – $100000. Таким образом, индекс рентабельности равен 0.87. Так как он меньше 1, проект является нерентабельным.

Для идентификатора проекта 2 эффективное значение NPV равно $123513.71, а начальное капиталовложение – $100000. Таким образом, индекс рентабельности равен 1.24. Так как он больше 1, проект является рентабельным.

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 По умолчанию результат будет отформатирован как валюта.

Аргументы:

Аргументы

Аргумент	Описание
discount_rate	**discount_rate** is the rate of discount over the length of the period.**discount_rate** – это процентная ставка примененной скидки.
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:

p *XNPV – функция диаграммы (page 397)*
p *Aggr – функция диаграммы (page 555)*

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:

p *IRR – функция диаграммы (page 378)*
p *Aggr – функция диаграммы (page 555)*

XNPV

Функция скрипта **XNPV()** принимает конкретные даты, соответствующие каждому движению денежных средств с применением скидки, отличной от указанной процентной скидки. Она отличается от функции **NPV()**, так как **NPV()** рассматривает все периоды времени как равные. По этой причине **XNPV()** дает более точный результат, чем **NPV()**.

Синтаксис:

```
XNPV(discount_rate, pmt, date)
```

Возвращаемые типы данных: числовой. По умолчанию результат будет отформатирован как валюта.

Для расчета XNPV используется следующая формула:

$$XNPV = \sum_{i=1}^{n} \frac{P_i}{(1+rate)^{(di-d1)/365}}$$

где:

- P_i = чистый приход-расход денежных средств в течение одного периода `i`
- d_1 = дата первого платежа
- d_i = дата `i`-го платежа
- `rate` = процент скидки

Чистая текущая стоимость (Net Present Value, NPV) используется для расчета текущей общей стоимости будущих потоков движения денежных средств. Чтобы рассчитать NPV, необходимо оценить будущие денежные потоки для каждого периода и определить правильный процент скидки.

`XNPV()` принимает процентную скидку и несколько значений, упорядоченных по периоду. Приходы (доходы) имеют положительное значение, а расходы (будущие платежи) имеют отрицательное значение. Они производятся в конце каждого периода.

Аргументы

Аргумент	Описание
discount_rate	**discount_rate** – это процентная ставка примененной скидки. Значение 0,1 соответствует скидке 10%.
value	Это поле содержит значения денежного потока. Первое значение – это денежный поток в начале, соответствующая дата используется в качестве ссылки для расчета текущего значения для всех будущих денежных потоков. *XNPV() не применяет скидку к начальному значению денежного потока. Последующие платежи учитываются на основе года с 365 днями. Эта функция отличается от **NPV()**, которая применяет скидку к каждому платежу.*
date	Это поле содержит дату, когда имел место денежный поток (**value**, второй параметр). Первое значение используется как начальная дата при расчетах смещения для будущих денежных потоков.

Ограничения:

Если имеются текстовые значения, значения NULL и отсутствующие значения в какой-либо или в обеих частях пары значений, такая пара данных будет игнорироваться.

Когда это следует использовать

- XNPV() используется при финансовом моделировании для расчета чистой текущей стоимости (NPV) инвестиционной возможности.
- Благодаря более высокой точности XNPV этой функции отдается предпочтение перед NPV для всех типов финансовых моделей.

Региональные настройки

Если не указано иное, в примерах, приведенных в данном разделе, используется следующий формат даты: ММ/ДД/ГГГГ. Формат даты указан в операторе SET DateFormat скрипта загрузки данных. В вашей системе может быть установлен другой формат даты по умолчанию в зависимости от региональных настроек и других факторов. Можно изменить формат в примерах в соответствии с потребностями. Или можно изменить форматы в скрипте загрузки в соответствии с этими примерами.

Региональные настройки по умолчанию в приложениях основаны на системных региональных настройках компьютера или сервера, где установлено ПО Qlik Sense. Если на сервере Qlik Sense, к которому обращается пользователь, выбран шведский язык, то редактор загрузки данных будет использовать шведские региональные настройки для даты, времени и валюты. Эти параметры регионального формата не связаны с языком, отображаемым в интерфейсе пользователя Qlik Sense. Qlik Sense будет отображаться на языке, который используется в браузере.

Пример 1. Один платеж (скрипт)

Скрипт загрузки и результаты

Обзор

Откройте редактор загрузки данных и добавьте приведенный ниже скрипт загрузки на новую вкладку.

Скрипт загрузки содержит следующее:

- Набор данных одного проекта и его денежного потока за один год в таблице под именем CashFlow. В качестве начальной даты для расчета задано 1 июля 2022 года со значением денежного потока 0. Через год регистрируется денежный поток $1000.
- Резидентная загрузка из таблицы CashFlow, используемая при расчете поля XNPV для проекта в таблице под именем XNPV.
- Жестко запрограммированный процент скидки 10% (0,1), используемый при расчете XNPV.
- Оператор Group By используется для группировки всех платежей в проекте.

Скрипт загрузки

```
CashFlow:
Load
*
Inline
[
PrjId,Dates,Values
```

```
1,'07/01/2022',0
1,'07/01/2023',1000
];

XNPV:
Load
       PrjId,
       XNPV(0.1,Values,Dates) as XNPV //Discount Rate of 10%
Resident CashFlow
Group By PrjId;
```

Результаты

Загрузите данные и откройте лист. Создайте новую таблицу и добавьте эти поля как измерения:

- PrjId
- XNPV

Результирующая таблица

PrjId	XNPV
1	$909.09

Согласно формуле, XNPV для первой записи имеет значение 0, а для второй записи XNPV = $909.09. Таким образом, общая величина XNPV составила $909.09.

Пример 2. Несколько платежей (скрипт)

Скрипт загрузки и результаты

Обзор

Откройте редактор загрузки данных и добавьте приведенный ниже скрипт загрузки на новую вкладку.

Скрипт загрузки содержит следующее:

- Набор данных одного проекта и его денежного потока за один год в таблице под именем CashFlow.
- Резидентная загрузка из таблицы CashFlow, используемая при расчете поля XNPV для проекта в таблице под именем XNPV.
- Жестко запрограммированный процент скидки 10% (0,1), используемый при расчете XNPV.
- Оператор Group By используется для группировки всех платежей в проекте.

Скрипт загрузки

```
CashFlow:
Load
*
```

```
Inline
[
PrjId,Dates,Values
1,'07/01/2022',0
1,'07/01/2024',500
1,'07/01/2023',1000
];

XNPV:
Load
       PrjId,
       XNPV(0.1,Values,Dates) as XNPV //Discount Rate of 10%
Resident CashFlow
Group By PrjId;
```

Результаты

Загрузите данные и откройте лист. Создайте новую таблицу и добавьте эти поля как измерения:

- PrjId
- XNPV

Результирующая таблица

PrjId	XNPV
1	$1322.21

В этом примере платеж $1000 получен в конце первого года, а платеж $500 получен в конце второго года. Так как действует скидка 10% за период, фактическое значение XNPV составляет $1322.21.

Обратите внимание, что только первая строка данных должна ссылаться на базовую дату для расчетов. Для остальных строк порядок не имеет значения, так как параметр даты используется для расчета истекшего периода.

Пример 3. Несколько платежей и нерегулярные денежные потоки (скрипт)

Скрипт загрузки и результаты

Обзор

Откройте редактор загрузки данных и добавьте приведенный ниже скрипт загрузки на новую вкладку.

Скрипт загрузки содержит следующее:

- Процентные ставки скидки для двух проектов в таблице под именем `Project`.
- Денежные потоки за несколько периодов для каждого проекта по идентификаторам проекта и датам. Поле `Dates` используется для расчета продолжительности периода, для которого к денежному потоку применяется скидка. Помимо первой записи (начальный денежный поток и дата), порядок записей не имеет значения, и его изменение не повлияет на расчеты.
- С использованием комбинации `NoConcatenate`, резидентных загрузок и функции `Left Join` создается временная таблица `tmpNPV`, которая объединяет записи таблиц `Project` и `CashFlow` в одну плоскую таблицу. В этой таблице процентные ставки скидки будут повторяться для каждого денежного потока.
- Резидентная загрузка из таблицы `tmpNPV`, используемая при расчете поля `XNPV` для каждого проекта в таблице под именем `XNPV`.
- Одна процентная ставка скидки, связанная с каждым проектом, извлекается с помощью функции `only()` и используется при расчете `XNPV` для каждого проекта.
- Оператор `Group By`, используемый для группировки по идентификатору проекта, применяется с целью группировки всех платежей и соответствующих дат для каждого проекта.
- Для предотвращения загрузки синтетических или избыточных данных в модель данных, таблица `tmpXNPV` удаляется в конце скрипта.

Скрипт загрузки

```
Project:
Load * inline [
PrjId,Discount_Rate
1,0.1
2,0.15
];

CashFlow:
Load
*
Inline
[
PrjId,Dates,Values
1,'07/01/2021',0
1,'07/01/2022',1000
1,'07/01/2023',1000
2,'07/01/2020',0
2,'07/01/2023',500
2,'07/01/2024',1000
2,'07/01/2022',500
];

tmpXNPV:
NoConcatenate Load *
Resident Project;
Left Join
Load *
Resident CashFlow;
```

```
XNPV:
Load
       PrjId,
       XNPV(Only(Discount_Rate),Values,Dates) as XNPV //Discount Rate will be 10% for Project 1 and
15% for Project 2
Resident tmpXNPV
Group By PrjId;

Drop table tmpXNPV;
```

Результаты

Загрузите данные и откройте лист. Создайте новую таблицу и добавьте эти поля как измерения:

- PrjId
- XNPV

Результирующая таблица

PrjId	XNPV
1	$1735.54
2	$278.36

Для идентификатора проекта 1 имеется начальный денежный поток $0 от 1 июля 2021 года. Имеются два платежа $1000 к получению в конце двух последующих лет со скидкой по 10% за период. Таким образом, фактическое значение XNPV составляет $1735.54.

Идентификатор проекта 2 содержит начальный расход $1000 (со знаком «минус») от 1 июля 2020 года. Через два года ожидается платеж $500. Через три года ожидается еще один платеж $500. В заключение 1 июля 2024 года ожидается платеж $1000. Так как действует скидка 10% за период, фактическое значение XNPV составляет $1322.21.

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 По умолчанию результат будет отформатирован как валюта.

Аргументы:

Аргументы

Аргумент	Описание
discount_rate	**discount_rate** is the rate of discount over the length of the period.**discount_rate** – это процентная ставка примененной скидки.
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 **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:

p *NPV – функция диаграммы (page 387)*
p *Aggr – функция диаграммы (page 555)*

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 449){[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:

p *Aggr – функция диаграммы (page 555)*

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:

p *Aggr – функция диаграммы (page 555)*
p *Avg – функция диаграммы (page 407)*
p *RangeCorrel (page 1356)*

Fractile

Функция **Fractile()** находит среди записей, определяемых условием **group by**, значение, соответствующее квантилю агрегированных данных, задаваемых выражением (метод включения).

Используйте функцию FractileExc (page 417) 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 418) 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:

p *Aggr – функция диаграммы (page 555)*

FractileExc

Функция **FractileExc()** находит среди записей, определяемых условием **group by**, значение, соответствующее квантилю агрегированных данных, задаваемых выражением (метод исключения).

Используйте функцию Fractile (page 412) 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 414) 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:

p *Aggr – функция диаграммы (page 555)*

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:

p *Avg – функция диаграммы (page 407)*

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:

p *Примеры использования функций linest (page 467)*

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:

p *Примеры использования функций linest (page 467)*
p *Avg – функция диаграммы (page 407)*

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:

p *Примеры использования функций linest (page 467)*

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:

p *Примеры использования функций linest (page 467)*
p *Avg – функция диаграммы (page 407)*

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:

p *Примеры использования функций linest (page 467)*

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:

p *Примеры использования функций linest (page 467)*
p *Avg – функция диаграммы (page 407)*

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:

p *Примеры использования функций linest (page 467)*

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:

p *Примеры использования функций linest (page 467)*
p *Avg – функция диаграммы (page 407)*

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:

p *Примеры использования функций linest (page 467)*

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:

p *Примеры использования функций linest (page 467)*
p *Avg – функция диаграммы (page 407)*

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:

p *Примеры использования функций linest (page 467)*

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:

p *Примеры использования функций linest (page 467)*
p *Avg – функция диаграммы (page 407)*

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:

p *Примеры использования функций linest (page 467)*

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:

p *Примеры использования функций linest (page 467)*
p *Avg – функция диаграммы (page 407)*

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:

p *Примеры использования функций linest (page 467)*

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:

p *Примеры использования функций linest (page 467)*
p *Avg – функция диаграммы (page 407)*

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:

p *Примеры использования функций linest (page 467)*

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:

p *Примеры использования функций linest (page 467)*
p *Avg – функция диаграммы (page 407)*

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:

p *Примеры использования функций linest (page 467)*

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:

p *Примеры использования функций linest (page 467)*
p *Avg – функция диаграммы (page 407)*

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:

p *Avg – функция диаграммы (page 407)*

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:

p *Avg – функция диаграммы (page 407)*

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:

p *Avg – функция диаграммы (page 407)*
p *STEYX – функция диаграммы (page 465)*

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:

p *Avg – функция диаграммы (page 407)*
p *STEYX – функция диаграммы (page 465)*

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:

p *Avg – функция диаграммы (page 407)*
p *Sterr – функция диаграммы (page 462)*

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

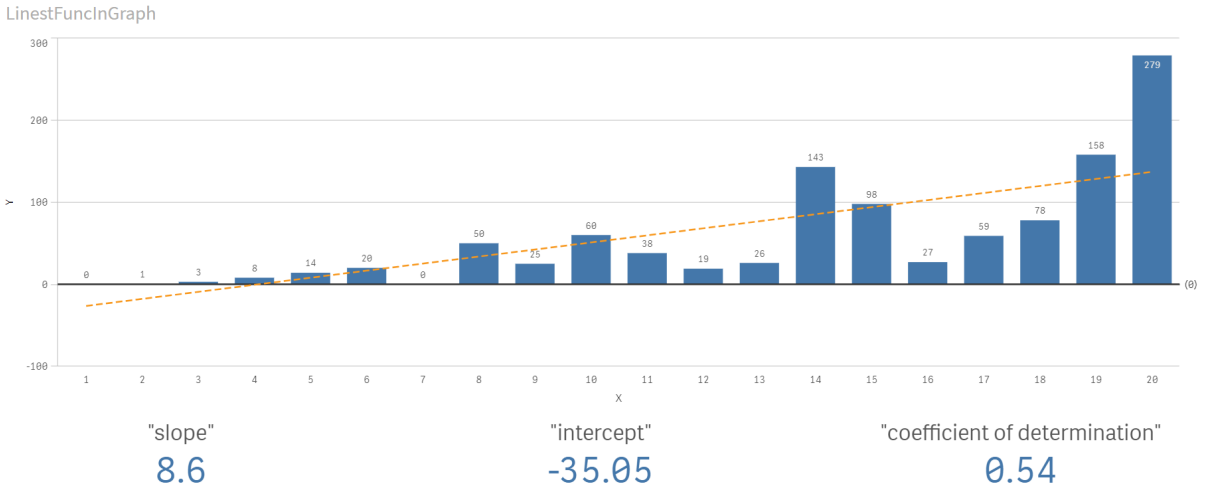
Linest functions	Linest function results
Linest_b	-35.047
Linest_df	18
Linest_f	20.788
Linest_m	8.605
Linest_r2	0.536
Linest_SEB	22.607
Linest_SEM	1.887
Linest_SEY	48.666
Linest_SSREG	49235.014
Linest_SSRESID	42631.186

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:

p *Функции T-критериев (page 474)*
p *Функции Z-критериев (page 511)*

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:

p *Примеры использования функций chi2-test в диаграммах (page 527)*
p *Примеры использования функций chi2-test в скрипте загрузки данных (page 530)*

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:

p *Примеры использования функций chi2-test в диаграммах (page 527)*
p *Примеры использования функций chi2-test в скрипте загрузки данных (page 530)*

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:

p *Примеры использования функций chi2-test в диаграммах (page 527)*
p *Примеры использования функций chi2-test в скрипте загрузки данных (page 530)*

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 532)

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:

p *Создание типичного отчета t-test (page 532)*

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:

p *Создание типичного отчета t-test (page 532)*

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:

p *Создание типичного отчета t-test (page 532)*

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:

p *Создание типичного отчета t-test (page 532)*

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:

p *Создание типичного отчета t-test (page 532)*

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:

p *Создание типичного отчета t-test (page 532)*

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:

p *Создание типичного отчета t-test (page 532)*

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:

p *Создание типичного отчета t-test (page 532)*

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:

p *Создание типичного отчета t-test (page 532)*

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:

p *Создание типичного отчета t-test (page 532)*

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:

p *Создание типичного отчета t-test (page 532)*

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:

p *Создание типичного отчета t-test (page 532)*

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:

p *Создание типичного отчета t-test (page 532)*

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:

p *Создание типичного отчета t-test (page 532)*

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:

p *Создание типичного отчета t-test (page 532)*

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:

p *Создание типичного отчета t-test (page 532)*

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:

p *Создание типичного отчета t-test (page 532)*

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:

p *Создание типичного отчета t-test (page 532)*

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:

p *Создание типичного отчета t-test (page 532)*

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:

p *Создание типичного отчета t-test (page 532)*

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:

p *Создание типичного отчета t-test (page 532)*

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:

p *Создание типичного отчета t-test (page 532)*

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:

p *Создание типичного отчета t-test (page 532)*

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:

p *Создание типичного отчета t-test (page 532)*

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:

p *Создание типичного отчета t-test (page 532)*

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:

p *Создание типичного отчета t-test (page 532)*

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:

p *Создание типичного отчета t-test (page 532)*

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:

p *Создание типичного отчета t-test (page 532)*

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:

p *Создание типичного отчета t-test (page 532)*

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:

p *Создание типичного отчета t-test (page 532)*

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:

p *Создание типичного отчета t-test (page 532)*

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:

p *Создание типичного отчета t-test (page 532)*

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 536)

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:

p *Примеры использования функций z-test (page 536)*

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:

p *Примеры использования функций z-test (page 536)*

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:

p *Примеры использования функций z-test (page 536)*

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:

p *Примеры использования функций z-test (page 536)*

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:

р *Примеры использования функций z-test (page 536)*

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:

p *Примеры использования функций z-test (page 536)*

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:

p *Примеры использования функций z-test (page 536)*

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:

p *Примеры использования функций z-test (page 536)*

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:

p *Примеры использования функций z-test (page 536)*

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:

p *Примеры использования функций z-test (page 536)*

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:

p *Примеры использования функций z-test (page 536)*

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:

p *Примеры использования функций z-test (page 536)*

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:

p *Примеры использования функций z-test (page 536)*

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:

p *Примеры использования функций z-test (page 536)*

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:

p *Aggr – функция диаграммы (page 555)*

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:

p *Базовые функции агрегирования (page 333)*

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:

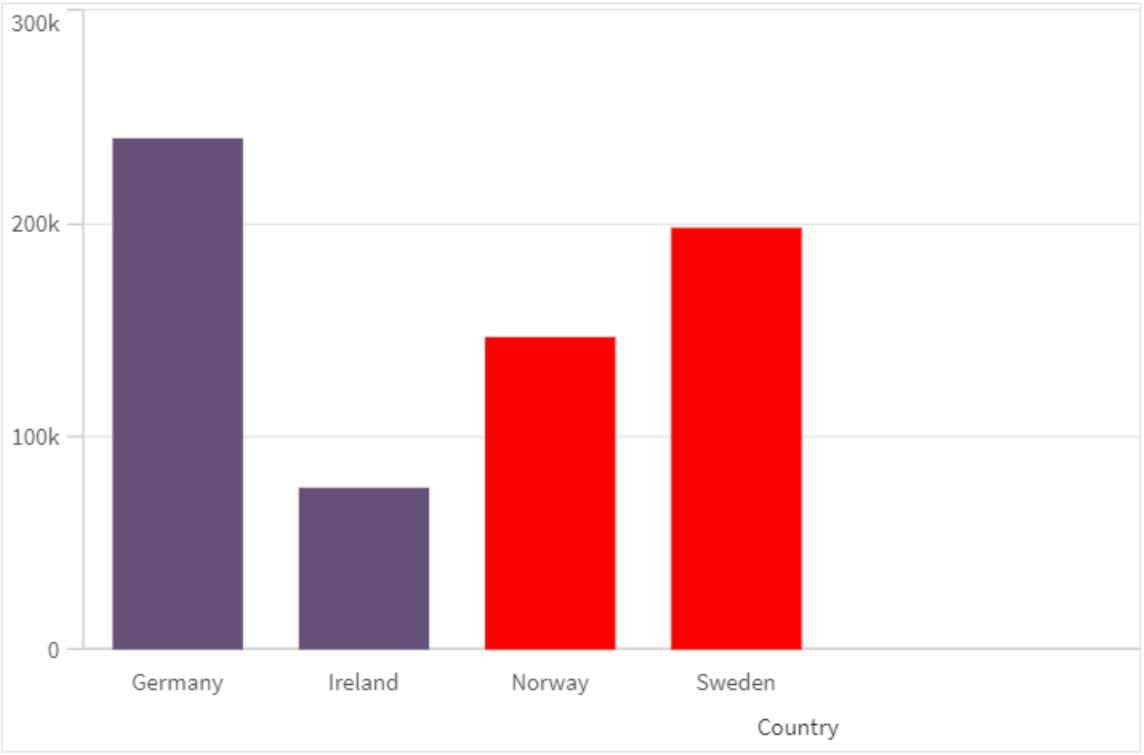
Данные, используемые в этом примере:

```
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])
```

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
Compare transaction_amount field from Transaction table to threshold of 100.
Output results into a new field called Compared to Threshold
 */

Transaction_Buckets:
Load
   transaction_id,
   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

transaction_id	По сравнению с пороговым значением
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
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

Cities
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
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
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
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

Cities	Cities & ' - ' & wildmatch (Cities, 'Tor*','???ton','Beijing','Stockholm','*urich')
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

Region	Year	Month	Sales	RYMkey
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
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

Region	Country	Population	Rank(Population)
Americas	Mexico	128,932,753	2
Americas	Canada	37,742,154	3
Americas	United States of America	331,002,651	1
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

Сортировка по y-значениям в диаграммах или сортировка по столбцам выражений в таблицах не допускается, если в каком-либо из выражений диаграммы используется эта функция диаграммы. Данные возможности сортировки автоматически отключаются. Когда используется эта функция диаграммы в визуализации или таблице, сортировка визуализации сбрасывается до сортированных входных данных этой функции.

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

Customer	UnitSales	Row in Segment	Row Number	If(RowNo( )=1, 0, UnitSales / Above(UnitSales))
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:

p *Above – функция диаграммы (page 1295)*

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. Эта функция возвращает значения в формате системной переменной **TimeStamp**. Значение **timer_mode** по умолчанию – 1.

```
now ([ timer_mode])
```

today

Эта функция возвращает текущую дату. Эта функция возвращает значения в формате системной переменной `DateFormat`.

```
today ([timer_mode])
```

LocalTime

Эта функция возвращает метку текущего времени 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 часовой пояс, определенный на компьютере, на котором работает подсистема Qlik.

```
timezone ( )
```

GMT

Эта функция возвращает текущее значение Greenwich Mean Time согласно региональным настройкам.

```
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 базовой даты в части месяца, двухмесячного периода, квартала, трети года (четыре месяца) или полугодия. Также можно проследить, находится ли метка времени в предыдущем или в 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, трети года (четыре месяца) 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**. При определении лунных недель в Qlik Sense первым днем первой недели считается 1 января. Все недели, кроме последней, будут содержать ровно 7 дней.

```
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. При определении лунных недель в Qlik Sense первым днем первой недели считается 1 января. Все недели, кроме последней будут содержать ровно 7 дней.

```
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, трети года (четыре месяца) или полугодия, содержащих базовую дату. Также можно найти метку времени для предыдущего или последующего временного периода.По умолчанию для вывода используется формат **DateFormat**, установленный в скрипте.

```
monthsstart (n, date [, shift = 0 [, first_month_of_year = 1]])
```

monthsend

Эта функция возвращает значение, соответствующее метке времени последней миллисекунды месяца, двухмесячного периода, квартала, трети года (четыре месяца) 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, трети года (четыре месяца) или полугодия, содержащих базовую дату.

```
monthsname (n, date [, shift = 0 [, first_month_of_year = 1]])
```

weekstart

Эта функция возвращает значение, которое соответствует метке времени, равной первой миллисекунде первого дня календарной недели, и содержит **date**. По умолчанию для вывода используется формат **DateFormat**, заданный в скрипте.

```
weekstart (date [, shift = 0 [,weekoffset = 0]])
```

weekend

Эта функция возвращает значение, которое соответствует метке времени, равной последней миллисекунде последнего дня (воскресенье) календарной недели, и содержит **date**. По умолчанию для вывода используется формат **DateFormat**, заданный 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 первого дня лунной недели, содержащей значение, указанное в поле **date**. При определении лунных недель в Qlik Sense первым днем первой недели считается 1 января. Все недели, кроме последней будут содержать ровно 7 дней.

```
lunarweekstart (date [, shift = 0 [,weekoffset = 0]])
```

lunarweekend

Эта функция возвращает значение, соответствующее метке времени последней миллисекунды последнего дня лунной недели, содержащей значение, указанное в поле **date**. При определении лунных недель в Qlik Sense первым днем первой недели считается 1 января. Все недели, кроме последней будут содержать ровно 7 дней.

```
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**. При определении лунных недель в Qlik Sense первым днем первой недели считается 1 января. Все

недели, кроме последней, будут содержать ровно 7 дней.

```
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. Эта функция используется при создании основного календаря.

```
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:

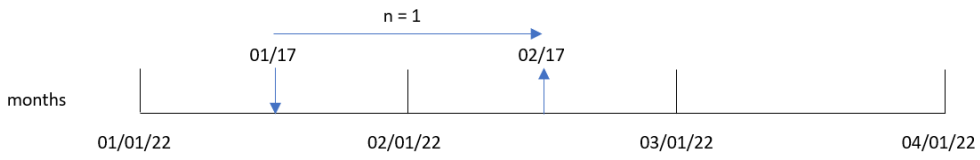
```
AddMonths(startdate, n , [ , mode])
```

Возвращаемые типы данных: двойное значение

Функция `addmonths()` прибавляет или вычитает определенное количество месяцев, `n`, из `startdate` и возвращает полученную дату.

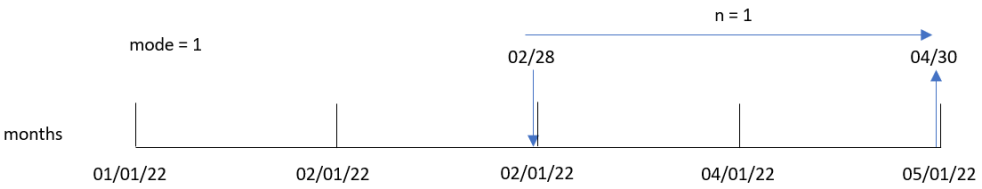
Аргумент `mode` оказывает влияние на значения `startdate` начиная с 28 числа месяца. Задавая аргументу `mode` значение 1, функция `addmonths()` возвращает дату, которая находится на том же относительном расстоянии до конца месяца, что и `startdate`.

Диаграмма примера функции `addmonths()`



Например, 28 февраля является последним днем месяца. Если функция `addmonths()` с `mode` = 1 используется для получения даты через два месяца после этой, функция возвращает последнюю дату апреля – 30 апреля.

Диаграмма с примером функции addmonths() с mode=1



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.

Когда это следует использовать

Функция addmonths() широко используется в составе выражения для поиска даты, которая на заданное количество месяцев отстоит от периода времени в прошлом или будущем.

Например, функцию addmonths() можно использовать для определения даты окончания договоров на обслуживание мобильных телефонов.

Примеры функции

Пример	Результат
addmonths ('01/29/2003' ,3)	Возвращает 04/29/2003.
addmonths ('01/29/2003',3,0)	Возвращает 04/29/2003.
addmonths ('01/29/2003',3,1)	Возвращает 04/28/2003.
addmonths ('01/29/2003',1,0)	Возвращает 02/28/2003.
addmonths ('01/29/2003',1,1)	Возвращает 02/26/2003.
addmonths ('02/28/2003',1,0)	Возвращает 03/28/2003.
addmonths ('02/28/2003',1,1)	Возвращает 03/31/2003.
addmonths ('01/29/2003',-3)	Возвращает 10/29/2002.

Региональные настройки

Если не указано иное, в примерах, приведенных в данном разделе, используется следующий формат даты: MM/ДД/ГГГГ. Формат даты указан в операторе `SET DateFormat` скрипта загрузки данных. В вашей системе может быть установлен другой формат даты по умолчанию в зависимости от региональных настроек и других факторов. Можно изменить формат в примерах в соответствии с потребностями. Или можно изменить форматы в скрипте загрузки в соответствии с этими примерами.

Региональные настройки по умолчанию в приложениях основаны на системных региональных настройках компьютера или сервера, где установлено ПО Qlik Sense. Если на сервере Qlik Sense, к которому обращается пользователь, выбран шведский язык, то редактор загрузки данных будет использовать шведские региональные настройки для даты, времени и валюты. Эти параметры регионального формата не связаны с языком, отображаемым в интерфейсе пользователя Qlik Sense. Qlik Sense будет отображаться на языке, который используется в браузере.

Пример 1. Без дополнительных аргументов

Скрипт загрузки и результаты

Обзор

Откройте редактор загрузки данных и добавьте приведенный ниже скрипт загрузки на новую вкладку.

Скрипт загрузки содержит следующее:

- Набор данных, содержащий набор транзакций с 2020 по 2022 год, загружается в таблицу под именем `Transactions`.
- Поле даты было предоставлено в формате системной переменной `DateFormat` (MM/DD/YYYY).
- Создание поля `two_months_later`, возвращающего дату, которая наступит через два месяца после транзакции.

Скрипт загрузки

```
SET DateFormat='MM/DD/YYYY';

Transactions:
    Load
        *,
        addmonths(date,2) as two_months_later
        ;
Load
*
Inline
[
id,date,amount
8188,'01/10/2020',37.23
8189,'02/28/2020',17.17
```

```
8190,'04/09/2020',88.27
8191,'04/16/2020',57.42
8192,'05/21/2020',53.80
8193,'08/14/2020',82.06
8194,'10/07/2020',40.39
8195,'12/05/2020',87.21
8196,'01/22/2021',95.93
8197,'02/03/2021',45.89
8198,'03/17/2021',36.23
8199,'04/23/2021',25.66
8200,'05/04/2021',82.77
8201,'06/30/2021',69.98
8202,'07/26/2021',76.11
8203,'12/27/2021',25.12
8204,'02/02/2022',46.23
8205,'02/26/2022',84.21
8206,'03/07/2022',96.24
8207,'03/11/2022',67.67
];
```

Результаты

Загрузите данные и откройте лист. Создайте новую таблицу и добавьте эти поля как измерения:

- `date`
- `two_months_later`

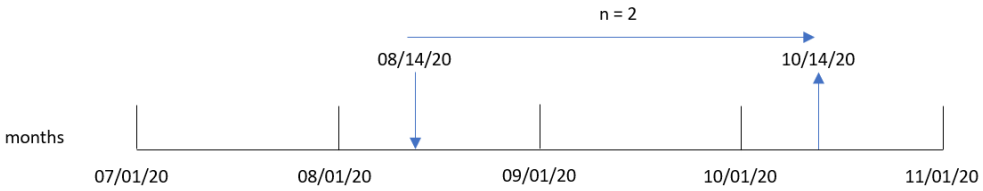
Результирующая таблица

date	two_months_later
01/10/2020	03/10/2020
02/28/2020	04/28/2020
04/09/2020	06/09/2020
04/16/2020	06/16/2020
05/21/2020	07/21/2020
08/14/2020	10/14/2020
10/07/2020	12/07/2020
12/05/2020	02/05/2021
01/22/2021	03/22/2021
02/03/2021	04/03/2021
03/17/2021	05/17/2021
04/23/2021	06/23/2021
05/04/2021	07/04/2021

date	two_months_later
06/30/2021	08/30/2021
07/26/2021	09/26/2021
12/27/2021	02/27/2022
02/02/2022	04/02/2022
02/26/2022	04/26/2022
03/07/2022	05/07/2022
03/11/2022	05/11/2022

Поле `two_months_later` создано в предыдущем операторе load с помощью функции `addmonths()`. Первый предоставленный аргумент определяет, какая дата оценивается. Второй аргумент указывает количество месяцев, которое необходимо прибавить к `startdate` или вычесть из этого значения. В данном примере предоставлено значение 2.

Диаграмма функции `addmonths()`, пример без дополнительных аргументов



Транзакция 8193 выполнена 14 августа. Таким образом, функция `addmonths()` возвращает 14 октября 2020 года для поля `two_months_later`.

Пример 2. Относительный конец месяца

Скрипт загрузки и результаты

Обзор

Откройте редактор загрузки данных и добавьте приведенный ниже скрипт загрузки на новую вкладку.

Скрипт загрузки содержит следующее:

- Набор данных, содержащий набор транзакций конца месяца за 2022 год, загруженный в таблицу под именем `Transactions`.
- Поле даты было предоставлено в формате системной переменной `DateFormat` (MM/DD/YYYY).
- Создание поля `relative_two_months_prior`, возвращающего дату, которая предшествует транзакции на два месяца.

Скрипт загрузки

```
SET DateFormat='MM/DD/YYYY';

Transactions:
    Load
        *,
        addmonths(date,-2,1) as relative_two_months_prior
        ;
Load
*
Inline
[
id,date,amount
8188,'01/28/2022',37.23
8189,'01/31/2022',57.54
8190,'02/28/2022',17.17
8191,'04/29/2022',88.27
8192,'04/30/2022',57.42
8193,'05/31/2022',53.80
8194,'08/14/2022',82.06
8195,'10/07/2022',40.39
];
```

Результаты

Загрузите данные и откройте лист. Создайте новую таблицу и добавьте эти поля как измерения:

- `date`
- `relative_two_months_prior`

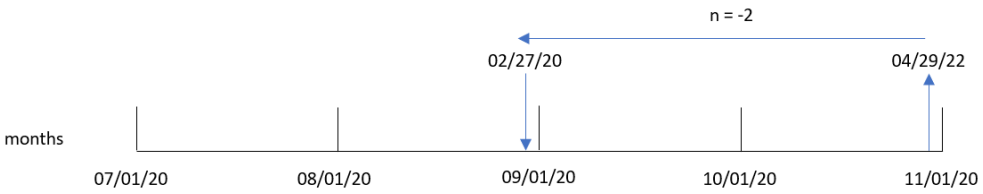
Результирующая таблица

date	relative_two_months_prior
01/28/2022	11/27/2021
01/31/2022	11/30/2021
02/28/2022	12/31/2021
04/29/2022	02/27/2022
04/30/2022	02/28/2022
05/31/2022	03/31/2022
08/14/2022	06/14/2022
10/07/2022	08/07/2022

Поле `relative_two_months_prior` создано в предыдущем операторе load с помощью функции `addmonths()`. Первый предоставленный аргумент определяет, какая дата оценивается. Второй аргумент указывает количество месяцев, которое необходимо прибавить к `startdate` или вычесть

из этого значения. В данном примере предоставлено значение -2. В качестве последнего аргумента задан mode = 1, который принуждает функцию вычислять дату относительного окончания месяца для всех дат начиная с 28 числа месяца.

Диаграмма функции addmonths() с n=-2



Транзакция 8199 совершена 29 апреля 2022 года. Первоначально перенос на два месяца назад дает дату в феврале. Затем функция вычисляет значение относительного конца месяца, так как используется третий аргумент функции, который задает mode = 1, и число месяца после 27. Функция определяет, что 29-е число является предпоследним днем апреля, и поэтому возвращает предпоследний день февраля, то есть 27-е число.

Пример 3. Пример объекта диаграммы

Скрипт загрузки и выражение диаграммы

Обзор

Откройте редактор загрузки данных и добавьте приведенный ниже скрипт загрузки на новую вкладку.

Скрипт загрузки содержит тот же набор данных и сценарий, что в первом примере.

Однако в этом примере в приложение загружается неизмененный набор данных. В качестве меры в объекте диаграммы создается вычисление, возвращающее дату, которая наступит через два месяца после транзакции.

Скрипт загрузки

```
SET DateFormat='MM/DD/YYYY';

Transactions:
Load
*
Inline
[
id,date,amount
8188,'01/10/2020',37.23
8189,'02/28/2020',17.17
8190,'04/09/2020',88.27
8191,'04/16/2020',57.42
8192,'05/21/2020',53.80
8193,'08/14/2020',82.06
```

```
8194,'10/07/2020',40.39
8195,'12/05/2020',87.21
8196,'01/22/2021',95.93
8197,'02/03/2021',45.89
8198,'03/17/2021',36.23
8199,'04/23/2021',25.66
8200,'05/04/2021',82.77
8201,'06/30/2021',69.98
8202,'07/26/2021',76.11
8203,'12/27/2021',25.12
8204,'02/02/2022',46.23
8205,'02/26/2022',84.21
8206,'03/07/2022',96.24
8207,'03/11/2022',67.67
];
```

Результаты

Загрузите данные и откройте лист. Создайте новую таблицу и добавьте это поле как измерение: `date`.

Создайте следующую меру:

```
=addmonths(date,2)
```

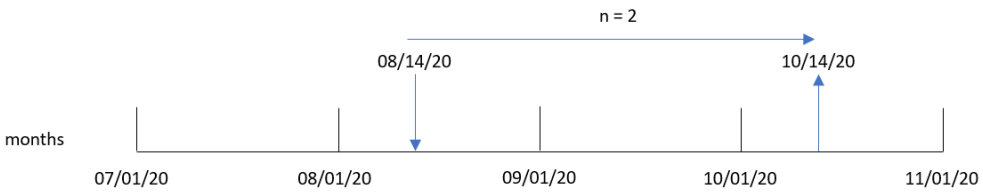
Результирующая таблица

date	=addmonths(date,2)
01/10/2020	03/10/2020
02/28/2020	04/28/2020
04/09/2020	06/09/2020
04/16/2020	06/16/2020
05/21/2020	07/21/2020
08/14/2020	10/14/2020
10/07/2020	12/07/2020
12/05/2020	02/05/2021
01/22/2021	03/22/2021
02/03/2021	04/03/2021
03/17/2021	05/17/2021
04/23/2021	06/23/2021
05/04/2021	07/04/2021
06/30/2021	08/30/2021
07/26/2021	09/26/2021

date	=addmonths(date,2)
12/27/2021	02/27/2022
02/02/2022	04/02/2022
02/26/2022	04/26/2022
03/07/2022	05/07/2022
03/11/2022	05/11/2022

Мера `two_months_later` создается в объекте диаграммы с помощью функции `addmonths()`. Первый предоставленный аргумент определяет, какая дата оценивается. Второй аргумент указывает количество месяцев, которое необходимо прибавить к `startdate` или вычесть из этого значения. В данном примере предоставлено значение 2.

Диаграмма функции `addmonths()`, пример с объектом диаграммы



Транзакция 8193 совершена 14 августа. Поэтому функция `addmonths()` возвращает 14 октября 2020 года для поля `two_months_later`.

Пример 4. Сценарий

Скрипт загрузки и выражение диаграммы

Обзор

Откройте редактор загрузки данных и добавьте приведенный ниже скрипт загрузки на новую вкладку.

Скрипт загрузки содержит следующее:

- Набор данных, загруженный в таблицу, называется `Mobile_Plans`
- Информация содержит ИД договора, дату начала, срок действия договора и ежемесячную плату.

Конечному пользователю требуется объект диаграммы, который показывает дату окончания договора для каждого телефона по ИД договора.

Скрипт загрузки

```
Mobile_Plans:
Load
*
```

```
Inline
[
contract_id,start_date,contract_length,monthly_fee
8188,'01/13/2020',18,37.23
8189,'02/26/2020',24,17.17
8190,'03/27/2020',36,88.27
8191,'04/16/2020',24,57.42
8192,'05/21/2020',24,53.80
8193,'08/14/2020',12,82.06
8194,'10/07/2020',18,40.39
8195,'12/05/2020',12,87.21
8196,'01/22/2021',12,95.93
8197,'02/03/2021',18,45.89
8198,'03/17/2021',24,36.23
8199,'04/23/2021',24,25.66
8200,'05/04/2021',12,82.77
8201,'06/30/2021',12,69.98
8202,'07/26/2021',12,76.11
8203,'12/27/2021',36,25.12
8204,'06/06/2022',24,46.23
8205,'07/18/2022',12,84.21
8206,'11/14/2022',12,96.24
8207,'12/12/2022',18,67.67
];
```

Результаты

Загрузите данные и откройте лист. Создайте новую таблицу и добавьте эти поля как измерения:

- `contract_id`
- `start_date`
- `contract_length`

Создайте следующую меру для расчета даты окончания действия каждого договора:

```
=addmonths(start_date,contract_length, 0)
```

Результирующая таблица

contract_id	start_date	contract_length	=addmonths(start_date,contract_length,0)
8188	01/13/2020	18	07/13/2021
8189	02/26/2020	24	02/26/2022
8190	03/27/2020	36	03/27/2023
8191	04/16/2020	24	04/16/2022
8192	05/21/2020	24	05/21/2022
8193	08/14/2020	12	08/14/2021
8194	10/07/2020	18	04/07/2022

contract_id	start_date	contract_length	=addmonths(start_date,contract_length,0)
8195	12/05/2020	12	12/05/2021
8196	01/22/2021	12	01/22/2022
8197	02/03/2021	18	08/03/2022
8198	03/17/2021	24	03/17/2023
8199	04/23/2021	24	04/23/2023
8200	05/04/2021	12	05/04/2022
8201	06/30/2021	12	06/30/2022
8202	07/26/2021	12	07/26/2022
8203	12/27/2021	36	12/27/2024
8204	06/06/2022	24	06/06/2024
8205	07/18/2022	12	07/18/2023
8206	11/14/2022	12	11/14/2023
8207	12/12/2022	18	06/12/2024

addyears

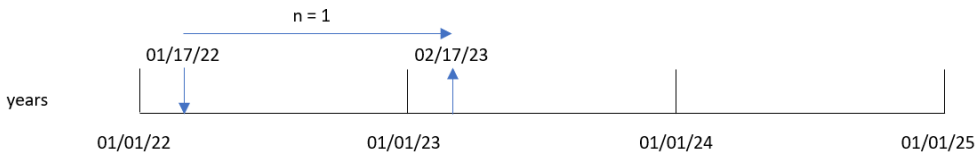
Эта функция возвращает дату через **n** лет после даты начала **startdate** или, если **n** является отрицательным числом, – дату за **n** лет до даты начала **startdate**.

Синтаксис:

```
AddYears(startdate, n)
```

Возвращаемые типы данных: двойное значение

Диаграмма с примером функции addyears()



Функция `addyears()` прибавляет или вычитает определенное количество лет, `n`, из `startdate`. Затем она возвращает полученную дату.

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 функции

Пример	Результат
`addyears ('01/29/2010',3)`	Возвращает 01/29/2013.
`addyears ('01/29/2010',-1)`	Возвращает 01/29/2009.

Региональные настройки

Если не указано иное, в примерах, приведенных в данном разделе, используется следующий формат даты: ММ/ДД/ГГГГ. Формат даты указан в операторе `SET DateFormat` скрипта загрузки данных. В вашей системе может быть установлен другой формат даты по умолчанию в зависимости от региональных настроек и других факторов. Можно изменить формат в примерах в соответствии с потребностями. Или можно изменить форматы в скрипте загрузки в соответствии с этими примерами.

Региональные настройки по умолчанию в приложениях основаны на системных региональных настройках компьютера или сервера, где установлено ПО Qlik Sense. Если на сервере Qlik Sense, к которому обращается пользователь, выбран шведский язык, то редактор загрузки данных будет использовать шведские региональные настройки для даты, времени и валюты. Эти параметры регионального формата не связаны с языком, отображаемым в интерфейсе пользователя Qlik Sense. Qlik Sense будет отображаться на языке, который используется в браузере.

Пример 1. Простой пример

Скрипт загрузки и результаты

Обзор

Откройте редактор загрузки данных и добавьте приведенный ниже скрипт загрузки на новую вкладку.

Скрипт загрузки содержит следующее:

- Набор данных, содержащий набор транзакций с 2020 по 2022 год, загружается в таблицу под именем `Transactions`.
- Поле даты было предоставлено в формате системной переменной `DateFormat` (MM/DD/YYYY).
- Создание поля `two_years_later`, возвращающего дату, которая наступит через два года после транзакции.

Скрипт загрузки

```
SET DateFormat='MM/DD/YYYY';

Transactions:
    Load
        *,
        addyears(date,2) as two_years_later
        ;
```

```
Load
*
Inline
[
id,date,amount
8188,'01/10/2020',37.23
8189,'02/28/2020',17.17
8190,'04/09/2020',88.27
8191,'04/16/2020',57.42
8192,'05/21/2020',53.80
8193,'08/14/2020',82.06
8194,'10/07/2020',40.39
8195,'12/05/2020',87.21
8196,'01/22/2021',95.93
8197,'02/03/2021',45.89
8198,'03/17/2021',36.23
8199,'04/23/2021',25.66
8200,'05/04/2021',82.77
8201,'06/30/2021',69.98
8202,'07/26/2021',76.11
8203,'12/27/2021',25.12
8204,'02/02/2022',46.23
8205,'02/26/2022',84.21
8206,'03/07/2022',96.24
8207,'03/11/2022',67.67
];
```

Результаты

Загрузите данные и откройте лист. Создайте новую таблицу и добавьте эти поля как измерения:

- `date`
- `two_years_later`

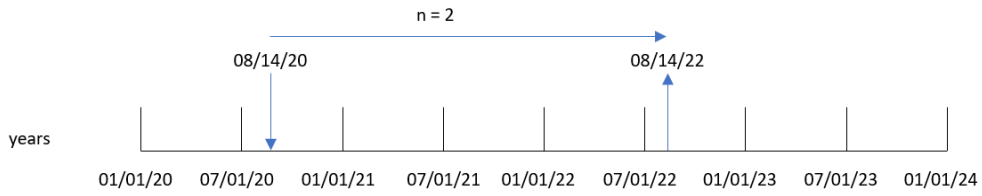
Результирующая таблица

date	two_years_later
01/10/2020	01/10/2022
02/28/2020	02/28/2022
04/09/2020	04/09/2022
04/16/2020	04/16/2022
05/21/2020	05/21/2022
08/14/2020	08/14/2022
10/07/2020	10/07/2022
12/05/2020	12/05/2022
01/22/2021	01/22/2023

date	two_years_later
02/03/2021	02/03/2023
03/17/2021	03/17/2023
04/23/2021	04/23/2023
05/04/2021	05/04/2023
06/30/2021	06/30/2023
07/26/2021	07/26/2023
12/27/2021	12/27/2023
02/02/2022	02/02/2024
02/26/2022	02/26/2024
03/07/2022	03/07/2024
03/11/2022	03/11/2024

Поле `two_years_later` создано в предыдущем операторе load с помощью функции `addyears()`. Первый предоставленный аргумент определяет, какая дата оценивается. Второй аргумент указывает количество лет, которое необходимо прибавить к дате начала или вычесть из этого значения. В данном примере предоставлено значение 2.

Диаграмма функции `addyears()`, базовый пример



Транзакция 8193 совершена 14 августа 2020 года. Поэтому функция `addyears()` возвращает 14 августа 2022 года для поля `two_years_later`.

Пример 2. Пример объекта диаграммы

Скрипт загрузки и выражение диаграммы

Обзор

Откройте редактор загрузки данных и добавьте приведенный ниже скрипт загрузки на новую вкладку.

Скрипт загрузки содержит следующее:

- Набор данных, содержащий набор транзакций с 2020 по 2022 год, загружается в таблицу под именем `Transactions`.
- Поле даты было предоставлено в формате системной переменной `DateFormat` (MM/DD/YYYY).

В объекте диаграммы создайте меру `prior_year_date`, которая возвращает дату за один год до транзакции.

Скрипт загрузки

```
SET DateFormat='MM/DD/YYYY';

Transactions:
Load
*
Inline
[
id,date,amount
8188,'01/10/2020',37.23
8189,'02/28/2020',17.17
8190,'04/09/2020',88.27
8191,'04/16/2020',57.42
8192,'05/21/2020',53.80
8193,'08/14/2020',82.06
8194,'10/07/2020',40.39
8195,'12/05/2020',87.21
8196,'01/22/2021',95.93
8197,'02/03/2021',45.89
8198,'03/17/2021',36.23
8199,'04/23/2021',25.66
8200,'05/04/2021',82.77
8201,'06/30/2021',69.98
8202,'07/26/2021',76.11
8203,'12/27/2021',25.12
8204,'02/02/2022',46.23
8205,'02/26/2022',84.21
8206,'03/07/2022',96.24
8207,'03/11/2022',67.67
];
```

Результаты

Загрузите данные и откройте лист. Создайте новую таблицу и добавьте это поле как измерение: `date`.

Создайте следующую меру для расчета даты за год до каждой транзакции:

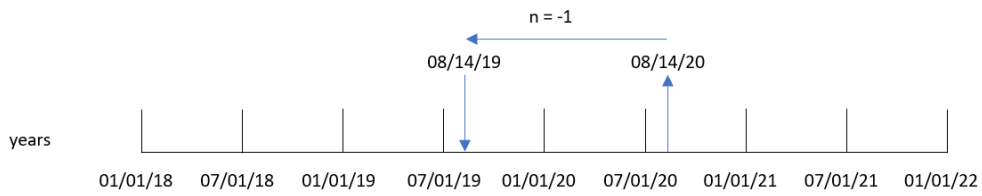
```
=addyears(date,-1)
```

Результирующая таблица

date	=addyears(date,-1)
01/10/2020	01/10/2019
02/28/2020	02/28/2019
04/09/2020	04/09/2019
04/16/2020	04/16/2019
05/21/2020	05/21/2019
08/14/2020	08/14/2019
10/07/2020	10/07/2019
12/05/2020	12/05/2019
01/22/2021	01/22/2020
02/03/2021	02/03/2020
03/17/2021	03/17/2020
04/23/2021	04/23/2020
05/04/2021	05/04/2020
06/30/2021	06/30/2020
07/26/2021	07/26/2020
12/27/2021	12/27/2020
02/02/2022	02/02/2021
02/26/2022	02/26/2021
03/07/2022	03/07/2021
03/11/2022	03/11/2021

Мера `one_year_prior` создается в объекте диаграммы с помощью функции `addyears()`. Первый предоставленный аргумент определяет, какая дата оценивается. Второй аргумент указывает количество лет, которое необходимо прибавить к `startdate` или вычесть из этого значения. В данном примере предоставлено значение -1.

Диаграмма функции `addyears()`, пример с объектом диаграммы



Транзакция 8193 совершена 14 августа. Поэтому функция `addyears()` возвращает 14 августа 2019 года для поля `one_year_prior`.

Пример 3. Сценарий

Скрипт загрузки и выражение диаграммы

Обзор

Откройте редактор загрузки данных и добавьте приведенный ниже скрипт загрузки на новую вкладку.

Скрипт загрузки содержит следующее:

- Набор данных, загруженный в таблицу, называется `Warranties`
- Информация содержит ИД продукта, дату приобретения, гарантийный срок и цену покупки.

Конечному пользователю требуется объект диаграммы, который показывает дату окончания действия гарантии для каждого продукта по ИД продукта.

Скрипт загрузки

```
Warranties:
Load
*
Inline
[
product_id,purchase_date,warranty_length,purchase_price
8188,'01/13/2020',4,32000
8189,'02/26/2020',2,28000
8190,'03/27/2020',3,41000
8191,'04/16/2020',4,17000
8192,'05/21/2020',2,25000
8193,'08/14/2020',1,59000
8194,'10/07/2020',2,12000
8195,'12/05/2020',3,12000
8196,'01/22/2021',4,24000
8197,'02/03/2021',1,50000
8198,'03/17/2021',2,80000
8199,'04/23/2021',3,10000
8200,'05/04/2021',4,30000
8201,'06/30/2021',3,30000
8202,'07/26/2021',4,20000
8203,'12/27/2021',4,10000
8204,'06/06/2022',2,25000
8205,'07/18/2022',1,32000
8206,'11/14/2022',1,30000
8207,'12/12/2022',4,22000
];
```

Результаты

Загрузите данные и откройте лист. Создайте новую таблицу и добавьте эти поля как измерения:

- product_id
- purchase_date
- warranty_length

Создайте следующую меру для расчета даты окончания действия гарантии на каждый продукт:

```
=addyears(purchase_date,warranty_length)
```

Результирующая таблица

product_id	purchase_date	warranty_length	=addyears(purchase_date,warranty_length)
8188	01/13/2020	4	01/13/2024
8189	02/26/2020	2	02/26/2022
8190	03/27/2020	3	03/27/2023
8191	04/16/2020	4	04/16/2024
8192	05/21/2020	2	05/21/2022
8193	08/14/2020	1	08/14/2021
8194	10/07/2020	2	10/07/2022
8195	12/05/2020	3	12/05/2023
8196	01/22/2021	4	01/22/2025
8197	02/03/2021	1	02/03/2022
8198	03/17/2021	2	03/17/2023
8199	04/23/2021	3	04/23/2024
8200	05/04/2021	4	05/04/2025
8201	06/30/2021	3	06/30/2024
8202	07/26/2021	4	07/26/2025
8203	12/27/2021	4	12/27/2025
8204	06/06/2022	2	06/06/2024
8205	07/18/2022	1	07/18/2023
8206	11/14/2022	1	11/14/2023
8207	12/12/2022	4	12/12/2026

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
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.
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

A-C	D-K	L-R	S-Z
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
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	-

A-C	D-K	L-R	S-Z
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

Пример 1. Набор данных DateFormat (скрипт)

Скрипт загрузки и результаты

Обзор

Откройте Редактор загрузки данных и добавьте приведенный ниже скрипт загрузки на новую вкладку.

Скрипт загрузки содержит следующее:

- Набор данных с датами под именем `Master_Calendar`. Системной переменной `DateFormat` задано значение DD/MM/YYYY.
- Предшествующая загрузка, создающая дополнительное поле под именем `day_of_month` с использованием функции `day()`.
- Дополнительное поле под именем `long_date` с использованием функции `date()` для выражения полного названия месяца.

Скрипт загрузки

```
SET DateFormat='DD/MM/YYYY';

Master_Calendar:
Load
        date,
        date(date,'dd-MMMM-YYYY') as long_date,
        day(date) as day_of_month
Inline
[
date
03/11/2022
03/12/2022
03/13/2022
03/14/2022
03/15/2022
03/16/2022
03/17/2022
03/18/2022
03/19/2022
03/20/2022
03/21/2022
];
```

Результаты

Загрузите данные и откройте лист. Создайте новую таблицу и добавьте эти поля как измерения:

- `date`
- `long_date`
- `day_of_month`

Результирующая таблица

date	long_date	day_of_month
03/11/2022	11-March- 2022	11
03/12/2022	12-March- 2022	12
03/13/2022	13-March- 2022	13
03/14/2022	14-March- 2022	14
03/15/2022	15-March- 2022	15
03/16/2022	16-March- 2022	16
03/17/2022	17-March- 2022	17
03/18/2022	18-March- 2022	18
03/19/2022	19-March- 2022	19
03/20/2022	20-March- 2022	20
03/21/2022	21-March- 2022	21

День месяца правильно вычисляется функцией `day()` в скрипте.

Пример 2. Даты ANSI (скрипт)

Скрипт загрузки и результаты

Обзор

Откройте Редактор загрузки данных и добавьте приведенный ниже скрипт загрузки на новую вкладку.

Скрипт загрузки содержит следующее:

- Набор данных с датами под именем `Master_Calendar`. Используется системная переменная DateFormat DD/MM/YYYY. Однако даты, включенные в набор данных, имеют формат стандарта ANSI.
- Предшествующая загрузка, создающая дополнительное поле под именем `day_of_month` с использованием функции `date()`.
- Дополнительное поле под именем `long_date` с использованием функции `date()` для выражения даты с использованием полного названия месяца.

Скрипт загрузки

```
SET DateFormat='DD/MM/YYYY';
Master_Calendar:
Load
       date,
       date(date,'dd-MMMM-YYYY') as long_date,
       day(date) as day_of_month
```

```
Inline
[
date
2022-03-11
2022-03-12
2022-03-13
2022-03-14
2022-03-15
2022-03-16
2022-03-17
2022-03-18
2022-03-19
2022-03-20
2022-03-21
 ];
```

Результаты

Загрузите данные и откройте лист. Создайте новую таблицу и добавьте эти поля как измерения:

- `date`
- `long_date`
- `day_of_month`

Результирующая таблица

date	long_date	day_of_month
03/11/2022	11-March- 2022	11
03/12/2022	12-March- 2022	12
03/13/2022	13-March- 2022	13
03/14/2022	14-March- 2022	14
03/15/2022	15-March- 2022	15
03/16/2022	16-March- 2022	16
03/17/2022	17-March- 2022	17
03/18/2022	18-March- 2022	18
03/19/2022	19-March- 2022	19
03/20/2022	20-March- 2022	20
03/21/2022	21-March- 2022	21

День месяца правильно вычисляется функцией `day()` в скрипте.

Пример 3. Неформатированные даты (скрипт)

Скрипт загрузки и результаты

Обзор

Откройте Редактор загрузки данных и добавьте приведенный ниже скрипт загрузки на новую вкладку.

Скрипт загрузки содержит следующее:

- Набор данных с датами под именем `Master_Calendar`. Используется системная переменная `DateFormat` DD/MM/YYYY.
- Предшествующая загрузка, создающая дополнительное поле под именем `day_of_month` с использованием функции `day()`.
- Первоначальная дата без форматирования под именем `unformatted_date`.
- Дополнительное поле под именем `long_date` с использованием `date()` служит для преобразования цифровой даты в поле форматированной даты.

Скрипт загрузки

```
SET DateFormat='DD/MM/YYYY';

Master_Calendar:
Load
      unformatted_date,
      date(unformatted_date,'dd-MMMM-YYYY') as long_date,
      day(date) as day_of_month

Inline
[
unformatted_date
44868
44898
44928
44958
44988
45018
45048
45078
45008
45038
45068
];
```

Результаты

Загрузите данные и откройте лист. Создайте новую таблицу и добавьте эти поля как измерения:

- unformatted_date
- long_date
- day_of_month

Результирующая таблица

unformatted_date	long_date	day_of_month
44868	03-November- 2022	3
44898	03-December- 2022	3
44928	02-January- 2023	2
44958	01-February- 2023	1
44988	03-March- 2023	3
45008	23-March- 2023	23
45018	02-April- 2023	2
45038	22-April- 2023	22
45048	02-May- 2023	2
45068	22-May- 2023	22
45078	01-June- 2023	1

День месяца правильно вычисляется функцией day() в скрипте.

Пример 4. Расчет месяца окончания срока действия (диаграмма)

Скрипт загрузки и выражение диаграммы

Обзор

Откройте Редактор загрузки данных и добавьте приведенный ниже скрипт загрузки на новую вкладку.

Скрипт загрузки содержит следующее:

- Набор данных с заказами, размещенными в марте, под именем Orders. Данная таблица содержит слишком много полей.
    - id
    - order_date
    - amount

Скрипт загрузки

```
Orders:
Load
        id,
```

```
        order_date,
        amount
Inline
[
id,order_date,amount
1,03/01/2022,231.24
2,03/02/2022,567.28
3,03/03/2022,364.28
4,03/04/2022,575.76
5,03/05/2022,638.68
6,03/06/2022,785.38
7,03/07/2022,967.46
8,03/08/2022,287.67
9,03/09/2022,764.45
10,03/10/2022,875.43
11,03/11/2022,957.35
];
```

Результаты

Загрузите данные и откройте лист. Создайте новую таблицу и добавьте это поле как измерение: `order_date`.

Чтобы рассчитать дату доставки, создайте эту меру: `=day(order_date+5)`.

Результирующая таблица

order_date	=day(order_date+5)
03/11/2022	16
03/12/2022	17
03/13/2022	18
03/14/2022	19
03/15/2022	20
03/16/2022	21
03/17/2022	22
03/18/2022	23
03/19/2022	24
03/20/2022	25
03/21/2022	26

Функция `day()` правильно определяет, что заказ, размещенный 11 марта, будет доставлен 16 марта с учетом 5-дневного периода доставки.

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]])
```

Когда это следует использовать

Функция `dayend()` широко используется в составе выражения, когда пользователь хочет использовать в расчетах часть дня, которая еще не прошла, например, для расчета общих расходов, которые будут иметь место в течение дня.

Возвращаемые типы данных: двойное значение

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, что дни начинаются не в полночь, укажите смещение в виде десятичного значения в параметре **day_start**. Например, 0,125 обозначает 3:00. Другими словами, чтобы создать смещение, разделите время начала на 24 часа. Например, чтобы день начинался в 7:00, используйте дробь 7/24.

Региональные настройки

Если не указано иное, в примерах, приведенных в данном разделе, используется следующий формат даты: ММ/ДД/ГГГГ. Формат даты указан в операторе `SET DateFormat` скрипта загрузки данных. В вашей системе может быть установлен другой формат даты по умолчанию в зависимости от региональных настроек и других факторов. Можно изменить формат в примерах в соответствии с потребностями. Или можно изменить форматы в скрипте загрузки в соответствии с этими примерами.

Региональные настройки по умолчанию в приложениях основаны на системных региональных настройках компьютера или сервера, где установлено ПО Qlik Sense. Если на сервере Qlik Sense, к которому обращается пользователь, выбран шведский язык, то редактор загрузки данных будет использовать шведские региональные настройки для даты, времени и валюты. Эти параметры регионального формата не связаны с языком, отображаемым в интерфейсе пользователя Qlik Sense. Qlik Sense будет отображаться на языке, который используется в браузере.

Примеры функции

Пример	Результат
dayend('01/25/2013 16:45:00')	Returns 01/25/2013 23:59:59. после полудня
dayend('01/25/2013 16:45:00', -1)	Возвращает 01/24/2013 23:59:59. после полудня
dayend('01/25/2013 16:45:00', 0, 0.5)	Returns 01/26/2013 11:59:59. после полудня

Пример 1. Базовый скрипт

Скрипт загрузки и результаты

Обзор

Откройте Редактор загрузки данных и добавьте приведенный ниже скрипт загрузки на новую вкладку.

Скрипт загрузки содержит следующее:

- Набор данных, содержащий список дат и загруженный в таблицу под именем Calendar.
- Переменная `DateFormat` системы `(MM/DD/YYYY)` по умолчанию.
- Предшествующая загрузка для создания дополнительного поля под именем `EOD_timestamp` с использованием функции `dayend()`.

Скрипт загрузки

```
SET TimestampFormat='M/D/YYYY h:mm:ss[.fff] TT';

Calendar:
    Load
      date,
      dayend(date) as EOD_timestamp
      ;
Load
date
Inline
[
date
03/11/2022 1:47:15 AM
03/12/2022 4:34:58 AM
03/13/2022 5:15:55 AM
03/14/2022 9:25:14 AM
03/15/2022 10:06:54 AM
03/16/2022 10:44:42 AM
03/17/2022 11:33:30 AM
03/18/2022 12:58:14 PM
03/19/2022 4:23:12 PM
03/20/2022 6:42:15 PM
03/21/2022 7:41:16 PM
];
```

Результаты

Загрузите данные и откройте лист. Создайте новую таблицу и добавьте эти поля как измерения:

- `date`
- `EOD_timestamp`

Результирующая таблица

date	EOD_timestamp
03/11/2022 1:47:15 AM	3/11/2022 11:59:59 PM
03/12/2022 4:34:58 AM	3/12/2022 11:59:59 PM
03/13/2022 5:15:55 AM	3/13/2022 11:59:59 PM
03/14/2022 9:25:14 AM	3/14/2022 11:59:59 PM
03/15/2022 10:06:54 AM	3/15/2022 11:59:59 PM
03/16/2022 10:44:42 AM	3/16/2022 11:59:59 PM
03/17/2022 11:33:30 AM	3/17/2022 11:59:59 PM
03/18/2022 12:58:14 PM	3/18/2022 11:59:59 PM
03/19/2022 4:23:12 PM	3/19/2022 11:59:59 PM
03/20/2022 6:42:15 PM	3/20/2022 11:59:59 PM
03/21/2022 7:41:16 PM	3/21/2022 11:59:59 PM

Как показано в приведенной выше таблице, отметка времени окончания дня генерируется для каждой даты в наборе данных. Для отметки времени используется формат системной переменной `TimestampFormat M/D/YYYY h:mm:ss[.fff] TT`.

Пример 2. Скрипт period_no

Скрипт загрузки и результаты

Обзор

Откройте Редактор загрузки данных и добавьте приведенный ниже скрипт загрузки на новую вкладку.

Вы загрузите набор данных, содержащий бронирования службы, в таблицу под именем Services.

Набор данных включает следующие поля:

- `service_id`
- `service_date`
- `amount`

Вы создадите два новых поля в таблице:

- deposit_due_date: дата, когда должен быть получен авансовый платеж. Это окончание дня за три дня до service_date.
- final_payment_due_date: дата, когда должен быть получен окончательный платеж. Это окончание дня через семь дней после service_date.

Два упомянутые выше поля созданы при предыдущей загрузке с использованием функции dayend () и предоставляют два первых параметра, time и period_no.

Скрипт загрузки

```
SET TimestampFormat='M/D/YYYY h:mm:ss[.fff] TT';

Services:
   Load
     *,
     dayend(service_date,-3) as deposit_due_date,
     dayend(service_date,7) as final_payment_due_date
     ;
Load
service_id,
service_date,
amount
Inline
[
service_id, service_date,amount
1,03/11/2022 9:25:14 AM,231.24
2,03/12/2022 10:06:54 AM,567.28
3,03/13/2022 10:44:42 AM,364.28
4,03/14/2022 11:33:30 AM,575.76
5,03/15/2022 12:58:14 PM,638.68
6,03/16/2022 4:23:12 PM,785.38
7,03/17/2022 6:42:15 PM,967.46
8,03/18/2022 7:41:16 PM,287.67
9,03/19/2022 8:14:15 PM,764.45
10,03/20/2022 9:23:51 PM,875.43
11,03/21/2022 10:04:41 PM,957.35
];
```

Результаты

Загрузите данные и откройте лист. Создайте новую таблицу и добавьте эти поля как измерения:

- service_date
- deposit_due_date
- final_payment_due_date

Результирующая таблица

service_date	deposit_due_date	final_payment_due_date
03/11/2022 9:25:14 AM	3/8/2022 11:59:59 PM	3/18/2022 11:59:59 PM

service_date	deposit_due_date	final_payment_due_date
03/12/2022 10:06:54 AM	3/9/2022 11:59:59 PM	3/19/2022 11:59:59 PM
03/13/2022 10:44:42 AM	3/10/2022 11:59:59 PM	3/20/2022 11:59:59 PM
03/14/2022 11:33:30 AM	3/11/2022 11:59:59 PM	3/21/2022 11:59:59 PM
03/15/2022 12:58:14 PM	3/12/2022 11:59:59 PM	3/22/2022 11:59:59 PM
03/16/2022 4:23:12 PM	3/13/2022 11:59:59 PM	3/23/2022 11:59:59 PM
03/17/2022 6:42:15 PM	3/14/2022 11:59:59 PM	3/24/2022 11:59:59 PM
03/18/2022 7:41:16 PM	3/15/2022 11:59:59 PM	3/25/2022 11:59:59 PM
03/19/2022 8:14:15 PM	3/16/2022 11:59:59 PM	3/26/2022 11:59:59 PM
03/20/2022 9:23:51 PM	3/17/2022 11:59:59 PM	3/27/2022 11:59:59 PM
03/21/2022 10:04:41 PM	3/18/2022 11:59:59 PM	3/28/2022 11:59:59 PM

Значения новых полей находятся в `TimestampFormat M/D/YYYY h:mm:ss[.fff] TT`. Так как была использована функция `dayend()`, значения отметок времени представляют последнюю миллисекунду дня.

Значения срока авансового платежа на три дня предшествуют сроку предоставления услуги, так как второй аргумент, переданный в функции `dayend()`, имеет отрицательное значение.

Значения срока окончательного платежа следуют через семь дней после срока обслуживания, так как второй аргумент, переданный в функции `dayend()`, имеет положительное значение.

Пример 3. Скрипт day_start

Скрипт загрузки и результаты

Обзор

Откройте Редактор загрузки данных и добавьте приведенный ниже скрипт загрузки на новую вкладку.

В этом примере используются те же набор данных и сценарий, что и в предыдущем примере.

Как и в предыдущем примере, вы создадите два новых поля:

- `deposit_due_date`: дата, когда должен быть получен авансовый платеж. Это окончание дня за три дня до `service_date`.
- `final_payment_due_date`: дата, когда должен быть получен окончательный платеж. Это окончание дня через семь дней после `service_date`.

Однако ваша компания хочет соблюдать следующее правило: рабочий день начинается в 17:00 и заканчивается в 17:00 следующего дня. Таким образом компания сможет осуществлять мониторинг транзакций в эти рабочие часы.

С целью удовлетворения этих требований создаются два вышеупомянутых поля в рамках предшествующей загрузки с использованием функции dayend() и всех трех параметров: time, period_no и day_start.

Скрипт загрузки

```
SET TimestampFormat='M/D/YYYY h:mm:ss[.fff] TT';

Services:
   Load
     *,
     dayend(service_date,-3,17/24) as deposit_due_date,
     dayend(service_date,7,17/24) as final_payment_due_date
     ;
Load
service_id,
service_date,
amount
Inline
[
service_id, service_date,amount
1,03/11/2022 9:25:14 AM,231.24
2,03/12/2022 10:06:54 AM,567.28
3,03/13/2022 10:44:42 AM,364.28
4,03/14/2022 11:33:30 AM,575.76
5,03/15/2022 12:58:14 PM,638.68
6,03/16/2022 4:23:12 PM,785.38
7,03/17/2022 6:42:15 PM,967.46
8,03/18/2022 7:41:16 PM,287.67
9,03/19/2022 8:14:15 PM,764.45
10,03/20/2022 9:23:51 PM,875.43
11,03/21/2022 10:04:41 PM,957.35
];
```

Результаты

Загрузите данные и откройте лист. Создайте новую таблицу и добавьте эти поля как измерения:

- service_date
- deposit_due_date
- final_payment_due_date

Результирующая таблица

service_date	deposit_due_date	final_payment_due_date
03/11/2022 9:25:14 AM	3/8/2022 4:59:59 PM	3/18/2022 4:59:59 PM
03/12/2022 10:06:54 AM	3/9/2022 4:59:59 PM	3/19/2022 4:59:59 PM
03/13/2022 10:44:42 AM	3/10/2022 4:59:59 PM	3/20/2022 4:59:59 PM
03/14/2022 11:33:30 AM	3/11/2022 4:59:59 PM	3/21/2022 4:59:59 PM

service_date	deposit_due_date	final_payment_due_date
03/15/2022 12:58:14 PM	3/12/2022 4:59:59 PM	3/22/2022 4:59:59 PM
03/16/2022 4:23:12 PM	3/13/2022 4:59:59 PM	3/23/2022 4:59:59 PM
03/17/2022 6:42:15 PM	3/14/2022 4:59:59 PM	3/24/2022 4:59:59 PM
03/18/2022 7:41:16 PM	3/15/2022 4:59:59 PM	3/25/2022 4:59:59 PM
03/19/2022 8:14:15 PM	3/16/2022 4:59:59 PM	3/26/2022 4:59:59 PM
03/20/2022 9:23:51 PM	3/17/2022 4:59:59 PM	3/27/2022 4:59:59 PM
03/21/2022 10:04:41 PM	3/18/2022 4:59:59 PM	3/28/2022 4:59:59 PM

Хотя даты остаются теми же, что и в примере 2, теперь их отметка времени — это последняя миллисекунда перед 17:00, так как третий аргумент, `day_start`, переданный в функцию `dayend()`, имеет значение 17/24.

Пример 4. Пример диаграммы

Скрипт загрузки и выражение диаграммы

Обзор

Откройте Редактор загрузки данных и добавьте приведенный ниже скрипт загрузки на новую вкладку.

В этом примере используются те же набор данных и сценарий, что и в двух предыдущих примерах. Компания хочет работать с соблюдением политики, согласно которой рабочий день начинается в 17:00 и заканчивается в 17:00 следующего дня.

Как и в предыдущем примере, вы создадите два новых поля:

- `deposit_due_date`: дата, когда должен быть получен авансовый платеж. Это окончание дня за три дня до `service_date`.
- `final_payment_due_date`: дата, когда должен быть получен окончательный платеж. Это окончание дня через семь дней после `service_date`.

Скрипт загрузки

```
SET TimestampFormat='M/D/YYYY h:mm:ss[.fff] TT';

Services:
Load
service_id,
service_date,
amount
Inline
[
service_id, service_date,amount
1,03/11/2022 9:25:14 AM,231.24
2,03/12/2022 10:06:54 AM,567.28
```

```
3,03/13/2022 10:44:42 AM,364.28
4,03/14/2022 11:33:30 AM,575.76
5,03/15/2022 12:58:14 PM,638.68
6,03/16/2022 4:23:12 PM,785.38
7,03/17/2022 6:42:15 PM,967.46
8,03/18/2022 7:41:16 PM,287.67
9,03/19/2022 8:14:15 PM,764.45
10,03/20/2022 9:23:51 PM,875.43
11,03/21/2022 10:04:41 PM,957.35
];
```

Результаты

Загрузите данные и откройте лист. Создайте новую таблицу и добавьте это поле как измерение:

`service_date`.

Чтобы создать поле `deposit_due_date`, создайте эту меру:

`=dayend(service_date,-3,17/24)`.

Затем для создания поля `final_payment_due_date` создайте эту меру:

`=dayend(service_date,7,17/24)`.

Результирующая таблица

service_date	=dayend(service_date,-3,17/24)	=dayend(service_date,7,17/24
03/11/2022	3/8/2022 16:59:59 PM	3/18/2022 16:59:59 PM
03/12/2022	3/9/2022 16:59:59 PM	3/19/2022 16:59:59 PM
03/13/2022	3/10/2022 16:59:59 PM	3/20/2022 16:59:59 PM
03/14/2022	3/11/2022 16:59:59 PM	3/21/2022 16:59:59 PM
03/15/2022	3/12/2022 16:59:59 PM	3/22/2022 16:59:59 PM
03/16/2022	3/13/2022 16:59:59 PM	3/23/2022 16:59:59 PM
03/17/2022	3/14/2022 16:59:59 PM	3/24/2022 16:59:59 PM
03/18/2022	3/15/2022 16:59:59 PM	3/25/2022 16:59:59 PM
03/19/2022	3/16/2022 16:59:59 PM	3/26/2022 16:59:59 PM
03/20/2022	3/17/2022 16:59:59 PM	3/27/2022 16:59:59 PM
03/21/2022	3/18/2022 16:59:59 PM	3/28/2022 16:59:59 PM

Значения новых полей находятся в `TimestampFormat M/D/YYYY h:mm:ss[.fff] TT`. Так как была использована функция `dayend()`, значения отметок времени представляют последнюю миллисекунду дня.

Значения срока платежа на три дня предшествуют сроку предоставления услуги, так как второй аргумент, переданный в функции `dayend()`, имеет отрицательное значение.

Значения срока окончательного платежа следуют через семь дней после срока обслуживания, так как второй аргумент, переданный в функции `dayend()`, имеет положительное значение.

В качестве отметки времени дат указана последняя миллисекунда перед 17:00, так как третий аргумент, `day_start`, переданный в функции `dayend()`, имеет значение 17/24.

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, что дни начинаются не в полночь, укажите смещение в виде десятичного значения в параметре **day_start**. Например, 0,125 обозначает 3:00.

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

InvDate	DName
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
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. Эта функция используется при создании основного календаря.

Синтаксис:

```
DayNumberOfQuarter(timestamp[,start_month])
```

Возвращаемые типы данных: целое

Аргументы

Аргумент	Описание
timestamp	Дата или метка времени 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.

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 функции

Пример	Результат
`DayNumberOfQuarter('12/09/2014')`	Возвращает 74, номер дня текущего квартала.

Пример	Результат
`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 1. Начало года в январе (скрипт)

Скрипт загрузки и результаты

Обзор

Откройте Редактор загрузки данных и добавьте приведенный ниже скрипт загрузки на новую вкладку.

Скрипт загрузки содержит следующее:

- Простой набор данных, содержащий список дат и загруженный в таблицу под именем `Calendar`. Используется системная переменная `DateFormat` со значением по умолчанию MM/DD/YYYY.
- Предшествующая загрузка, создающая дополнительное поле под именем `DayNrQtr` с использованием функции `DayNumberOfQuarter()`.

Помимо даты, в функцию не передаются дополнительные параметры.

Скрипт загрузки

```
SET DateFormat='MM/DD/YYYY';

Calendar:
Load
        date,
        DayNumberOfQuarter(date) as DayNrQtr
        ;
Load
date
Inline
[
date
01/01/2022
01/10/2022
01/31/2022
02/01/2022
02/10/2022
02/28/2022
03/01/2022
03/31/2022
04/01/2022
];
```

Результаты

Загрузите данные и откройте лист. Создайте новую таблицу и добавьте эти поля как измерения:

- `date`
- `daynrqtr`

Результирующая таблица

date	daynrqtr
01/01/2022	1
01/10/2022	10
01/31/2022	31
02/01/2022	32
02/10/2022	41
02/28/2022	59
03/01/2022	61
03/31/2022	91
04/01/2022	1

Первый день года – 1 января, так как в функцию `DayNumberOfQuarter()` не передан второй аргумент.

1 января – это первый день квартала, а 1 февраля – тридцать второй день квартала. 31 марта – это девяносто первый и последний день квартала, а 1 апреля – это первый день второго квартала.

Пример 2. Начало года в феврале (скрипт)

Скрипт загрузки и результаты

Обзор

Откройте Редактор загрузки данных и добавьте приведенный ниже скрипт загрузки на новую вкладку.

Скрипт загрузки содержит следующее:

- Используется тот же набор данных, что и в первом примере.
- Используется системная переменная `DateFormat` со значением по умолчанию MM/DD/YYYY.
- Аргумент `start_month`, указывающий на 1 февраля. Это задает в качестве начала финансового года 1 февраля.

Скрипт загрузки

```
SET DateFormat='MM/DD/YYYY';

Calendar:
Load
       date,
       DayNumberOfQuarter(date,2) as DayNrQtr
       ;
Load
date
Inline
[
date
01/01/2022
01/10/2022
01/31/2022
02/01/2022
02/10/2022
02/28/2022
03/01/2022
03/31/2022
04/01/2022
];
```

Результаты

Загрузите данные и откройте лист. Создайте новую таблицу и добавьте эти поля как измерения:

- `date`
- `daynrqtr`

Результирующая таблица

date	daynrqtr
01/01/2022	62
01/10/2022	71
01/31/2022	92
02/01/2022	1
02/10/2022	10
02/28/2022	28
03/01/2022	30
03/31/2022	60
04/01/2022	61

Первый день года – 1 февраля, так как в функцию `DayNumberOfQuarter()` был передан второй аргумент со значением 2.

Первый квартал года длится с февраля по апрель, а четвертый квартал – с ноября по январь. Это показано в результирующей таблице, где 1 февраля – это первый день квартала, а 31 января – девяносто второй и последний день квартала.

Пример 3. Начало года в январе (диаграмма)

Скрипт загрузки и выражение диаграммы

Обзор

Откройте Редактор загрузки данных и добавьте приведенный ниже скрипт загрузки на новую вкладку.

Скрипт загрузки содержит следующее:

- Используется тот же набор данных, что и в первом примере.
- Используется системная переменная `DateFormat` со значением по умолчанию MM/DD/YYYY.

Однако в этом примере в приложение загружается неизмененный набор данных. Значение дня квартала рассчитывается с использованием меры в объекте диаграммы.

Скрипт загрузки

```
SET DateFormat='MM/DD/YYYY';

Calendar:
Load
date
Inline
[
date
01/01/2022
01/10/2022
01/31/2022
02/01/2022
02/10/2022
02/28/2022
03/01/2022
03/31/2022
04/01/2022
];
```

Результаты

Загрузите данные и откройте лист. Создайте новую таблицу и добавьте это поле как измерение: `date`.

Создайте следующую меру:

```
=daynumberofquarter(date)
```

Результирующая таблица

date	=daynumberofquarter(date)
01/01/2022	1
01/10/2022	10
01/31/2022	31
02/01/2022	32
02/10/2022	41
02/28/2022	59
03/01/2022	61
03/31/2022	91
04/01/2022	1

Первый день года – 1 января, так как в функцию `DayNumberOfQuarter()` не передан второй аргумент.

1 января – это первый день квартала, а 1 февраля – тридцать второй день квартала. 31 марта – это девяносто первый и последний день квартала, а 1 апреля – это первый день второго квартала.

Пример 4. Начало года в феврале (диаграмма)

Скрипт загрузки и выражение диаграммы

Обзор

Откройте Редактор загрузки данных и добавьте приведенный ниже скрипт загрузки на новую вкладку.

Скрипт загрузки содержит следующее:

- Используется тот же набор данных, что и в первом примере.
- Используется системная переменная `DateFormat` со значением по умолчанию MM/DD/YYYY.
- Финансовый год длится с 1 февраля по 31 января.

Однако в этом примере в приложение загружается неизмененный набор данных. Значение дня квартала рассчитывается с использованием меры в объекте диаграммы.

Скрипт загрузки

```
SET DateFormat='MM/DD/YYYY';

Calendar:
Load
date
Inline
[
date
```

```
01/01/2022
01/10/2022
01/31/2022
02/01/2022
02/10/2022
02/28/2022
03/01/2022
03/31/2022
04/01/2022
];
```

Объект диаграммы

Загрузите данные и откройте лист. Создайте новую таблицу и добавьте это поле как измерение: `date`.

Создайте следующую меру:

```
=daynumberofquarter(date,2)
```

Результаты

Результирующая таблица

date	=daynumberofquarter(date,2)
01/01/2022	62
01/10/2022	71
01/31/2022	92
02/01/2022	1
02/10/2022	10
02/28/2022	28
03/01/2022	30
03/31/2022	60
04/01/2022	61

Первый день года – 1 февраля, так как в функцию `DayNumberOfQuarter()` был передан второй аргумент со значением 2.

Первый квартал года длится с февраля по апрель, а четвертый квартал – с ноября по январь. Это показано в результирующей таблице, где 1 февраля – это первый день квартала, а 31 января – девяносто второй и последний день квартала.

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

Аргументы

Аргумент	Описание
timestamp	Дата или метка времени 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.

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 функции

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 1. Начало года в январе (скрипт)

Скрипт загрузки и результаты

Обзор

Откройте Редактор загрузки данных и добавьте приведенный ниже скрипт загрузки на новую вкладку.

Скрипт загрузки содержит следующее:

- Простой набор данных, содержащий список дат и загруженный в таблицу под именем `Calendar`. Используется системная переменная `DateFormat` со значением по умолчанию MM/DD/YYYY.
- Предшествующая загрузка, создающая дополнительное поле под именем `daynryear` с использованием функции `DayNumberOfYear()`.

Помимо даты, в функцию не передаются дополнительные параметры.

Скрипт загрузки

```
SET DateFormat='MM/DD/YYYY';

Calendar:
Load
       date,
       DayNumberOfYear(date) as daynryear
       ;
Load
date
Inline
[
date
01/01/2022
01/10/2022
01/31/2022
02/01/2022
02/10/2022
06/30/2022
07/26/2022
10/31/2022
11/01/2022
12/31/2022
];
```

Результаты

Загрузите данные и откройте лист. Создайте новую таблицу и добавьте эти поля как измерения:

- `date`
- `daynryear`

Результирующая таблица

date	daynryear
01/01/2022	1
01/10/2022	10
01/31/2022	31
02/01/2022	32
02/10/2022	41

date	daynryear
06/30/2022	182
07/26/2022	208
10/31/2022	305
11/01/2022	306
12/31/2022	366

Первый день года – 1 января, так как в функцию `DayNumberOfYear()` не передан второй аргумент.

1 января – это первый день квартала, а 1 февраля – тридцать второй день года. 30 июня – это сто двадцать восьмой день, а 31 декабря – это триста шестьдесят шестой и последний день года.

Пример 2. Начало года в ноябре (скрипт)

Скрипт загрузки и результаты

Обзор

Откройте Редактор загрузки данных и добавьте приведенный ниже скрипт загрузки на новую вкладку.

Скрипт загрузки содержит следующее:

- Используется тот же набор данных, что и в первом примере.
- Используется системная переменная `DateFormat` со значением по умолчанию MM/DD/YYYY
- Аргумент `start_month`, начинающийся с 1 ноября. Это задает в качестве начала финансового года 1 ноября.

Скрипт загрузки

```
SET DateFormat='MM/DD/YYYY';

Calendar:
Load
       date,
       DayNumberOfYear(date,11) as daynryear
       ;
Load
date
Inline
[
date
01/01/2022
01/10/2022
01/31/2022
02/01/2022
02/10/2022
06/30/2022
```

```
07/26/2022
10/31/2022
11/01/2022
12/31/2022
];
```

Результаты

Загрузите данные и откройте лист. Создайте новую таблицу и добавьте эти поля как измерения:

- `date`
- `daynryear`

Результирующая таблица

date	daynryear
01/01/2022	62
01/10/2022	71
01/31/2022	92
02/01/2022	93
02/10/2022	102
06/30/2022	243
07/26/2022	269
10/31/2022	366
11/01/2022	1
12/31/2022	61

Первый день года – 1 ноября, так как в функцию `DayNumberOfYear()` был передан второй аргумент со значением 11.

1 января – это первый день квартала, а 1 февраля – тридцать второй день года. 30 июня – это сто восемьдесят второй день, а 31 декабря – это триста шестьдесят шестой и последний день года.

Пример 3. Начало года в январе (диаграмма)

Скрипт загрузки и выражение диаграммы

Обзор

Откройте Редактор загрузки данных и добавьте приведенный ниже скрипт загрузки на новую вкладку.

Скрипт загрузки содержит следующее:

- Используется тот же набор данных, что и в первом примере.
- Используется системная переменная `DateFormat` со значением по умолчанию MM/DD/YYYY.

Однако в этом примере в приложение загружается неизмененный набор данных. Значение дня квартала рассчитывается с использованием меры в объекте диаграммы.

Скрипт загрузки

```
SET DateFormat='MM/DD/YYYY';

Calendar:
Load
date
Inline
[
date
01/01/2022
01/10/2022
01/31/2022
02/01/2022
02/10/2022
06/30/2022
07/26/2022
10/31/2022
11/01/2022
12/31/2022
];
```

Результаты

Загрузите данные и откройте лист. Создайте новую таблицу и добавьте это поле как измерение: `date`.

Создайте следующую меру:

```
=daynumberofyear(date)
```

Результирующая таблица

date	=daynumberofyear(date)
01/01/2022	1
01/10/2022	10
01/31/2022	31
02/01/2022	32
02/10/2022	41
06/30/2022	182
07/26/2022	208

date	=daynumberofyear(date)
10/31/2022	305
11/01/2022	306
12/31/2022	366

Первый день года – 1 января, так как в функцию `DayNumberOfYear()` не передан второй аргумент.

1 января – это первый день года, а 1 февраля – тридцать второй день года. 30 июня – это сто восемьдесят второй день, а 31 декабря – это триста шестьдесят шестой и последний день года.

Пример 4. Начало года в ноябре (диаграмма)

Скрипт загрузки и выражение диаграммы

Обзор

Откройте Редактор загрузки данных и добавьте приведенный ниже скрипт загрузки на новую вкладку.

Скрипт загрузки содержит следующее:

- Используется тот же набор данных, что и в первом примере.
- Используется системная переменная `DateFormat` со значением по умолчанию MM/DD/YYYY.
- Финансовый год длится с 1 ноября по 31 октября.

Однако в этом примере в приложение загружается неизмененный набор данных. Значение дня года рассчитывается с использованием меры в объекте диаграммы.

Скрипт загрузки

```
SET DateFormat='MM/DD/YYYY';
Calendar:
Load
date
Inline
[
date
01/01/2022
01/10/2022
01/31/2022
02/01/2022
02/10/2022
06/30/2022
07/26/2022
10/31/2022
11/01/2022
12/31/2022
];
```

Результаты

Загрузите данные и откройте лист. Создайте новую таблицу и добавьте это поле как измерение: `date`.

Создайте следующую меру:

```
=daynumberofyear(date)
```

Результирующая таблица

date	=daynumberofyear(date,11)
01/01/2022	62
01/10/2022	71
01/31/2022	92
02/01/2022	93
02/10/2022	102
06/30/2022	243
07/26/2022	269
10/31/2022	366
11/01/2022	1
12/31/2022	61

Первый день года – 1 ноября, так как в функцию `DayNumberOfYear()` был передан второй аргумент со значением 11 .

Финансовый год длится с ноября по октябрь. Это показано в результирующей таблице, где 1 ноября – это первый день года, а 31 октября – триста шестьдесят шестой и последний день года.

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

Аргумент	Описание
time	Метка времени для вычисления.

Аргумент	Описание
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, что дни начинаются не в полночь, укажите смещение в виде десятичного значения в параметре **day_start**. Например, 0,125 обозначает 3:00. Другими словами, чтобы создать смещение, разделите время начала на 24 часа. Например, чтобы день начинался в 7:00, используйте дробь 7/24.

Когда это следует использовать

Функция `daystart()` широко используется в составе выражения, когда пользователю требуется учитывать в расчетах часть дня, которая уже прошла. Например, с ее помощью можно вычислить совокупную заработную плату, заработанную сотрудниками за этот день по состоянию на текущий момент.

В этих примерах используется формат метки времени `'M/D/YYYY h:mm:ss[.fff] TT'`. Формат метки времени указан в операторе `SET TimeStamp` в верхней части скрипта загрузки данных. Измените формат в примерах согласно своим пожеланиям.

Примеры функции

Пример	Результат
`daystart('01/25/2013 4:45:00 PM')`	Возвращает 1/25/2013 12:00:00 AM.
`daystart('1/25/2013 4:45:00 PM', -1)`	Возвращает 1/24/2013 12:00:00 AM.
`daystart('1/25/2013 16:45:00',0,0.5 )`	Возвращает 1/25/2013 12:00:00 PM.

Региональные настройки

Если не указано иное, в примерах, приведенных в данном разделе, используется следующий формат даты: ММ/ДД/ГГГГ. Формат даты указан в операторе `SET DateFormat` скрипта загрузки данных. В вашей системе может быть установлен другой формат даты по умолчанию в зависимости от региональных настроек и других факторов. Можно изменить формат в примерах в соответствии с потребностями. Или можно изменить форматы в скрипте загрузки в соответствии с этими примерами.

Региональные настройки по умолчанию в приложениях основаны на системных региональных настройках компьютера или сервера, где установлено ПО Qlik Sense. Если на сервере Qlik Sense, к которому обращается пользователь, выбран шведский язык, то редактор загрузки данных будет использовать шведские региональные настройки для даты, времени и валюты. Эти параметры регионального формата не связаны с языком, отображаемым в интерфейсе пользователя Qlik Sense. Qlik Sense будет отображаться на языке, который используется в браузере.

Пример 1. Простой пример

Скрипт загрузки и результаты

Обзор

Откройте редактор загрузки данных и добавьте приведенный ниже скрипт загрузки на новую вкладку.

Скрипт загрузки содержит следующее:

- Простой набор данных, содержащий список дат и загруженный в таблицу под именем `Calendar`.
- Используется системная переменная `TimeStampFormat` со значением по умолчанию `(M/D/YYYY h:mm:ss[.fff] TT`.
- Предшествующая загрузка, создающая дополнительное поле под именем `SOD_timestamp` с использованием функции `daystart()`.

Помимо даты, в функцию не передаются дополнительные параметры.

Скрипт загрузки

```
SET TimestampFormat='M/D/YYYY h:mm:ss[.fff] TT';

Calendar:
    Load
        date,
        daystart(date) as SOD_timestamp
        ;
Load
date
Inline
[
date
03/11/2022 1:47:15 AM
03/12/2022 4:34:58 AM
03/13/2022 5:15:55 AM
03/14/2022 9:25:14 AM
03/15/2022 10:06:54 AM
03/16/2022 10:44:42 AM
03/17/2022 11:33:30 AM
03/18/2022 12:58:14 PM
03/19/2022 4:23:12 PM
03/20/2022 6:42:15 PM
03/21/2022 7:41:16 PM
];
```

Результаты

Загрузите данные и откройте лист. Создайте новую таблицу и добавьте эти поля как измерения:

- date
- SOD_timestamp

Результирующая таблица

date	SOD_timestamp
03/11/2022 1:47:15 AM	3/11/2022 12:00:00 AM
03/12/2022 4:34:58 AM	3/12/2022 12:00:00 AM
03/13/2022 5:15:55 AM	3/13/2022 12:00:00 AM
03/14/2022 9:25:14 AM	3/14/2022 12:00:00 AM
03/15/2022 10:06:54 AM	3/15/2022 12:00:00 AM
03/16/2022 10:44:42 AM	3/16/2022 12:00:00 AM
03/17/2022 11:33:30 AM	3/17/2022 12:00:00 AM
03/18/2022 12:58:14 PM	3/18/2022 12:00:00 AM
03/19/2022 4:23:12 PM	3/19/2022 12:00:00 AM
03/20/2022 6:42:15 PM	3/20/2022 12:00:00 AM
03/21/2022 7:41:16 PM	3/21/2022 12:00:00 AM

Как показано в приведенной выше таблице, отметка времени окончания дня генерируется для каждой даты в наборе данных. Для отметки времени используется формат системной переменной `TimestampFormat M/D/YYYY h:mm:ss[.fff] TT`.

Пример 2. Скрипт period_no

Скрипт загрузки и результаты

Обзор

Откройте редактор загрузки данных и добавьте приведенный ниже скрипт загрузки на новую вкладку.

Скрипт загрузки содержит следующее:

- Набор данных, содержащий штрафы за парковку и загруженный в таблицу под именем `Fines`. Набор данных включает следующие поля:
  - id
  - due_date
  - number_plate
  - amount

- Предшествующая загрузка с использованием функции daystart(), которая предоставляет все три параметра: time, period_no и day_start. Эта предшествующая загрузка создает следующие два новых поля даты:
    - Поле даты early_repayment_period за семь дней до срока оплаты.
    - Поле даты late_penalty_period через 14 дней после срока оплаты.

Скрипт загрузки

```
SET TimestampFormat='M/D/YYYY h:mm:ss[.fff] TT';

Fines:
    Load
        *,
        daystart(due_date,-7) as early_repayment_period,
        daystart(due_date,14) as late_penalty_period
        ;
Load
*
Inline
[
id, due_date, number_plate,amount
1,02/11/2022, 573RJG,50.00
2,03/25/2022, SC41854,50.00
3,04/14/2022, 8EHZ378,50.00
4,06/28/2022, 8HSS198,50.00
5,08/15/2022, 1221665,50.00
6,11/16/2022, EAK473,50.00
7,01/17/2023, KD6822,50.00
8,03/22/2023, 1GGLB,50.00
];
```

Результаты

Загрузите данные и откройте лист. Создайте новую таблицу и добавьте эти поля как измерения:

- due_date
- early_repayment_period
- late_penalty_period

Результирующая таблица

due_date	early_repayment_period	late_penalty_period
02/11/2022 9:25:14 AM	2/4/2022 12:00:00 AM	2/25/2022 12:00:00 AM
03/25/2022 10:06:54 AM	3/18/2022 12:00:00 AM	4/8/2022 12:00:00 AM
04/14/2022 10:44:42 AM	4/7/2022 12:00:00 AM	4/28/2022 12:00:00 AM
06/28/2022 11:33:30 AM	6/21/2022 12:00:00 AM	7/12/2022 12:00:00 AM
08/15/2022 12:58:14 PM	8/8/2022 12:00:00 AM	8/29/2022 12:00:00 AM

due_date	early_repayment_period	late_penalty_period
11/16/2022 4:23:12 PM	11/9/2022 12:00:00 AM	11/30/2022 12:00:00 AM
01/17/2023 6:42:15 PM	1/10/2023 12:00:00 AM	1/31/2023 12:00:00 AM
03/22/2023 7:41:16 PM	3/15/2023 12:00:00 AM	4/5/2023 12:00:00 AM

Значения новых полей находятся в `TimestampFormat M/DD/YYYY tt`. Так как была использована функция `daystart()`, все значения меток времени представляют первую миллисекунду дня.

Значения досрочного платежа на семь дней предшествуют сроку платежа, переданному во втором аргументе в функции `daystart()`, и являются отрицательными.

Значения просроченного платежа следуют через 14 дней после срока платежа, переданного во втором аргументе в функции `daystart()`, и являются положительными.

Пример 3. Скрипт day_start

Скрипт загрузки и результаты

Обзор

Откройте редактор загрузки данных и добавьте приведенный ниже скрипт загрузки на новую вкладку.

Скрипт загрузки содержит следующее:

- Используется тот же набор данных и сценарий, что в предыдущем примере.
- Используется та же предшествующая загрузка, что в предыдущем примере.

В этом примере начало и окончание рабочего дня устанавливается на 7:00 утра (7:00 AM) каждый день.

Скрипт загрузки

```
SET DateFormat='MM/DD/YYYY';

Fines:
    Load
        *,
        daystart(due_date,-7,7/24) as early_repayment_period,
        daystart(due_date,14, 7/24) as late_penalty_period
        ;
Load
*
Inline
[
id, due_date, number_plate,amount
1,02/11/2022, 573RJG,50.00
2,03/25/2022, SC41854,50.00
3,04/14/2022, 8EHZ378,50.00
```

```
4,06/28/2022, 8HSS198,50.00
5,08/15/2022, 1221665,50.00
6,11/16/2022, EAK473,50.00
7,01/17/2023, KD6822,50.00
8,03/22/2023, 1GGLB,50.00
];
```

Результаты

Загрузите данные и откройте лист. Создайте новую таблицу и добавьте эти поля как измерения:

- due_date
- early_repayment_period
- late_penalty_period

Результирующая таблица

due_date	early_repayment_period	late_penalty_period
02/11/2022	2/3/2022 7:00:00 AM	2/24/2022 7:00:00 AM
03/25/2022	3/17/2022 7:00:00 AM	4/7/2022 7:00:00 AM
04/14/2022	4/6/2022 7:00:00 AM	4/27/2022 7:00:00 AM
06/28/2022	6/20/2022 7:00:00 AM	7/11/2022 7:00:00 AM
08/15/2022	8/7/2022 7:00:00 AM	8/28/2022 7:00:00 AM
11/16/2022	11/8/2022 7:00:00 AM	11/29/2022 7:00:00 AM
01/17/2023	1/9/2023 7:00:00 AM	1/30/2023 7:00:00 AM
03/22/2023	3/14/2023 7:00:00 AM	4/4/2023 7:00:00 AM

Теперь в качестве метки времени дат указано 7:00 AM, так как аргумент day_start, переданный в функции daystart(), имеет значение 7/24. В результате этого начало рабочего дня устанавливается на 7:00 утра.

Так как поле due_date не имеет метки времени, для него используется значение 12:00 AM (полночь), то есть оно относится к предыдущему дню, так как рабочий день начинается и заканчивается в 7:00 утра. Поэтому период досрочного погашения штрафа со сроком оплаты 11 февраля начинается 3 февраля в 7:00 утра.

Пример 4. Пример объекта диаграммы

Скрипт загрузки и выражение диаграммы

Обзор

Откройте редактор загрузки данных и добавьте приведенный ниже скрипт загрузки на новую вкладку.

В этом примере используется тот же набор данных и сценарий, что в предыдущем примере.

Однако в приложение загружается только оригинальная таблица `Fines` с двумя дополнительными значениями сроков, вычисляемыми в объекте диаграммы.

Скрипт загрузки

```
SET TimestampFormat='M/D/YYYY h:mm:ss[.fff] TT';

Fines:
    Load
*
Inline
[
id, due_date, numer_plate,amount
1,02/11/2022 9:25:14 AM, 573RJG,50.00
2,03/25/2022 10:06:54 AM, SC41854,50.00
3,04/14/2022 10:44:42 AM, 8EHZ378,50.00
4,06/28/2022 11:33:30 AM, 8HSS198,50.00
5,08/15/2022 12:58:14 PM, 1221665,50.00
6,11/16/2022 4:23:12 PM, EAK473,50.00
7,01/17/2023 6:42:15 PM, KD6822,50.00
8,03/22/2023 7:41:16 PM, 1GGLB,50.00
];
```

Результаты

Выполните следующие действия.

1. Загрузите данные и откройте лист. Создайте новую таблицу и добавьте это поле как измерение: `due_date`.
2. Чтобы создать поле `early_repayment_period`, создайте следующую меру: `=daystart(due_date,-7,7/24)`
3. Чтобы создать поле `late_penalty_period`, создайте следующую меру: =daystart(due_date,14,7/24)

Результирующая таблица

due_date	=daystart(due_date,-7,7/24)	=daystart(due_date,14,7/24)
02/11/2022 9:25:14 AM	2/4/2022 7:00:00 AM	2/25/2022 7:00:00 AM
03/25/2022 10:06:54 AM	3/18/2022 7:00:00 AM	4/8/2022 7:00:00 AM
04/14/2022 10:44:42 AM	4/7/2022 7:00:00 AM	4/28/2022 7:00:00 AM
06/28/2022 11:33:30 AM	6/21/2022 7:00:00 AM	7/12/2022 7:00:00 AM
08/15/2022 12:58:14 PM	8/8/2022 7:00:00 AM	8/29/2022 7:00:00 AM
11/16/2022 4:23:12 PM	11/9/2022 7:00:00 AM	11/30/2022 7:00:00 AM
01/17/2023 6:42:15 PM	1/10/2023 7:00:00 AM	1/31/2023 7:00:00 AM
03/22/2023 7:41:16 PM	3/15/2023 7:00:00 AM	4/5/2023 7:00:00 AM

Значения новых полей находятся в `TimestampFormat M/D/YYYY h:mm:ss[.fff] TT`. Так как была использована функция `daystart()`, значения меток времени соответствуют первой миллисекунде дня.

Значения досрочного платежа на семь дней предшествуют сроку платежа, так как второй аргумент, переданный в функции `daystart()`, имеет отрицательное значение.

Значения просроченного платежа наступают через 14 дней после срока платежа, так как второй аргумент, переданный в функции `daystart()`, имеет положительное значение.

В качестве отметки времени дат указано 7:00 утра (7:00 AM), так как третий аргумент `daystart()`, переданный в функции `day_start`, имеет значение 7/24.

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. Праздник обозначен как постоянная строка даты. Можно указать несколько дат праздников, разделенных запятыми. **Пример:** `'12/25/2013', '12/26/2013', '12/31/2013', '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 значение Greenwich Mean Time согласно региональным настройкам. Эта функция возвращает значения в формате системной переменной `TimestampFormat`.

При каждой перезагрузке приложения все таблицы скрипта загрузки, переменные или объекты диаграммы, использующие функцию GMT, будут скорректированы с учетом самого актуального текущего среднего времени по Гринвичу, полученного от системных часов.

Синтаксис:

```
GMT( )
```

Возвращаемые типы данных: двойное значение

В этих примерах используется формат метки времени M/D/YYYY h:mm:ss[.fff] TT. Формат даты указан в операторе SET TimestampFormat в верхней части скрипта загрузки данных. Измените формат в примерах согласно своим пожеланиям.

Примеры функции

Пример	Результат
GMT()	3/28/2022 2:47:36 PM

Региональные настройки

Если не указано иное, в примерах, приведенных в данном разделе, используется следующий формат даты: ММ/ДД/ГГГГ. Формат даты указан в операторе SET DateFormat скрипта загрузки данных. В вашей системе может быть установлен другой формат даты по умолчанию в зависимости от региональных настроек и других факторов. Можно изменить формат в примерах в соответствии с потребностями. Или можно изменить форматы в скрипте загрузки в соответствии с этими примерами.

Региональные настройки по умолчанию в приложениях основаны на системных региональных настройках компьютера или сервера, где установлено ПО Qlik Sense. Если на сервере Qlik Sense, к которому обращается пользователь, выбран шведский язык, то редактор загрузки данных будет использовать шведские региональные настройки для даты, времени и валюты. Эти параметры регионального формата не связаны с языком, отображаемым в интерфейсе пользователя Qlik Sense. Qlik Sense будет отображаться на языке, который используется в браузере.

Пример 1. Переменная (скрипт)

Скрипт загрузки и результаты

Обзор

Откройте редактор загрузки данных и добавьте приведенный ниже скрипт загрузки на новую вкладку. В этом примере текущее среднее время по Гринвичу задается в качестве переменной в скрипте загрузки с использованием функции GMT.

Скрипт загрузки

```
LET vGMT = GMT();
```

Результаты

Загрузите данные и создайте лист. Создайте текстовое поле, используя объект диаграммы **Текст и изображение**.

Добавьте эту меру в текстовое поле:

```
=vGMT
```

Текстовое поле должно содержать строку текста с датой и временем, как показано ниже:

```
3/28/2022 2:47:36 PM
```

Пример 2. Начало года в ноябре (скрипт)

Скрипт загрузки и результаты

Обзор

Откройте редактор загрузки данных и добавьте приведенный ниже скрипт загрузки на новую вкладку.

Скрипт загрузки содержит следующее:

- Набор данных, содержащий не возвращенные в срок библиотечные книги и загруженный в таблицу под именем `Overdue`. Используется системная переменная `DateFormat` со значением по умолчанию MM/DD/YYYY.
- Создается новое поле `days_overdue`, которое вычисляет срок задержки возврата каждой книги.

Скрипт загрузки

```
SET DateFormat='MM/DD/YYYY';

Overdue:
    Load
        *,
        Floor(GMT()-due_date) as days_overdue
        ;
Load
*
Inline
[
cust_id,book_id,due_date
1,4,01/01/2021,
2,24,01/10/2021,
6,173,01/31/2021,
31,281,02/01/2021,
86,265,02/10/2021,
52,465,06/30/2021,
26,537,07/26/2021,
92,275,10/31/2021,
```

```
27,455,11/01/2021,
27,46,12/31/2021
];
```

Результаты

Загрузите данные и откройте лист. Создайте новую таблицу и добавьте эти поля как измерения:

- due_date
- book_id
- days_overdue

Результирующая таблица

due_date	book_id	days_overdue
01/01/2021	4	455
01/10/2021	24	446
01/31/2021	173	425
02/01/2021	281	424
02/10/2021	265	415
06/30/2021	465	275
07/26/2021	537	249
10/31/2021	275	152
11/01/2021	455	151
12/31/2021	46	91

Значения в поле days_overdue вычисляются как разница между текущим средним временем по Гринвичу, полученным с помощью функции GMT(), и первоначальным сроком. Чтобы вычислить только дни, результаты округляются до ближайшего целого числа с использованием функции Floor ().

Пример 3. Объект диаграммы (диаграмма)

Скрипт загрузки и выражение диаграммы

Обзор

Откройте редактор загрузки данных и добавьте приведенный ниже скрипт загрузки на новую вкладку. Скрипт загрузки содержит тот же набор данных, что в первом примере. Используется системная переменная DateFormat со значением по умолчанию MM/DD/YYYY.

Однако в этом примере в приложение загружается неизмененный набор данных. Количество дней задержки вычисляется с использованием меры в объекте диаграммы.

Скрипт загрузки

```
SET DateFormat='MM/DD/YYYY';

Overdue:
Load
*
Inline
[
cust_id,book_id,due_date
1,4,01/01/2021,
2,24,01/10/2021,
6,173,01/31/2021,
31,281,02/01/2021,
86,265,02/10/2021,
52,465,06/30/2021,
26,537,07/26/2021,
92,275,10/31/2021,
27,455,11/01/2021,
27,46,12/31/2021
];
```

Результаты

Загрузите данные и откройте лист. Создайте новую таблицу и добавьте эти поля как измерения:

- due_date
- book_id

Создайте следующую меру:

```
=Floor(GMT() - due_date)
```

Результирующая таблица

due_date	book_id	=Floor(GMT()-due_date)
01/01/2021	4	455
01/10/2021	24	446
01/31/2021	173	425
02/01/2021	281	424
02/10/2021	265	415
06/30/2021	465	275
07/26/2021	537	249
10/31/2021	275	152
11/01/2021	455	151
12/31/2021	46	91

Значения в поле days_overdue вычисляются как разница между текущим средним временем по Гринвичу, полученным с помощью функции GMT(), и первоначальным сроком. Чтобы вычислить только дни, результаты округляются до ближайшего целого числа с использованием функции Floor ().

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

Региональные настройки

Если не указано иное, в примерах, приведенных в данном разделе, используется следующий формат даты: ММ/ДД/ГГГГ. Формат даты указан в операторе SET DateFormat скрипта загрузки данных. В вашей системе может быть установлен другой формат даты по умолчанию в зависимости от региональных настроек и других факторов. Можно изменить формат в примерах в соответствии с потребностями. Или можно изменить форматы в скрипте загрузки в соответствии с этими примерами.

Региональные настройки по умолчанию в приложениях основаны на системных региональных настройках компьютера или сервера, где установлено ПО Qlik Sense. Если на сервере Qlik Sense, к которому обращается пользователь, выбран шведский язык, то редактор загрузки данных будет использовать шведские региональные настройки для даты, времени и валюты. Эти параметры регионального формата не связаны с языком, отображаемым в интерфейсе пользователя Qlik Sense. Qlik Sense будет отображаться на языке, который используется в браузере.

Примеры функции

Пример	Результат
hour('09:14:36')	Переданная текстовая строка неявно преобразуется в метку времени, так как она соответствует формату метки времени, определенному в переменной TimestampFormat. Это выражение возвращает 9.
hour('0.5555')	Выражение возвращает 13 (так как 0,5555 = 13:19:55)

Пример 1. Переменная (скрипт)

Скрипт загрузки и результаты

Обзор

Откройте Редактор загрузки данных и добавьте приведенный ниже скрипт загрузки на новую вкладку.

Скрипт загрузки содержит следующее:

- Набор данных, содержащий транзакции по метке времени
- Переменная `TimeStamp` системы `(M/D/YYYY h:mm:ss[.fff] TT)` по умолчанию.

Создайте поле, «`hour`», рассчитывающее время совершения покупок.

Скрипт загрузки

```
SET TimestampFormat='M/D/YYYY h:mm:ss[.fff] TT';

Transactions:
  Load
    *,
    hour(date) as hour
    ;
Load
*
Inline
[
id,date,amount
9497,'2022-01-05 19:04:57',47.25,
9498,'2022-01-03 14:21:53',51.75,
9499,'2022-01-03 05:40:49',73.53,
9500,'2022-01-04 18:49:38',15.35,
9501,'2022-01-01 22:10:22',31.43,
9502,'2022-01-05 19:34:46',13.24,
9503,'2022-01-04 22:58:34',74.34,
9504,'2022-01-06 11:29:38',50.00,
9505,'2022-01-02 08:35:54',36.34,
9506,'2022-01-06 08:49:09',74.23
];
```

Результаты

Загрузите данные и откройте лист. Создайте новую таблицу и добавьте эти поля как измерения:

- `date`
- `hour`

Результирующая таблица

date	hour
2022-01-01 22:10:22	22
2022-01-02 08:35:54	8
2022-01-03 05:40:49	5
2022-01-03 14:21:53	14
2022-01-04 18:49:38	18

date	hour
2022-01-04 22:58:34	22
2022-01-05 19:04:57	19
2022-01-05 19:34:46	19
2022-01-06 08:49:09	8
2022-01-06 11:29:38	11

Значение в поле часа создаются с использованием функции `hour()` и путем передачи даты в качестве выражения в предшествующем операторе load.

Пример 2. Объект диаграммы (диаграмма)

Скрипт загрузки и выражение диаграммы

Обзор

Откройте Редактор загрузки данных и добавьте приведенный ниже скрипт загрузки на новую вкладку.

Скрипт загрузки содержит следующее:

- Используется тот же набор данных, что и в первом примере.
- Переменная `TimeStamp` системы `(M/D/YYYY h:mm:ss[.fff] TT)` по умолчанию.

Однако в этом примере в приложение загружается неизмененный набор данных. Значения `hour` рассчитывается с использованием меры в объекте диаграммы.

Скрипт загрузки

```
SET TimestampFormat='M/D/YYYY h:mm:ss[.fff] TT';

Transactions:
Load
*
Inline
[
id,date,amount
9497,'2022-01-05 19:04:57',47.25,
9498,'2022-01-03 14:21:53',51.75,
9499,'2022-01-03 05:40:49',73.53,
9500,'2022-01-04 18:49:38',15.35,
9501,'2022-01-01 22:10:22',31.43,
9502,'2022-01-05 19:34:46',13.24,
9503,'2022-01-04 22:58:34',74.34,
9504,'2022-01-06 11:29:38',50.00,
9505,'2022-01-02 08:35:54',36.34,
9506,'2022-01-06 08:49:09',74.23
];
```

Результаты

Загрузите данные и откройте лист. Создайте новую таблицу и добавьте это поле как измерение: `date`.

Чтобы рассчитать `hour`, создайте следующую меру:

```
=hour(date)
```

Результирующая таблица

due_date	=hour(date)
2022-01-01 22:10:22	22
2022-01-02 08:35:54	8
2022-01-03 05:40:49	5
2022-01-03 14:21:53	14
2022-01-04 18:49:38	18
2022-01-04 22:58:34	22
2022-01-05 19:04:57	19
2022-01-05 19:34:46	19
2022-01-06 08:49:09	8
2022-01-06 11:29:38	11

Значения для «`hour`» создаются с использованием функции `hour()`, где дата передается в виде выражения в мере для объекта диаграммы.

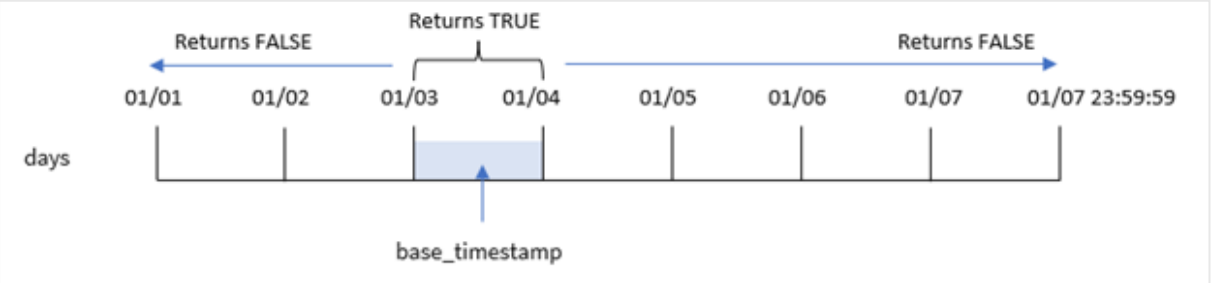
inday

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])
```

Схема функции `inday`



Функция `inday()` использует аргумент `base_timestamp` для идентификации дня, на который выпадает метка времени. Временем начала дня по умолчанию является полночь, но его можно изменить с помощью аргумента `day_start` функции `inday()`. Когда этот день определен, функция будет возвращать логические результаты при сравнении предварительно заданных значений метки времени с этим днем.

Когда это следует использовать

Функция `inday()` возвращает результат в виде логического значения. Обычно этот тип функции используется в качестве условия в `if expression`. Она возвращает агрегирование или расчет в зависимости от того, совпадает ли проверяемая дата с днем рассматриваемой метки времени.

Например, функцию `inday()` можно использовать для идентификации всего оборудования, изготовленного в указанный день.

Возвращаемые типы данных: Булево значение

В Qlik Sense логическое значение «истина» представлено как -1, а «ложь» – как 0.

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).

Региональные настройки

Если не указано иное, в примерах, приведенных в данном разделе, используется следующий формат даты: ММ/ДД/ГГГГ. Формат даты указан в операторе `SET DateFormat` скрипта загрузки данных. В вашей системе может быть установлен другой формат даты по умолчанию в зависимости от региональных настроек и других факторов. Можно изменить формат в примерах в соответствии с потребностями. Или можно изменить форматы в скрипте загрузки в соответствии с этими примерами.

Региональные настройки по умолчанию в приложениях основаны на системных региональных настройках компьютера или сервера, где установлено ПО Qlik Sense. Если на сервере Qlik Sense, к которому обращается пользователь, выбран шведский язык, то редактор загрузки данных будет

использовать шведские региональные настройки для даты, времени и валюты. Эти параметры регионального формата не связаны с языком, отображаемым в интерфейсе пользователя Qlik Sense. Qlik Sense будет отображаться на языке, который используется в браузере.

Примеры функции

Пример	Результат
`inday` ('01/12/2006 12:23:00 PM', '01/12/2006 12:00:00 AM', 0)	Возвращает True
`inday` ('01/12/2006 12:23:00 PM', '01/13/2006 12:00:00 AM', 0)	Возвращает False
`inday` ('01/12/2006 12:23:00 PM', '01/12/2006 12:00:00 AM', -1)	Возвращает False
`inday` ('01/11/2006 12:23:00 PM', '01/12/2006 12:00:00 AM', -1)	Возвращает True
`inday` ('01/12/2006 12:23:00 PM', '01/12/2006 12:00:00 AM', 0, 0.5)	Возвращает False
`inday` ('01/12/2006 11:23:00 AM', '01/12/2006 12:00:00 AM', 0, 0.5)	Возвращает True

Пример 1. Оператор Load (скрипт)

Скрипт загрузки и результаты

Обзор

Откройте редактор загрузки данных и добавьте приведенный ниже скрипт загрузки на новую вкладку.

Скрипт загрузки содержит следующее:

- Набор данных, содержащий транзакции по метке времени, загруженный в таблицу под именем `Transactions`.
- Поле данных, предоставленное в формате переменной `(M/D/YYYY h:mm:ss[.fff] TT)` системы `TimeStamp`.
- Предшествующая загрузка, которая содержит функцию `inday()`, настроенную как поле `in_day`.

Скрипт загрузки

```
SET TimestampFormat='M/D/YYYY h:mm:ss[.fff] TT';

Transactions:
    Load
        *,
        inday(date,'01/05/2022 12:00:00 AM', 0) as in_day
        ;
Load
*
Inline
[
id,date,amount
9497,'01/01/2022 7:34:46 PM',13.24
```

```
9498,'01/01/2022 10:10:22 PM',31.43
9499,'01/02/2022 8:35:54 AM',36.34
9500,'01/03/2022 2:21:53 PM',51.75
9501,'01/04/2022 6:49:38 PM',15.35
9502,'01/04/2022 10:58:34 PM',74.34
9503,'01/05/2022 5:40:49 AM',73.53
9504,'01/05/2022 11:29:38 AM',50.00
9505,'01/05/2022 7:04:57 PM',47.25
9506,'01/06/2022 8:49:09 AM',74.23
];
```

Результаты

Загрузите данные и откройте лист. Создайте новую таблицу и добавьте эти поля как измерения:

- `date`
- `in_day`

Результирующая таблица

date	in_day
01/01/2022 7:34:46 PM	0
01/01/2022 10:10:22 PM	0
01/02/2022 8:35:54 AM	0
01/03/2022 2:21:53 PM	0
01/04/2022 6:49:38 PM	0
01/04/2022 10:58:34 PM	0
01/05/2022 5:40:49 AM	-1
01/05/2022 11:29:38 AM	-1
01/05/2022 7:04:57 PM	-1
01/06/2022 8:49:09 AM	0

Поле `in_day` создано предшествующим оператором load с использованием функции `inday()`, где в качестве аргументов функции переданы поле даты, жестко запрограммированная метка времени 5 января и `period_no = 0`.

Пример 2. Скрипт period_no

Скрипт загрузки и результаты

Обзор

В скрипте загрузки используются те же набор данных и сценарий, что и в первом примере.

Однако в этом примере стоит задача рассчитать, произошла ли транзакция за два дня до 5 января.

Скрипт загрузки

```
SET TimestampFormat='M/D/YYYY h:mm:ss[.fff] TT';

Transactions:
    Load
        *,
        inday(date,'01/05/2022 12:00:00 AM', -2) as in_day
        ;
Load
*
Inline
[
id,date,amount
9497,'01/01/2022 7:34:46 PM',13.24
9498,'01/01/2022 10:10:22 PM',31.43
9499,'01/02/2022 8:35:54 AM',36.34
9500,'01/03/2022 2:21:53 PM',51.75
9501,'01/04/2022 6:49:38 PM',15.35
9502,'01/04/2022 10:58:34 PM',74.34
9503,'01/05/2022 5:40:49 AM',73.53
9504,'01/05/2022 11:29:38 AM',50.00
9505,'01/05/2022 7:04:57 PM',47.25
9506,'01/06/2022 8:49:09 AM',74.23
];
```

Результаты

Загрузите данные и откройте лист. Создайте новую таблицу и добавьте эти поля как измерения:

- date
- in_day

Результирующая таблица

date	in_day
01/01/2022 7:34:46 PM	0
01/01/2022 10:10:22 PM	0
01/02/2022 8:35:54 AM	0
01/03/2022 2:21:53 PM	-1
01/04/2022 6:49:38 PM	0
01/04/2022 10:58:34 PM	0
01/05/2022 5:40:49 AM	0
01/05/2022 11:29:38 AM	0
01/05/2022 7:04:57 PM	0
01/06/2022 8:49:09 AM	0

В этом примере в качестве аргумента смещения в функции `inday()` использовалось значение `period_no = -2`, поэтому функция определяет, произошла ли транзакция 3 января. Это можно проверить в таблице выходных данных, где одна транзакция возвращает результат в виде логического значения «ИСТИНА».

Пример 3. Скрипт day_start

Скрипт загрузки и результаты

Обзор

В скрипте загрузки используются те же набор данных и сценарий, что и в предыдущих примерах.

Однако в этом примере используется политика компании, согласно которой рабочий день начинается и заканчивается в 7:00 утра.

Скрипт загрузки

```
SET TimestampFormat='M/D/YYYY h:mm:ss[.fff] TT';

Transactions:
    Load
        *,
        inday(date,'01/05/2022 12:00:00 AM', 0, 7/24) as in_day
        ;
Load
*
Inline
[
id,date,amount
9497,'01/01/2022 7:34:46 PM',13.24
9498,'01/01/2022 10:10:22 PM',31.43
9499,'01/02/2022 8:35:54 AM',36.34
9500,'01/03/2022 2:21:53 PM',51.75
9501,'01/04/2022 6:49:38 PM',15.35
9502,'01/04/2022 10:58:34 PM',74.34
9503,'01/05/2022 5:40:49 AM',73.53
9504,'01/05/2022 11:29:38 AM',50.00
9505,'01/05/2022 7:04:57 PM',47.25
9506,'01/06/2022 8:49:09 AM',74.23
];
```

Результаты

Загрузите данные и откройте лист. Создайте новую таблицу и добавьте эти поля как измерения:

- `date`
- `in_day`

Результирующая таблица

date	in_day
01/01/2022 7:34:46 PM	0
01/01/2022 10:10:22 PM	0
01/02/2022 8:35:54 AM	0
01/03/2022 2:21:53 PM	0
01/04/2022 6:49:38 PM	-1
01/04/2022 10:58:34 PM	-1
01/05/2022 5:40:49 AM	-1
01/05/2022 11:29:38 AM	0
01/05/2022 7:04:57 PM	0
01/06/2022 8:49:09 AM	0

В функции `inday()` используется аргумент `start_day` = 7/24, который задает время 7:00, поэтому функция определяет, осуществлена ли каждая транзакция в период с 7:00 4 января до 7:00 5 января.

Это можно проверить в таблице выходных данных, где транзакции, которые имели место после 7:00 4 января, возвращают логическое значение «ИСТИНА», а транзакции после 7:00 5 января возвращают логическое значение «ЛОЖЬ».

Пример 4. Объект диаграммы

Скрипт загрузки и выражение диаграммы

Обзор

В скрипте загрузки используются те же набор данных и сценарий, что и в предыдущих примерах.

Однако в этом примере в приложение загружается неизмененный набор данных. Выполнив расчет, вы определите, осуществлена ли транзакция 5 января, создав меру в объекте диаграммы.

Скрипт загрузки

```
Transactions:
Load
*
Inline
[
id,date,amount
9497,'01/01/2022 7:34:46 PM',13.24
9498,'01/01/2022 10:10:22 PM',31.43
9499,'01/02/2022 8:35:54 AM',36.34
9500,'01/03/2022 2:21:53 PM',51.75
9501,'01/04/2022 6:49:38 PM',15.35
```

```
9502,'01/04/2022 10:58:34 PM',74.34
9503,'01/05/2022 5:40:49 AM',73.53
9504,'01/05/2022 11:29:38 AM',50.00
9505,'01/05/2022 7:04:57 PM',47.25
9506,'01/06/2022 8:49:09 AM',74.23
];
```

Результаты

Загрузите данные и откройте лист. Создайте новую таблицу и добавьте это поле как измерение:

- date

Чтобы рассчитать, совершена ли транзакция 5 января, создайте следующую меру:

```
=inday(date,'01/05/2022 12:00:00 AM',0)
```

Результирующая таблица

date	inday(date,'01/05/2022 12:00:00 AM',0)
01/01/2022 7:34:46 PM	0
01/01/2022 10:10:22 PM	0
01/02/2022 8:35:54 AM	0
01/03/2022 2:21:53 PM	0
01/04/2022 6:49:38 PM	0
01/04/2022 10:58:34 PM	0
01/05/2022 5:40:49 AM	-1
01/05/2022 11:29:38 AM	-1
01/05/2022 7:04:57 PM	-1
01/06/2022 8:49:09 AM	0

Пример 5. Сценарий

Скрипт загрузки и результаты

Обзор

В этом примере выявлено, что вследствие ошибки оборудования изделия, изготовленные 5 января, являются дефектными. Конечному пользователю требуется объект диаграммы, который отображает по дате, какие изготовленные продукты были дефектными, а какие бездефектными, а также стоимость продуктов, изготовленных 5 января.

Откройте редактор загрузки данных и добавьте приведенный ниже скрипт загрузки на новую вкладку.

Скрипт загрузки содержит следующее:

- Набор данных, загруженный в таблицу, называется Products (Продукты).
- Данная таблица содержит следующие поля:
    - product ID (ИД продукта)
    - manufacture time (время изготовления)
    - cost price (себестоимость)

Скрипт загрузки

```
Products:
Load
*
Inline
[
product_id,manufacture_date,cost_price
9497,'01/01/2022 7:34:46 PM',13.24
9498,'01/01/2022 10:10:22 PM',31.43
9499,'01/02/2022 8:35:54 AM',36.34
9500,'01/03/2022 2:21:53 PM',51.75
9501,'01/04/2022 6:49:38 PM',15.35
9502,'01/04/2022 10:58:34 PM',74.34
9503,'01/05/2022 5:40:49 AM',73.53
9504,'01/05/2022 11:29:38 AM',50.00
9505,'01/05/2022 7:04:57 PM',47.25
9506,'01/06/2022 8:49:09 AM',74.23
];
```

Результаты

Загрузите данные и откройте лист. Создайте новую таблицу и добавьте это поле как измерение:

```
=dayname(manufacture_date)
```

Создайте следующие меры:

- `=if(only(InDay(manufacture_date,makedate(2022,01,05),0)),'Defective','Faultless')`
- `=sum(cost_price)`

Задайте параметру **Формат чисел** меры значение **Денежный**.

В области **Вид** выключите параметр **Итоги**.

Результирующая таблица

dayname (manufacture_date)	=if(only(InDay(manufacture_date,makedate (2022,01,05),0)),'Defective','Faultless')	=sum(cost_ price)
01/01/2022	Бездефектный	44.67
01/02/2022	Бездефектный	36.34
01/03/2022	Бездефектный	51.75

dayname (manufacture_date)	=if(only(InDay(manufacture_date,makedate (2022,01,05),0)),'Defective','Faultless')	=sum(cost_ price)
01/04/2022	Бездефектный	89.69
01/05/2022	Дефектный	170.78
01/06/2022	Бездефектный	74.23

Функция `inday()` возвращает логическое значение при проверки дат производства каждого продукта. Для любого изделия, изготовленного 5 января, функция `inday()` возвращает логическое значение «ИСТИНА» и ставит отметку «Дефектный». Любое изделие, которое возвращает значение «ЛОЖЬ», то есть изготовлено в другой день, отмечается как «Бездефектный».

indaytotime

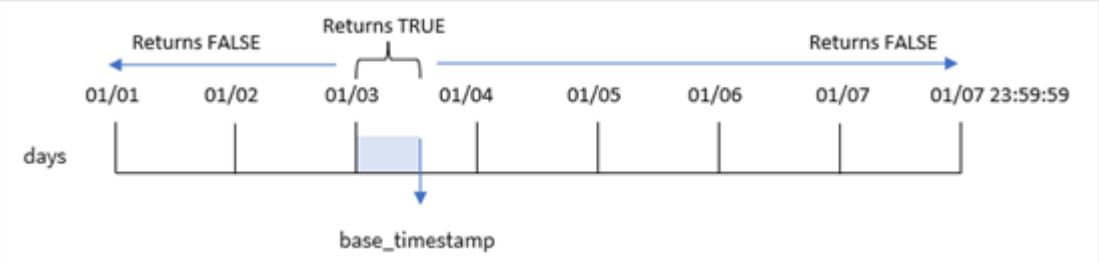
Эта функция возвращает значение True, если значение **timestamp** находится в пределах части дня, включающей значение, заданное в поле **base_timestamp** до определенной миллисекунды, указанной в поле **base_timestamp**, включительно.

Синтаксис:

```
InDayToTime (timestamp, base_timestamp, period_no[, day_start])
```

Функция `indaytotime()` возвращает результат в виде логического значения в зависимости от того, в какой момент сегмента дня попадает ли значение метки времени. Начальная граница этого сегмента совпадает с началом дня, который по умолчанию наступает в полночь; это можно изменить помощью аргумента `day_start` функции `indaytotime()`. Конечная граница сегмента дня определяется аргументом `base_timestamp` функции.

Схема функции `indaytotime`.



Когда это следует использовать

Функция `indaytotime()` возвращает результат в виде логического значения. Обычно этот тип функции используется в качестве условия в `if expression`. Функция `indaytotime()` возвращает агрегирование или расчет в зависимости от того, возникла ли метка времени в сегменте дня вплоть до момента базовой метки времени.

Например, функция `indaytotime()` может использоваться для отображения суммы, вырученной от продажи билетов на сеансы, состоявшиеся на данный момент текущего дня.

Возвращаемые типы данных: Булево значение

В Qlik Sense логическое значение «истина» представлено как -1, а «ложь» – как 0.

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`	(необязательно) Если необходимо работать с днями, которые начинаются не в полночь, задайте смещение в виде десятичного значения в `day_start`. Например 0,125, чтобы день начинался в 3 часа ночи (3 AM).

Региональные настройки

Если не указано иное, в примерах, приведенных в данном разделе, используется следующий формат даты: ММ/ДД/ГГГГ. Формат даты указан в операторе `SET DateFormat` скрипта загрузки данных. В вашей системе может быть установлен другой формат даты по умолчанию в зависимости от региональных настроек и других факторов. Можно изменить формат в примерах в соответствии с потребностями. Или можно изменить форматы в скрипте загрузки в соответствии с этими примерами.

Региональные настройки по умолчанию в приложениях основаны на системных региональных настройках компьютера или сервера, где установлено ПО Qlik Sense. Если на сервере Qlik Sense, к которому обращается пользователь, выбран шведский язык, то редактор загрузки данных будет использовать шведские региональные настройки для даты, времени и валюты. Эти параметры регионального формата не связаны с языком, отображаемым в интерфейсе пользователя Qlik Sense. Qlik Sense будет отображаться на языке, который используется в браузере.

Примеры функции

Пример	Результат
`indaytotime` ('01/12/2006 12:23:00 PM', '01/12/2006 11:59:00 PM', 0)	Возвращает True
`indaytotime` ('01/12/2006 12:23:00 PM', '01/12/2006 12:00:00 AM', 0)	Возвращает False
`indaytotime` '01/11/2006 12:23:00 PM', '01/12/2006 11:59:00 PM', -1)	Возвращает True

Пример 1. Без дополнительных аргументов

Скрипт загрузки и результаты

Обзор

Откройте Редактор загрузки данных и добавьте приведенный ниже скрипт загрузки на новую вкладку.

Скрипт загрузки содержит следующее:

- Набор данных, содержащий набор транзакций для периода с 4 по 5 января, загружается в таблицу Transactions.
- Поле данных, предоставленное в формате переменной (`M/D/YYYY h:mm:ss[.fff] TT`) системы `TimeStamp`.
- Предшествующий оператор load, содержащий функцию `indaytotime()`, которая задана как `'in_day_to_time'` — поле, определяющее, совершена ли каждая транзакция до 9:00 утра.

Скрипт загрузки

```
SET TimestampFormat='M/D/YYYY h:mm:ss[.fff] TT';

Transactions:
    Load
        *,
        indaytotime(date,'01/05/2022 9:00:00 AM',0) as in_day_to_time
        ;
Load
*
Inline
[
id,date,amount
8188,'01/04/2022 3:41:54 AM',25.66
8189,'01/04/2022 4:19:43 AM',87.21
8190,'01/04/2022 4:53:47 AM',53.80
8191,'01/04/2022 8:38:53 AM',69.98
8192,'01/04/2022 10:37:52 AM',57.42
8193,'01/04/2022 1:54:10 PM',45.89
8194,'01/04/2022 5:53:23 PM',82.77
8195,'01/04/2022 8:13:26 PM',36.23
8196,'01/04/2022 10:00:49 PM',76.11
8197,'01/05/2022 7:45:37 AM',82.06
8198,'01/05/2022 8:44:36 AM',17.17
8199,'01/05/2022 11:26:08 AM',40.39
8200,'01/05/2022 6:43:08 PM',37.23
8201,'01/05/2022 10:54:10 PM',88.27
8202,'01/05/2022 11:09:09 PM',95.93
];
```

Результаты

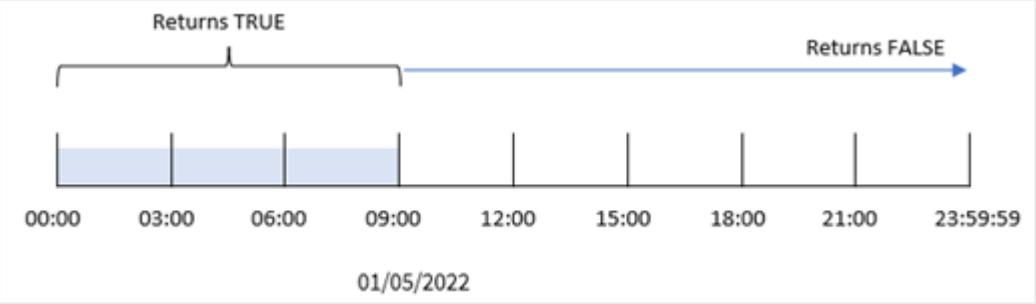
Загрузите данные и откройте лист. Создайте новую таблицу и добавьте эти поля как измерения:

- `date`
- `in_day_to_time`

Результирующая таблица

date	in_day_to_time
01/04/2022 3:41:54 AM	0
01/04/2022 4:19:43 AM	0
01/04/2022 04:53:47 AM	0
01/04/2022 8:38:53 AM	0
01/04/2022 10:37:52 AM	0
01/04/2022 1:54:10 PM	0
01/04/2022 5:53:23 PM	0
01/04/2022 8:13:26 PM	0
01/04/2022 10:00:49 PM	0
01/05/2022 7:45:37 AM	-1
01/05/2022 8:44:36 AM	-1
01/05/2022 11:26:08 AM	0
01/05/2022 6:43:08 PM	0
01/05/2022 10:54:10 PM	0
01/05/2022 11:09:09 PM	0

Пример 1. Схема функции `indaytotime` с ограничением 9:00



Поле `in_day_to_time field` создано предшествующим оператором load с использованием функции `indaytotime()`, где в качестве аргументов функции переданы поле даты, жестко запрограммированная метка времени для 9:00 утра 5 января и смещение 0. Любая транзакция, которая совершается с полуночи до 9:00 утра 5 января, возвращает TRUE.

Пример 2. Скрипт period_no

Скрипт загрузки и результаты

Обзор

В скрипте загрузки используются те же набор данных и сценарий, что и в первом примере.

Однако в этом примере вы рассчитаете, произошла ли транзакция за день до 5 января.

Скрипт загрузки

```
SET TimestampFormat='M/D/YYYY h:mm:ss[.fff] TT';

Transactions:
    Load
        *,
        indaytotime(date,'01/05/2022 9:00:00 AM', -1) as in_day_to_time
        ;
Load
*
Inline
[
id,date,amount
8188,'01/04/2022 3:41:54 AM',25.66
8189,'01/04/2022 4:19:43 AM',87.21
8190,'01/04/2022 4:53:47 AM',53.80
8191,'01/04/2022 8:38:53 AM',69.98
8192,'01/04/2022 10:37:52 AM',57.42
8193,'01/04/2022 1:54:10 PM',45.89
8194,'01/04/2022 5:53:23 PM',82.77
8195,'01/04/2022 8:13:26 PM',36.23
8196,'01/04/2022 10:00:49 PM',76.11
8197,'01/05/2022 7:45:37 AM',82.06
8198,'01/05/2022 8:44:36 AM',17.17
8199,'01/05/2022 11:26:08 AM',40.39
8200,'01/05/2022 6:43:08 PM',37.23
8201,'01/05/2022 10:54:10 PM',88.27
8202,'01/05/2022 11:09:09 PM',95.93
];
```

Результаты

Загрузите данные и откройте лист. Создайте новую таблицу и добавьте эти поля как измерения:

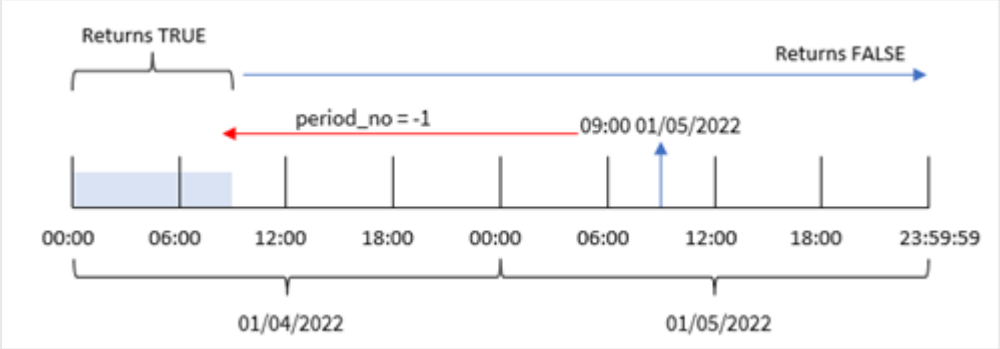
- date
- in_day_to_time

Результирующая таблица

date	in_day_to_time
01/04/2022 3:41:54 AM	-1
01/04/2022 4:19:43 AM	-1
01/04/2022 04:53:47 AM	-1
01/04/2022 8:38:53 AM	-1
01/04/2022 10:37:52 AM	0
01/04/2022 1:54:10 PM	0

date	in_day_to_time
01/04/2022 5:53:23 PM	0
01/04/2022 8:13:26 PM	0
01/04/2022 10:00:49 PM	0
01/05/2022 7:45:37 AM	0
01/05/2022 8:44:36 AM	0
01/05/2022 11:26:08 AM	0
01/05/2022 6:43:08 PM	0
01/05/2022 10:54:10 PM	0
01/05/2022 11:09:09 PM	0

Пример 2. Схема функции `indaytotime` с транзакциями за 4 января.



В этом примере в качестве аргумента смещения в функции `indaytotime()` использовалось значение -1, поэтому функция определяет, произошла ли транзакция 4 января. Это можно проверить в таблице выходных данных, где одна транзакция возвращает результат в виде логического значения TRUE.

Пример 3. Скрипт day_start

Скрипт загрузки и результаты

Обзор

Используется тот же набор данных и сценарий, что в первом примере.

Однако в этом примере используется политика компании, согласно которой рабочий день начинается и заканчивается в 8:00 утра.

Скрипт загрузки

```
SET TimestampFormat='M/D/YYYY h:mm:ss[.fff] TT';

Transactions:
    Load
```

```
        *,
        indaytotime(date,'01/05/2022 9:00:00 AM', 0,8/24) as in_day_to_time
        ;
Load
*
Inline
[
id,date,amount
8188,'01/04/2022 3:41:54 AM',25.66
8189,'01/04/2022 4:19:43 AM',87.21
8190,'01/04/2022 4:53:47 AM',53.80
8191,'01/04/2022 8:38:53 AM',69.98
8192,'01/04/2022 10:37:52 AM',57.42
8193,'01/04/2022 1:54:10 PM',45.89
8194,'01/04/2022 5:53:23 PM',82.77
8195,'01/04/2022 8:13:26 PM',36.23
8196,'01/04/2022 10:00:49 PM',76.11
8197,'01/05/2022 7:45:37 AM',82.06
8198,'01/05/2022 8:44:36 AM',17.17
8199,'01/05/2022 11:26:08 AM',40.39
8200,'01/05/2022 6:43:08 PM',37.23
8201,'01/05/2022 10:54:10 PM',88.27
8202,'01/05/2022 11:09:09 PM',95.93
];
```

Результаты

Загрузите данные и откройте лист. Создайте новую таблицу и добавьте эти поля как измерения:

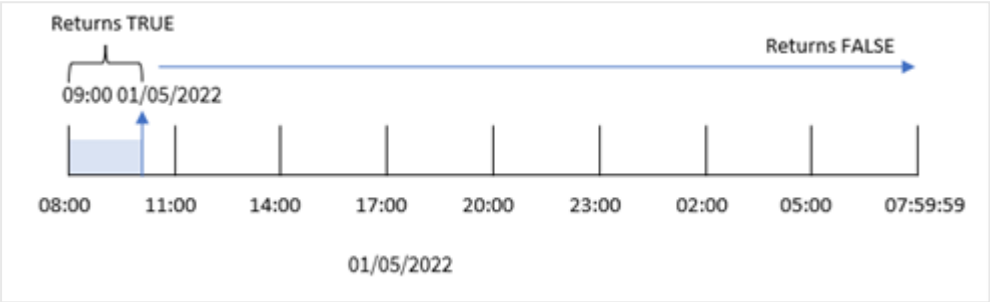
- date
- in_day_to_time

Результирующая таблица

date	in_day_to_time
01/04/2022 3:41:54 AM	0
01/04/2022 4:19:43 AM	0
01/04/2022 04:53:47 AM	0
01/04/2022 8:38:53 AM	0
01/04/2022 10:37:52 AM	0
01/04/2022 1:54:10 PM	0
01/04/2022 5:53:23 PM	0
01/04/2022 8:13:26 PM	0
01/04/2022 10:00:49 PM	0
01/05/2022 7:45:37 AM	0
01/05/2022 8:44:36 AM	-1

date	in_day_to_time
01/05/2022 11:26:08 AM	0
01/05/2022 6:43:08 PM	0
01/05/2022 10:54:10 PM	0
01/05/2022 11:09:09 PM	0

Пример 3. Схема функции `indaytotime` с транзакциями с 8:00 до 9:00 утра.



Так как аргумент `start_day` = 8/24, который соответствует 8:00 утра, используется в функции `indaytotime()`, каждый день начинается и заканчивается в 8:00 утра. Поэтому функция `indaytotime()` будет возвращать логическое значение «ИСТИНА» для любой транзакции, которая имела место в период с 8:00 до 9:00 5 января.

Пример 4. Объект диаграммы

Скрипт загрузки и выражение диаграммы

Обзор

Используется тот же набор данных и сценарий, что в первом примере.

Однако в этом примере в приложение загружается неизмененный набор данных. Выполнив расчет, вы определите, осуществлена ли транзакция до 9:00 утра 5 января, создав меру в объекте диаграммы.

Скрипт загрузки

```
Transactions:
Load
*
Inline
[
id,date,amount
8188,'01/04/2022 3:41:54 AM',25.66
8189,'01/04/2022 4:19:43 AM',87.21
8190,'01/04/2022 4:53:47 AM',53.80
8191,'01/04/2022 8:38:53 AM',69.98
8192,'01/04/2022 10:37:52 AM',57.42
8193,'01/04/2022 1:54:10 PM',45.89
```

```
8194,'01/04/2022 5:53:23 PM',82.77
8195,'01/04/2022 8:13:26 PM',36.23
8196,'01/04/2022 10:00:49 PM',76.11
8197,'01/05/2022 7:45:37 AM',82.06
8198,'01/05/2022 8:44:36 AM',17.17
8199,'01/05/2022 11:26:08 AM',40.39
8200,'01/05/2022 6:43:08 PM',37.23
8201,'01/05/2022 10:54:10 PM',88.27
8202,'01/05/2022 11:09:09 PM',95.93
];
```

Результаты

Загрузите данные и откройте лист. Создайте новую таблицу и добавьте это поле как измерение:

`date`.

Чтобы определить, совершена ли транзакция до 9:00 утра 5 января, создайте следующую меру:

```
=indaytotime(date,'01/05/2022 9:00:00 AM',0)
```

Результирующая таблица

date	=indaytotime(date,'01/05/2022 9:00:00 AM',0)
01/04/2022 3:41:54 AM	0
01/04/2022 4:19:43 AM	0
01/04/2022 04:53:47 AM	0
01/04/2022 8:38:53 AM	0
01/04/2022 10:37:52 AM	0
01/04/2022 1:54:10 PM	0
01/04/2022 5:53:23 PM	0
01/04/2022 8:13:26 PM	0
01/04/2022 10:00:49 PM	0
01/05/2022 7:45:37 AM	-1
01/05/2022 8:44:36 AM	-1
01/05/2022 11:26:08 AM	0
01/05/2022 6:43:08 PM	0
01/05/2022 10:54:10 PM	0
01/05/2022 11:09:09 PM	0

Мера `in_day_to_time` создана в объекте диаграммы с использованием функции `indaytotime()`, где в качестве аргументов функции переданы поле даты, жестко запрограммированная метка времени для 9:00 утра 5 января и смещение `0`. Любая транзакция, которая совершается с полуночи до 9:00 утра 5 января, возвращает TRUE. Это можно проверить в таблице результатов.

Пример 5. Сценарий

Скрипт загрузки и результаты

Обзор

В этом примере набор данных, содержащий продажи билетов для местного кинотеатра, загружается в таблицу Ticket_Sales. Сегодня 3 мая 2022 года, сейчас 11:00 утра.

Пользователь хочет, чтобы объект диаграммы ключевого показателя эффективности отображал доход, полученный от всех сеансов, которые сегодня уже состоялись на текущий момент.

Скрипт загрузки

```
SET TimestampFormat='M/D/YYYY h:mm:ss[.fff] TT';

Ticket_Sales:
Load
*
Inline
[
sale ID, show time, ticket price
1,05/01/2022 09:30:00 AM,10.50
2,05/03/2022 05:30:00 PM,21.00
3,05/03/2022 09:30:00 AM,10.50
4,05/03/2022 09:30:00 AM,31.50
5,05/03/2022 09:30:00 AM,10.50
6,05/03/2022 12:00:00 PM,42.00
7,05/03/2022 12:00:00 PM,10.50
8,05/03/2022 05:30:00 PM,42.00
9,05/03/2022 08:00:00 PM,31.50
10,05/04/2022 10:30:00 AM,31.50
11,05/04/2022 12:00:00 PM,10.50
12,05/04/2022 05:30:00 PM,10.50
13,05/05/2022 05:30:00 PM,21.00
14,05/06/2022 12:00:00 PM,21.00
15,05/07/2022 09:30:00 AM,42.00
16,05/07/2022 10:30:00 AM,42.00
17,05/07/2022 10:30:00 AM,10.50
18,05/07/2022 05:30:00 PM,10.50
19,05/08/2022 05:30:00 PM,21.00
20,05/11/2022 09:30:00 AM,10.50
];
```

Результаты

Выполните следующие действия.

1. Создайте объект ключевого показателя эффективности.
2. Создайте меру, которая будет отображать сумму всех продаж билетов на сеансы, которые состоялись сегодня на данный момент, с использованием функции `indaytotime()`:

```
=sum(if(indaytotime([show time],'05/03/2022 11:00:00 AM',0),[ticket price],0))
```

3. Создайте метку для объекта ключевого показателя эффективности – «Текущий доход».
4. Задайте параметру **Формат чисел** меры значение **Денежный**.

Общая сумма продажи билетов за период до 11:00 утра 3 мая 2022 года составила $52,50.

Функция `indaytotime ()` возвращает логическое значение при сравнении времени сеансов для каждой продажи билетов с текущим временем ('05/03/2022 11:00:00 AM'). Для любого сеанса 3 мая до 11:00 утра функция `indaytotime ()` возвращает логическое значение «ИСТИНА», и цена билета будет включена в итоговую сумму.

inlunarweek

Эта функция определяет, находится ли значение **timestamp** в пределах лунной недели, включающей значение, указанное в поле **base_date**. При определении лунных недель в Qlik Sense первым днем первой недели считается 1 января. Все недели, кроме последней, будут содержать ровно 7 дней.

Синтаксис:

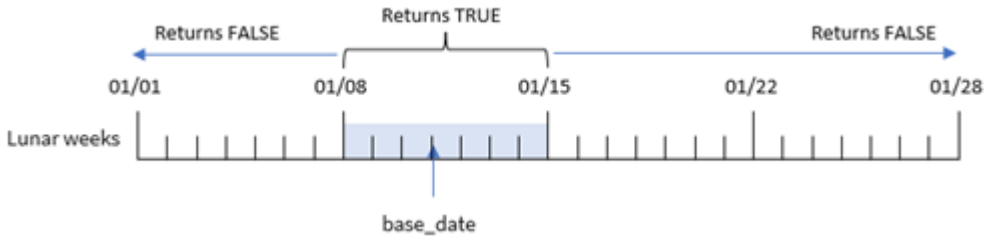
```
InLunarWeek (timestamp, base_date, period_no[, first_week_day])
```

Возвращаемые типы данных: Булево значение

> *В Qlik Sense логическое значение «истина» представлено как -1, а «ложь» – как 0.*

Функция `inlunarweek()` определяет, на какую лунную неделю выпадает `base_date`. Затем она возвращает булев результат, когда установлено, что каждая метка времени относится к той же лунной неделе, что `base_date`.

Схема функции `inlunarweek()`



Когда это следует использовать

Функция `inlunarweek()` возвращает результат в виде булева значения. Обычно этот тип функции используется в качестве условия в выражении IF. Она возвращает агрегирование или вычисление в зависимости от того, попадает ли проверяемая дата в рассматриваемую лунную неделю.

Например, функцию `inlunarweek()` можно использовать для идентификации всего оборудования, изготовленного в определенную лунную неделю.

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 функции

Пример	Результат
`inlunarweek ('01/12/2013', '01/14/2013', 0)`	Возвращает `TRUE`, поскольку значение `timestamp`, 01/12/2013, выпадает на неделю с 01/08/2013 по 01/14/2013.
`inlunarweek ('01/12/2013', '01/07/2013', 0)`	Возвращает `FALSE`, так как `base_date` 01/07/2013 относится к лунной неделе с 01/01/2013 по 01/07/2013.
`inlunarweek ('01/12/2013', '01/14/2013', -1)`	Возвращает `FALSE`. При указании значения period_no = -1 происходит сдвиг на одну неделю назад с 01/01/2013 по 01/07/2013.
`inlunarweek ('01/07/2013', 01/14/2013', -1)`	Возвращает `TRUE`. По сравнению с предыдущим примером `timestamp` выпадает на следующую неделю с учетом сдвига назад
`inlunarweek ('01/11/2006', '01/08/2006', 0, 3)`	Возвращает `FALSE`. При указании значения 3 для `first_week_day` начало года отсчитывается от 01/04/2013. Таким образом, значение `base_`date выпадает на первую неделю, а значение `timestamp` на неделю с 01/11/2013 по 01/17/2013.

Функция `inlunarweek()` часто используется в сочетании со следующими функциями:

Связанные функции

Функция	Взаимодействие
lunarweekname (page 872)	Эта функция позволяет определить, к какой по порядку лунной неделе года относится входная дата.

Региональные настройки

Если не указано иное, в примерах, приведенных в данном разделе, используется следующий формат даты: ММ/ДД/ГГГГ. Формат даты указан в операторе `SET DateFormat` скрипта загрузки данных. В вашей системе может быть установлен другой формат даты по умолчанию в

зависимости от региональных настроек и других факторов. Можно изменить формат в примерах в соответствии с потребностями. Или можно изменить форматы в скрипте загрузки в соответствии с этими примерами.

Региональные настройки по умолчанию в приложениях основаны на системных региональных настройках компьютера или сервера, где установлено ПО Qlik Sense. Если на сервере Qlik Sense, к которому обращается пользователь, выбран шведский язык, то редактор загрузки данных будет использовать шведские региональные настройки для даты, времени и валюты. Эти параметры регионального формата не связаны с языком, отображаемым в интерфейсе пользователя Qlik Sense. Qlik Sense будет отображаться на языке, который используется в браузере.

Пример 1. Без дополнительных аргументов

Скрипт загрузки и результаты

Обзор

Откройте редактор загрузки данных и добавьте приведенный ниже скрипт загрузки на новую вкладку.

Скрипт загрузки содержит следующее:

- Набор данных с транзакциями за январь, загруженный в таблицу под именем `Transactions`.
- Поле даты было предоставлено в формате системной переменной `DateFormat` (MM/DD/YYYY).

Создайте поле `in_lunar_week`, которое определяет, совершены ли транзакции в ту же лунную неделю, к которой относится 10 января.

Скрипт загрузки

```
SET DateFormat='MM/DD/YYYY';

Transactions:
    Load
        *,
        inlunarweek(date,'01/10/2022', 0) as in_lunar_week
        ;
Load
*
Inline
[
id,date,amount
8183,'1/5/2022',42.32
8184,'1/6/2022',68.22
8185,'1/7/2022',15.25
8186,'1/8/2022',25.26
8187,'1/9/2022',37.23
8188,'1/10/2022',37.23
8189,'1/11/2022',17.17
8190,'1/12/2022',88.27
```

```
8191,'1/13/2022',57.42
8192,'1/14/2022',53.80
8193,'1/15/2022',82.06
8194,'1/16/2022',87.21
8195,'1/17/2022',95.93
8196,'1/18/2022',45.89
8197,'1/19/2022',36.23
8198,'1/20/2022',25.66
8199,'1/21/2022',82.77
8200,'1/22/2022',69.98
8201,'1/23/2022',76.11
];
```

Результаты

Загрузите данные и откройте лист. Создайте новую таблицу и добавьте эти поля как измерения:

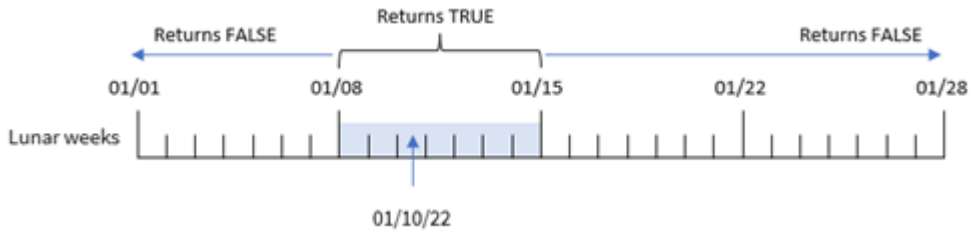
- `date`
- `in_lunar_week`

Результирующая таблица

date	in_lunar_week
1/5/2022	0
1/6/2022	0
1/7/2022	0
1/8/2022	-1
1/9/2022	-1
1/10/2022	-1
1/11/2022	-1
1/12/2022	-1
1/13/2022	-1
1/14/2022	-1
1/15/2022	0
1/16/2022	0
1/17/2022	0
1/18/2022	0
1/19/2022	0
1/20/2022	0
1/21/2022	0

date	in_lunar_week
1/22/2022	0
1/23/2022	0

Функция `inlunarweek()`, базовый пример



Поле `in_lunar_week` создано предшествующим оператором load с использованием функции `inlunarweek()`, для который переданы следующие аргументы:

- Поле `date`
- Жестко закодированная дата, 10 января, в качестве `base_date`
- `period_no` = 0

Так как лунные недели начинаются с 1 января, 10 января попадает в лунную неделю, которая начинается 8 января и заканчивается 14 января. Поэтому все транзакции, которые совершаются между этими числами января будут возвращать булево значение `TRUE`. Это можно проверить в таблице результатов.

Пример 2. Скрипт period_no

Примеры и результаты:

Обзор

Откройте редактор загрузки данных и добавьте приведенный ниже скрипт загрузки на новую вкладку.

Скрипт загрузки содержит следующее:

- Используется тот же набор данных и сценарий, что в первом примере.
- Поле даты было предоставлено в формате системной переменной `DateFormat` (MM/DD/YYYY).

Однако в этом примере стоит задача создать поле `2_lunar_weeks_later`, которое определяет, совершены ли транзакции через две лунные недели после 10 января.

Скрипт загрузки

```
SET DateFormat='MM/DD/YYYY';
```

```
Transactions:
    Load
        *,
        inlunarweek(date,'01/10/2022', 2) as [2_lunar_weeks_later]
        ;
Load
*
Inline
[
id,date,amount
8183,'1/5/2022',42.32
8184,'1/6/2022',68.22
8185,'1/7/2022',15.25
8186,'1/8/2022',25.26
8187,'1/9/2022',37.23
8188,'1/10/2022',37.23
8189,'1/11/2022',17.17
8190,'1/12/2022',88.27
8191,'1/13/2022',57.42
8192,'1/14/2022',53.80
8193,'1/15/2022',82.06
8194,'1/16/2022',87.21
8195,'1/17/2022',95.93
8196,'1/18/2022',45.89
8197,'1/19/2022',36.23
8198,'1/20/2022',25.66
8199,'1/21/2022',82.77
8200,'1/22/2022',69.98
8201,'1/23/2022',76.11
];
```

Результаты

Загрузите данные и откройте лист. Создайте новую таблицу и добавьте эти поля как измерения:

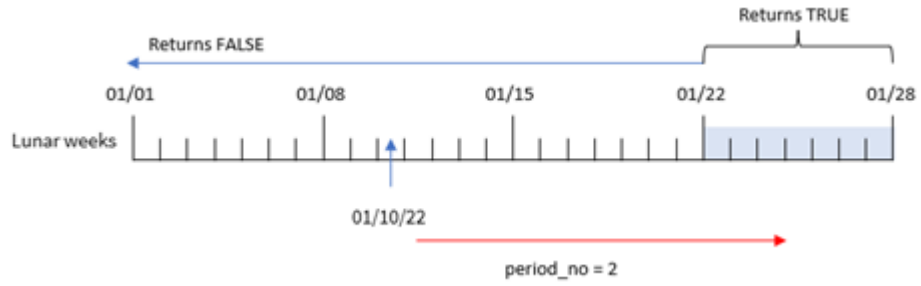
- `date`
- `2_lunar_weeks_later`

Результирующая таблица

date	2_lunar_weeks_later
1/5/2022	0
1/6/2022	0
1/7/2022	0
1/8/2022	0
1/9/2022	0
1/10/2022	0
1/11/2022	0

date	2_lunar_weeks_later
1/12/2022	0
1/13/2022	0
1/14/2022	0
1/15/2022	0
1/16/2022	0
1/17/2022	0
1/18/2022	0
1/19/2022	0
1/20/2022	0
1/21/2022	0
1/22/2022	-1
1/23/2022	-1

Функция `inlunarweek()`, пример с аргументом period_no



Так как в этом примере в качестве аргумента сдвига используется `period_no` = 2, функция `inlunarweek()` определяет неделю, которая начинается 22 января, как лунную неделю, для которой проверяются транзакции. Таким образом, любая транзакция, совершенная в период с 22 по 28 января, возвращает булев результат `TRUE`.

Пример 3. Аргумент first_week_day

Скрипт загрузки и результаты

Обзор

Откройте редактор загрузки данных и добавьте приведенный ниже скрипт загрузки на новую вкладку.

Скрипт загрузки использует тот же набор данных и сценарий, что в первом примере. Однако в этом примере лунные недели начинаются с 6 января.

- Используется тот же набор данных и сценарий, что в первом примере.
- Используется системная переменная `DateFormat` со значением по умолчанию MM/DD/YYYY.
- Аргумент `first_week_day` имеет значение 5. Это задает в качестве начала лунных недель 5 января.

Скрипт загрузки

```
SET DateFormat='MM/DD/YYYY';

Transactions:
    Load
        *,
        inlunarweek(date,'01/10/2022', 0,5) as in_lunar_week
        ;
Load
*
Inline
[
id,date,amount
8183,'1/5/2022',42.32
8184,'1/6/2022',68.22
8185,'1/7/2022',15.25
8186,'1/8/2022',25.26
8187,'1/9/2022',37.23
8188,'1/10/2022',37.23
8189,'1/11/2022',17.17
8190,'1/12/2022',88.27
8191,'1/13/2022',57.42
8192,'1/14/2022',53.80
8193,'1/15/2022',82.06
8194,'1/16/2022',87.21
8195,'1/17/2022',95.93
8196,'1/18/2022',45.89
8197,'1/19/2022',36.23
8198,'1/20/2022',25.66
8199,'1/21/2022',82.77
8200,'1/22/2022',69.98
8201,'1/23/2022',76.11
];
```

Результаты

Загрузите данные и откройте лист. Создайте новую таблицу и добавьте эти поля как измерения:

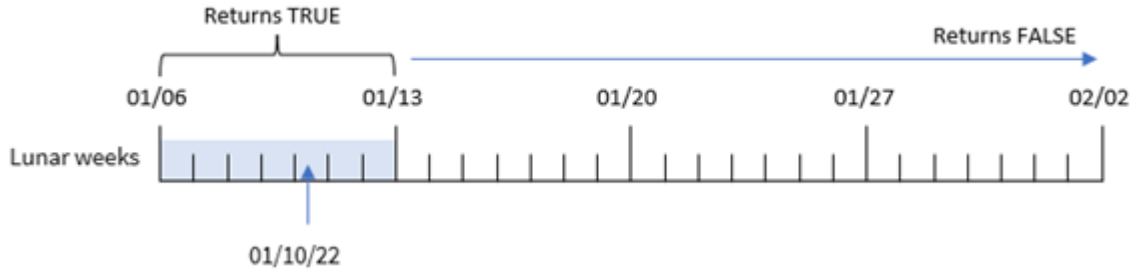
- `date`
- `in_lunar_week`

Результирующая таблица

date	in_lunar_week
1/5/2022	0

date	in_lunar_week
1/6/2022	-1
1/7/2022	-1
1/8/2022	-1
1/9/2022	-1
1/10/2022	-1
1/11/2022	-1
1/12/2022	-1
1/13/2022	0
1/14/2022	0
1/15/2022	0
1/16/2022	0
1/17/2022	0
1/18/2022	0
1/19/2022	0
1/20/2022	0
1/21/2022	0
1/22/2022	0
1/23/2022	0

Функция `inlunarweek()` , пример с аргументом first_week_day



В этом примере используется аргумент `first_week_date` = 5 в функции `inlunarweek()`, поэтому начало календаря лунных недель сдвигается на 6 января. Таким образом, 10 января выпадает на лунную неделю, которая начинается 6 января и заканчивается 12 января. Любая транзакция, которая совершается между этими датами, возвращает булево значение `TRUE`.

Пример 4. Объект диаграммы

Скрипт загрузки и выражение диаграммы.

Обзор

Откройте редактор загрузки данных и добавьте приведенный ниже скрипт загрузки на новую вкладку.

Скрипт загрузки содержит следующее:

- Используется тот же набор данных и сценарий, что в первом примере.
- Поле даты было предоставлено в формате системной переменной `DateFormat` (MM/DD/YYYY).

Однако в этом примере в приложение загружается неизмененный набор данных. Вычисление, которое определяет, совершены ли транзакции в ту же лунную неделю, к которой относится 10 января, создается как мера в объекте диаграммы в приложении.

Скрипт загрузки

```
SET DateFormat='MM/DD/YYYY';

Transactions:
Load
*
Inline
[
id,date,amount
8183,'1/5/2022',42.32
8184,'1/6/2022',68.22
8185,'1/7/2022',15.25
8186,'1/8/2022',25.26
8187,'1/9/2022',37.23
8188,'1/10/2022',37.23
8189,'1/11/2022',17.17
8190,'1/12/2022',88.27
8191,'1/13/2022',57.42
8192,'1/14/2022',53.80
8193,'1/15/2022',82.06
8194,'1/16/2022',87.21
8195,'1/17/2022',95.93
8196,'1/18/2022',45.89
8197,'1/19/2022',36.23
8198,'1/20/2022',25.66
8199,'1/21/2022',82.77
8200,'1/22/2022',69.98
8201,'1/23/2022',76.11
];
```

Результаты

Загрузите данные и откройте лист. Создайте новую таблицу и добавьте это поле как измерение: `date`.

Чтобы рассчитать, совершена ли транзакция в течение той же лунной недели, к которой относится 10 января, создайте следующую меру:

```
= inlunarweek(date,'01/10/2022', 0)
```

Результирующая таблица

date	=inlunarweek(date,'01/10/2022', 0)
1/5/2022	0
1/6/2022	0
1/7/2022	0
1/8/2022	-1
1/9/2022	-1
1/10/2022	-1
1/11/2022	-1
1/12/2022	-1
1/13/2022	-1
1/14/2022	-1
1/15/2022	0
1/16/2022	0
1/17/2022	0
1/18/2022	0
1/19/2022	0
1/20/2022	0
1/21/2022	0
1/22/2022	0
1/23/2022	0

Пример 5. Сценарий

Скрипт загрузки и выражение диаграммы.

Обзор

Откройте редактор загрузки данных и добавьте приведенный ниже скрипт загрузки на новую вкладку.

Скрипт загрузки содержит следующее:

- Набор данных, загруженный в таблицу, называется `Products`
- Информация, содержащая идентификатор продукта, дату изготовления и себестоимость.

Установлено, что вследствие ошибки оборудования продукты, изготовленные в течение лунной недели, на которую выпадает 12 января, являются дефектными. Конечному пользователю требуется объект диаграммы, который отображает по имени лунной недели, какие изготовленные продукты были дефектными, а какие бездефектными, а также стоимость продуктов, изготовленных в этом месяце.

Скрипт загрузки

```
SET DateFormat='MM/DD/YYYY';

Transactions:
Load
*
Inline
[
product_id,manufacture_date,cost_price
8183,'1/5/2022',42.32
8184,'1/6/2022',68.22
8185,'1/7/2022',15.25
8186,'1/8/2022',25.26
8187,'1/9/2022',37.23
8188,'1/10/2022',37.23
8189,'1/11/2022',17.17
8190,'1/12/2022',88.27
8191,'1/13/2022',57.42
8192,'1/14/2022',53.80
8193,'1/15/2022',82.06
8194,'1/16/2022',87.21
8195,'1/17/2022',95.93
8196,'1/18/2022',45.89
8197,'1/19/2022',36.23
8198,'1/20/2022',25.66
8199,'1/21/2022',82.77
8200,'1/22/2022',69.98
8201,'1/23/2022',76.11
];
```

Результаты

Выполните следующие действия.

1. Загрузите данные и откройте лист. Создайте новую таблицу.
2. Создайте измерение для отображения названий месяцев:
   `=lunarweekname(manufacture_date)`
3. Создайте меру, чтобы определить, какие из продуктов дефектные, а какие нет, с использованием функции `inlunarweek()`:
   `=if(only(inlunarweek(manufacture_date,makedate(2022,01,12),0)), 'Defective','Faultless')`
4. Создайте меру для суммирования `cost_price` продуктов:
   `=sum(cost_price)`
5. Задайте параметру меры **Формат чисел** значение **Денежный**.
6. В области **Вид** выключите параметр **Итоги**.

Результирующая таблица

lunarweekname (manufacture_date)	=if(only(inlunarweek(manufacture_date,makedate (2022,01,12),0)), 'Defective','Faultless')	sum(cost_ price)
2022/01	Бездефектный	$125.79
2022/02	Дефектный	$316.38
2022/03	Бездефектный	$455.75
2022/04	Бездефектный	$146.09

Функция `inlunarweek()` возвращает булево значение при проверке дат производства каждого продукта. Для любого продукта, изготовленного в течение лунной недели, к которой относится 10 января, функция `inlunarweek()` возвращает булево значение `TRUE` и ставит отметку «Дефектный». Любой продукт, который возвращает значение `FALSE`, то есть изготовлен в другую неделю, отмечается как «Бездефектный».

inlunarweektodate

Эта функция определяет, находится ли значение **timestamp** в пределах части лунной недели до последней миллисекунды, указанной в поле **base_date**, включительно. При определении лунных недель в Qlik Sense первым днем первой недели считается 1 января. Все недели, кроме последней будут содержать ровно 7 дней.

Синтаксис:

```
InLunarWeekToDate (timestamp, base_date, period_no [, first_week_day])
```

Возвращаемые типы данных: Булево значение

> *В Qlik Sense логическое значение «истина» представлено как -1, а «ложь» — как 0.*

Диаграмма с примером функции `inlunarweektodate()`

Функция `inlunarweektodate()` выступает в роли конечной точки лунной недели. В противоположность этому, функция `inlunarweek()` определяет, на какую лунную неделю выпадает `base_date`. Например, если `base_date` = 5 января, любая метка времени с 1 по 5 января может возвращать булев результат `TRUE`, а 6, 7 января и более поздние метки времени будут возвращать булев результат `FALSE`.

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.

Когда это следует использовать

Функция `inlunarweektodate()` возвращает результат в виде булева значения. Обычно этот тип функции используется в качестве условия в выражении IF. Функция `inlunarweektodate()` используется, когда пользователю требуется произвести расчеты с целью агрегирования или вычисления в зависимости от того, относится ли рассматриваемая дата к определенному отрезку рассматриваемой недели.

Например, функцию `inlunarweektodate()` можно использовать для идентификации всего оборудования, изготовленного в определенную неделю: с начала недели и до определенной даты включительно.

Примеры функции

Пример	Результат
`inlunarweektodate ('01/12/2013', '01/13/2013', 0)`	Возвращает `TRUE`, поскольку значение `timestamp`, 01/12/2013, выпадает на часть недели с 01/08/2013 по 01/13/2013.

Пример	Результат
`inlunarweektodate ('01/12/2013', '01/11/2013', 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 01/12/2012.
`inlunarweektodate ('01/12/2006', '01/05/2006', 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.

Функция `inlunarweektodate()` часто используется в сочетании со следующими функциями:

Связанные функции

Функция	Взаимодействие
lunarweekname (page 872)	Эта функция позволяет определить, к какой по порядку лунной неделе года относится входная дата.

Региональные настройки

Если не указано иное, в примерах, приведенных в данном разделе, используется следующий формат даты: MM/ДД/ГГГГ. Формат даты указан в операторе `SET DateFormat` скрипта загрузки данных. В вашей системе может быть установлен другой формат даты по умолчанию в зависимости от региональных настроек и других факторов. Можно изменить формат в примерах в соответствии с потребностями. Или можно изменить форматы в скрипте загрузки в соответствии с этими примерами.

Региональные настройки по умолчанию в приложениях основаны на системных региональных настройках компьютера или сервера, где установлено ПО Qlik Sense. Если на сервере Qlik Sense, к которому обращается пользователь, выбран шведский язык, то редактор загрузки данных будет использовать шведские региональные настройки для даты, времени и валюты. Эти параметры регионального формата не связаны с языком, отображаемым в интерфейсе пользователя Qlik Sense. Qlik Sense будет отображаться на языке, который используется в браузере.

Пример 1. Без дополнительных аргументов

Скрипт загрузки и результаты

Обзор

Откройте редактор загрузки данных и добавьте приведенный ниже скрипт загрузки на новую вкладку.

Скрипт загрузки содержит следующее:

- Набор данных, содержащий набор транзакций за январь, загруженный в таблицу под именем `Transactions`. Используется системная переменная `DateFormat` со значением по умолчанию MM/DD/YYYY.

- Создайте поле in_lunar_week_to_date, которое определяет, какие транзакции совершены в ту же лунную неделю, к которой относится 10 января.

Скрипт загрузки

```
SET DateFormat='MM/DD/YYYY';

Transactions:
    Load
        *,
        inlunarweektodate(date,'01/10/2022', 0) as in_lunar_week_to_date
        ;
Load
*
Inline
[
id,date,amount
8188,'1/10/2022',37.23
8189,'1/17/2022',17.17
8190,'1/26/2022',88.27
8191,'1/12/2022',57.42
8192,'1/19/2022',53.80
8193,'1/21/2022',82.06
8194,'1/1/2022',40.39
8195,'1/27/2022',87.21
8196,'1/11/2022',95.93
8197,'1/29/2022',45.89
8198,'1/31/2022',36.23
8199,'1/18/2022',25.66
8200,'1/23/2022',82.77
8201,'1/15/2022',69.98
8202,'1/4/2022',76.11
];
```

Результаты

Загрузите данные и откройте лист. Создайте новую таблицу и добавьте эти поля как измерения:

- date
- in_lunar_week_to_date

Результирующая таблица

date	in_lunar_week_to_date
1/1/2022	0
1/4/2022	0
1/10/2022	-1
1/11/2022	0
1/12/2022	0

date	in_lunar_week_to_date
1/15/2022	0
1/17/2022	0
1/18/2022	0
1/19/2022	0
1/21/2022	0
1/23/2022	0
1/26/2022	0
1/27/2022	0
1/29/2022	0
1/31/2022	0

Функция `inlunarweektodate()`, без дополнительных аргументов

Поле `in_lunar_week_to_date` создано предшествующим оператором load с использованием функции `inlunarweektodate()`, где в качестве аргументов функции переданы поле `date`, жестко запрограммированная дата 10 января в качестве `base_date` и сдвиг 0.

Так как лунная неделя начинается 1 января, 10 января относится к лунной неделе, которая начинается 8 января; и поскольку используется функция `inlunarweektodate()`, эта лунная неделя заканчивается 10 января. Поэтому все транзакции, которые совершаются между этими числами января будут возвращать булево значение `TRUE`. Это можно проверить в таблице результатов.

Пример 2. Скрипт period_no

Скрипт загрузки и результаты

Обзор

Откройте редактор загрузки данных и добавьте приведенный ниже скрипт загрузки на новую вкладку.

Скрипт загрузки содержит тот же набор данных и сценарий, что в первом примере. Однако в этом примере стоит задача создать поле `2_lunar_weeks_later`, которое определяет, совершены ли транзакции через две недели после лунной недели, которая началась 1 января.

Скрипт загрузки

```
SET DateFormat='MM/DD/YYYY';
Transactions:
    Load
        *,
        inlunarweektodate(date,'01/10/2022', 2) as [2_lunar_weeks_later]
        ;
Load
*
Inline
[
id,date,amount
8188,'1/10/2022',37.23
8189,'1/17/2022',17.17
8190,'1/26/2022',88.27
8191,'1/12/2022',57.42
8192,'1/19/2022',53.80
8193,'1/21/2022',82.06
8194,'1/1/2022',40.39
8195,'1/27/2022',87.21
8196,'1/11/2022',95.93
8197,'1/29/2022',45.89
8198,'1/31/2022',36.23
8199,'1/18/2022',25.66
8200,'1/23/2022',82.77
8201,'1/15/2022',69.98
8202,'1/4/2022',76.11
];
```

Результаты

Загрузите данные и откройте лист. Создайте новую таблицу и добавьте эти поля как измерения:

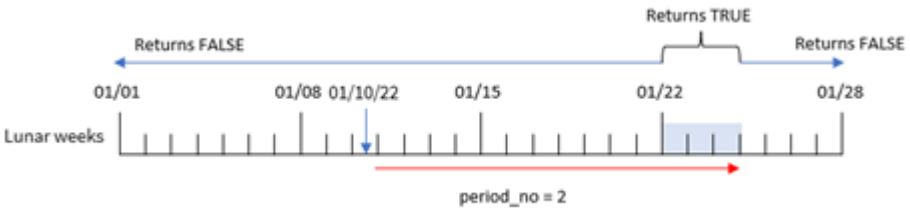
- date
- 2_lunar_weeks_later

Результирующая таблица

date	2_lunar_weeks_later
1/1/2022	0
1/4/2022	0
1/10/2022	0
1/11/2022	0
1/12/2022	0
1/15/2022	0
1/17/2022	0

date	2_lunar_weeks_later
1/18/2022	0
1/19/2022	0
1/21/2022	0
1/23/2022	-1
1/26/2022	0
1/27/2022	0
1/29/2022	0
1/31/2022	0

Функция `inlunarweektodate()`, пример с аргументом period_no



В данном примере функция `inlunarweektodate()` определяет, что лунная неделя до 10 января содержит три дня (8, 9 и 10 января). Так как в качестве аргумента сдвига использовался `period_no` = 2, эта лунная неделя сдвигается на 14 дней. Поэтому создается определение трехдневной недели, которая включает 22, 23 и 24 января. Любая транзакция, совершенная в период с 22 по 24 января, возвращает булев результат `TRUE`.

Пример 3. Аргумент first_week_day

Скрипт загрузки и результаты

Обзор

Откройте редактор загрузки данных и добавьте приведенный ниже скрипт загрузки на новую вкладку.

Скрипт загрузки содержит следующее:

- Используется тот же набор данных и сценарий, что в первом примере.
- Используется системная переменная `DateFormat` со значением по умолчанию MM/DD/YYYY.
- Аргумент `first_week_date` имеет значение 3. Это задает в качестве начала лунных недель 3 января.

Скрипт загрузки

```
SET DateFormat='MM/DD/YYYY';

Transactions:
    Load
        *,
        inlunarweek(date,'01/10/2022', 0,3) as in_lunar_week_to_date
        ;
Load
*
Inline
[
id,date,amount
8188,'1/10/2022',37.23
8189,'1/17/2022',17.17
8190,'1/26/2022',88.27
8191,'1/12/2022',57.42
8192,'1/19/2022',53.80
8193,'1/21/2022',82.06
8194,'1/1/2022',40.39
8195,'1/27/2022',87.21
8196,'1/11/2022',95.93
8197,'1/29/2022',45.89
8198,'1/31/2022',36.23
8199,'1/18/2022',25.66
8200,'1/23/2022',82.77
8201,'1/15/2022',69.98
8202,'1/4/2022',76.11
];
```

Результаты

Загрузите данные и откройте лист. Создайте новую таблицу и добавьте эти поля как измерения:

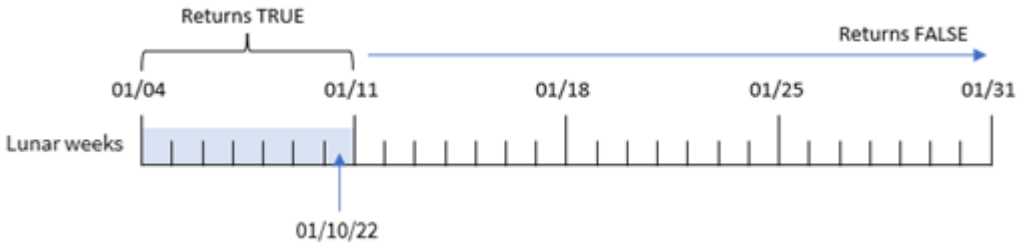
- `date`
- `in_lunar_week_to_date`

Результирующая таблица

date	in_lunar_week_to_date
1/1/2022	0
1/4/2022	-1
1/10/2022	-1
1/11/2022	0
1/12/2022	0
1/15/2022	0

date	in_lunar_week_to_date
1/17/2022	0
1/18/2022	0
1/19/2022	0
1/21/2022	0
1/23/2022	0
1/26/2022	0
1/27/2022	0
1/29/2022	0
1/31/2022	0

Функция `inlunarweektodate()`, пример с аргументом first_week_day



Так как в данном примере используется аргумент `the first_week_date` = 3 для функции `inlunarweek()`, первая лунная неделя начинается 3 января и заканчивается 10 января. Так как 10 января также является значением `base_date`, любая транзакция, которая совершается между этими датами, возвращает булево значение `TRUE`.

Пример 4. Пример объекта диаграммы

Скрипт загрузки и выражение диаграммы

Обзор

Откройте редактор загрузки данных и добавьте приведенный ниже скрипт загрузки на новую вкладку.

Скрипт загрузки содержит тот же набор данных и сценарий, что в первом примере.

Однако в этом примере в приложение загружается неизмененный набор данных. Расчет, который определяет, совершены ли транзакции в лунную неделю до 10 января, создается как мера в объекте диаграммы в приложении.

Скрипт загрузки

```
SET DateFormat='MM/DD/YYYY';

Transactions:
Load
*
Inline
[
id,date,amount
8188,'1/10/2022',37.23
8189,'1/17/2022',17.17
8190,'1/26/2022',88.27
8191,'1/12/2022',57.42
8192,'1/19/2022',53.80
8193,'1/21/2022',82.06
8194,'1/1/2022',40.39
8195,'1/27/2022',87.21
8196,'1/11/2022',95.93
8197,'1/29/2022',45.89
8198,'1/31/2022',36.23
8199,'1/18/2022',25.66
8200,'1/23/2022',82.77
8201,'1/15/2022',69.98
8202,'1/4/2022',76.11
];
```

Результаты

Загрузите данные и откройте лист. Создайте новую таблицу и добавьте это поле как измерение: `date`.

Создайте следующую меру:

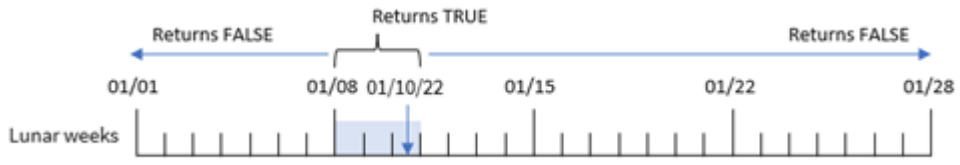
```
=inlunarweektodate(date,'01/10/2022', 0)
```

Результирующая таблица

date	=inlunarweektodate(date,'01/10/2022', 0)
1/1/2022	0
1/4/2022	0
1/10/2022	-1
1/11/2022	0
1/12/2022	0
1/15/2022	0
1/17/2022	0
1/18/2022	0

date	=inlunarweektodate(date,'01/10/2022', 0)
1/19/2022	0
1/21/2022	0
1/23/2022	0
1/26/2022	0
1/27/2022	0
1/29/2022	0
1/31/2022	0

Функция `inlunarweektodate()`, пример с объектом диаграммы



Мера `in_lunar_week_to_date` создана в объекте диаграммы с использованием функции `inlunarweektodate()`, где в качестве аргументов функции переданы поле даты, жестко запрограммированная дата 10 января в качестве `base_date` и сдвиг 0.

Так как лунная неделя начинается 1 января, 10 января относится к лунной неделе, которая начинается 8 января. Кроме того, поскольку используется функция `inlunarweektodate()`, эта лунная неделя заканчивается 10 января. Поэтому все транзакции, которые совершаются между этими числами января будут возвращать булево значение `TRUE`. Это можно проверить в таблице результатов.

Пример 5. Сценарий

Скрипт загрузки и выражения диаграммы

Обзор

Откройте редактор загрузки данных и добавьте приведенный ниже скрипт загрузки на новую вкладку.

Скрипт загрузки содержит следующее:

- Набор данных, загруженный в таблицу, называется `Products`
- Информация, содержащая идентификатор продукта, дату изготовления и себестоимость.

Установлено, что вследствие ошибки оборудования продукты, изготовленные в течение лунной недели, на которую выпадает 12 января, являются дефектными. Проблема была решена 13 января. Конечному пользователю требуется объект диаграммы, который отображает по неделе, какие изготовленные продукты были дефектными, а какие бездефектными, а также стоимость продуктов, изготовленных на этой неделе.

Скрипт загрузки

```
SET TimestampFormat='M/D/YYYY h:mm:ss[.fff]';

Products:
Load
*
Inline
[
product_id,manufacture_date,cost_price
8188,'01/02/2022 12:22:06',37.23
8189,'01/05/2022 01:02:30',17.17
8190,'01/06/2022 15:36:20',88.27
8191,'01/08/2022 10:58:35',57.42
8192,'01/09/2022 08:53:32',53.80
8193,'01/10/2022 21:13:01',82.06
8194,'01/11/2022 00:57:13',40.39
8195,'01/12/2022 09:26:02',87.21
8196,'01/13/2022 15:05:09',95.93
8197,'01/14/2022 18:44:57',45.89
8198,'01/15/2022 06:10:46',36.23
8199,'01/16/2022 06:39:27',25.66
8200,'01/17/2022 10:44:16',82.77
8201,'01/18/2022 18:48:17',69.98
8202,'01/26/2022 04:36:03',76.11
8203,'01/27/2022 08:07:49',25.12
8204,'01/28/2022 12:24:29',46.23
8205,'01/30/2022 11:56:56',84.21
8206,'01/30/2022 14:40:19',96.24
8207,'01/31/2022 05:28:21',67.67
];
```

Результаты

Выполните следующие действия.

1. Загрузите данные и откройте лист. Создайте новую таблицу.
2. Создайте измерение для отображения названий недель:
   `=weekname(manufacture_date)`
3. Затем создайте измерение, которое использует функцию `inlunarweektodate()`, чтобы определить, какие из продуктов дефектные, а какие нет:
   `=if(inlunarweektodate(manufacture_date,makedate(2022,01,12),0),'Defective','Faultless')`
4. Создайте меру для суммирования `cost_price` продуктов:

```
=sum(cost_price)
```

5. Задайте параметру меры **Формат чисел** значение **Денежный**.

Результирующая таблица

=lunarweekname (manufacture_date)	=if(InLunarWeekToDate(manufacture_date,makedate (2022,01,12),0),'Defective','Faultless')	=Sum(cost_ price)
2022/01	Бездефектный	$142.67
2022/02	Дефектный	$320.88
2022/02	Бездефектный	$141.82
2022/03	Бездефектный	$214.64
2022/04	Бездефектный	$147.46
2022/05	Бездефектный	$248.12

Функция `inlunarweektodate()` возвращает булево значение при проверке дат производства каждого продукта. Продукты, для которых возвращается булево значение `TRUE`, помечаются как `'Defective'`. Продукты, для которых возвращается значение `FALSE` и которые, следовательно, не произведены в течение лунной недели до 12 января, помечаются как `'Faultless'`.

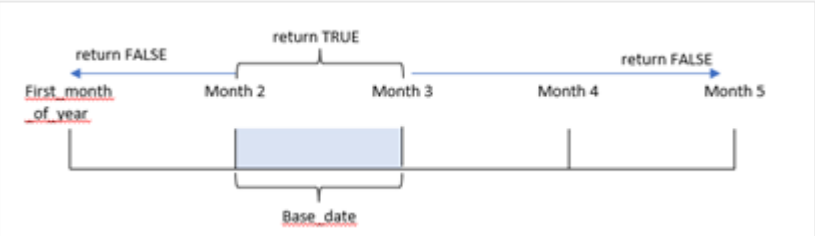
inmonth

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)
```

Схема функции `indaytotime`.



Другими словами, функция `inmonth()` определяет, попадает ли набор дат в этот месяц, и возвращает булево значение на основе `base_date`, которое идентифицирует месяц.

Когда это следует использовать

Функция `inmonth()` возвращает результат в виде логического значения. Обычно этот тип функции используется в качестве условия в `if expression`. Она возвращает агрегирование или расчет в зависимости от того, выпадает ли дата на указанный месяц, включая рассматриваемую дату.

Например, функцию `inmonth()` можно использовать для идентификации всего оборудования, изготовленного в указанном месяце.

Возвращаемые типы данных: Булево значение

В Qlik Sense логическое значение «истина» представлено как -1, а «ложь» – как 0.

Аргументы

Аргумент	Описание
timestamp	Дата, которую необходимо сравнить со значением, указанным в поле `base_date`.
base_date	Дата, использующаяся для оценки месяца. Важно отметить, что `base_date` может быть любым днем в течение 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.

Региональные настройки

Если не указано иное, в примерах, приведенных в данном разделе, используется следующий формат даты: ММ/ДД/ГГГГ. Формат даты указан в операторе `SET DateFormat` скрипта загрузки данных. В вашей системе может быть установлен другой формат даты по умолчанию в зависимости от региональных настроек и других факторов. Можно изменить формат в примерах в соответствии с потребностями. Или можно изменить форматы в скрипте загрузки в соответствии с этими примерами.

Региональные настройки по умолчанию в приложениях основаны на системных региональных настройках компьютера или сервера, где установлено ПО Qlik Sense. Если на сервере Qlik Sense, к которому обращается пользователь, выбран шведский язык, то редактор загрузки данных будет использовать шведские региональные настройки для даты, времени и валюты. Эти параметры регионального формата не связаны с языком, отображаемым в интерфейсе пользователя Qlik Sense. Qlik Sense будет отображаться на языке, который используется в браузере.

Примеры функции

Пример	Результат
`inmonth ('25/01/2013', '01/01/2013', 0)`	Возвращает True
`inmonth('25/01/2013', '23/04/2013', 0)`	Возвращает False
`inmonth ('25/01/2013', '01/01/2013', -1)`	Возвращает False
`inmonth ('25/12/2012', '17/01/2013', -1)`	Возвращает True

Пример 1. Без дополнительных аргументов

Скрипт загрузки и результаты

Обзор

Откройте редактор загрузки данных и добавьте приведенный ниже скрипт загрузки на новую вкладку.

Скрипт загрузки содержит следующее:

- Набор данных, содержащий набор транзакций за первую половину 2022 года.
- Предыдущая загрузка с дополнительной переменной 'in_month', которая определяет, осуществлены ли транзакции в апреле.

Скрипт загрузки

```
SET DateFormat='MM/DD/YYYY';

Transactions:
  Load
    *,
    inmonth(date,'04/01/2022', 0) as in_month
    ;
Load
*
Inline
[
id,date,amount
8188,'1/10/2022',37.23
8189,'1/14/2022',17.17
8190,'1/20/2022',88.27
8191,'1/22/2022',57.42
8192,'2/1/2022',53.80
8193,'2/2/2022',82.06
8194,'2/20/2022',40.39
8195,'4/11/2022',87.21
8196,'4/13/2022',95.93
8197,'4/15/2022',45.89
8198,'4/25/2022',36.23
8199,'5/20/2022',25.66
8200,'5/22/2022',82.77
8201,'6/19/2022',69.98
8202,'6/22/2022',76.11
];
```

Результаты

Загрузите данные и откройте лист. Создайте новую таблицу и добавьте эти поля как измерения:

- date
- in_month

Примеры функции

date	in_month
1/10/2022	0
1/14/2022	0
1/20/2022	0
1/22/2022	0
2/1/2022	0
2/2/2022	0
2/20/2022	0
4/11/2022	-1
4/13/2022	-1
4/15/2022	-1
4/25/2022	-1
5/20/2022	0
5/22/2022	0
6/19/2022	0
6/22/2022	0

Поле `in_month` создано предшествующим оператором load с использованием функции `inmonth()`, где в качестве аргументов функции переданы поле даты, жестко запрограммированная метка времени «1 апреля» в качестве `base_date` и `period_no` = `0`.

`base_date` определяет месяц, который вернет булев результат TRUE. Таким образом, все транзакции, совершенные в апреле, возвращают TRUE, что подтверждается таблицей результатов.

Пример 2. Скрипт period_no

Скрипт загрузки и результаты

Обзор

Используется тот же набор данных и сценарий, что в первом примере.

Однако в этом примере вы создадите поле `2_months_prior`, которое определяет, были ли транзакции совершены за два месяца до апреля.

Скрипт загрузки

```
SET DateFormat='MM/DD/YYYY';

Transactions:
Load
```

```
   *,
   inmonth(date,'04/01/2022', -2) as [2_months_prior]
Inline
[
id,date,amount
8188,'1/10/2022',37.23
8189,'1/14/2022',17.17
8190,'1/20/2022',88.27
8191,'1/22/2022',57.42
8192,'2/1/2022',53.80
8193,'2/2/2022',82.06
8194,'2/20/2022',40.39
8195,'4/11/2022',87.21
8196,'4/13/2022',95.93
8197,'4/15/2022',45.89
8198,'4/25/2022',36.23
8199,'5/20/2022',25.66
8200,'5/22/2022',82.77
8201,'6/19/2022',69.98
8202,'6/22/2022',76.11
];
```

Результаты

Загрузите данные и откройте лист. Создайте новую таблицу и добавьте эти поля как измерения:

- `date`
- `2_months_prior`

Примеры функции

date	2_months_prior
1/10/2022	0
1/14/2022	0
1/20/2022	0
1/22/2022	0
2/1/2022	-1
2/2/2022	-1
2/20/2022	-1
4/11/2022	0
4/13/2022	0
4/15/2022	0
4/25/2022	0
5/20/2022	0
5/22/2022	0

date	2_months_prior
6/19/2022	0
6/22/2022	0

Использование -2 в качестве аргумента `period_no` в функции `inmonth()` сдвигает месяц, определенный аргументом `base_date`, на два месяца ранее. В этом примере он изменяет заданный месяц с апреля на февраль.

Таким образом, любая транзакция, совершенная в феврале, вернет булев результат TRUE.

Пример 3. Объект диаграммы

Скрипт загрузки и выражение диаграммы

Обзор

Используется тот же набор данных и сценарий, что в предыдущих примерах.

Однако в этом примере в приложение загружается неизмененный набор данных. Расчет, который определяет, совершены ли транзакции в апреле, создается как мера в объекте диаграммы в приложении.

Скрипт загрузки

```
SET DateFormat='MM/DD/YYYY';

Transactions:
Load
*
Inline
[
id,date,amount
8188,'1/10/2022',37.23
8189,'1/14/2022',17.17
8190,'1/20/2022',88.27
8191,'1/22/2022',57.42
8192,'2/1/2022',53.80
8193,'2/2/2022',82.06
8194,'2/20/2022',40.39
8195,'4/11/2022',87.21
8196,'4/13/2022',95.93
8197,'4/15/2022',45.89
8198,'4/25/2022',36.23
8199,'5/20/2022',25.66
8200,'5/22/2022',82.77
8201,'6/19/2022',69.98
8202,'6/22/2022',76.11
];
```

Объект диаграммы

Загрузите данные и откройте лист. Создайте новую таблицу и добавьте это поле как измерение:

```
date
```

Чтобы рассчитать, совершена ли транзакция в апреле, создайте следующую меру:

```
=inmonth(date,'04/01/2022', 0)
```

Результаты

	Примеры функции
date	**=inmonth(date,'04/01/2022', 0)**
1/10/2022	0
1/14/2022	0
1/20/2022	0
1/22/2022	0
2/1/2022	0
2/2/2022	0
2/20/2022	0
4/11/2022	-1
4/13/2022	-1
4/15/2022	-1
4/25/2022	-1
5/20/2022	0
5/22/2022	0
6/19/2022	0
6/22/2022	0

Пример 4. Сценарий

Скрипт загрузки и результаты

Обзор

В этом примере набор данных загружается в таблицу под именем `Products`. Данная таблица содержит следующие поля:

- ИД продукта
- Время изготовления
- Себестоимость

Из-за ошибки оборудования продукты, произведенные в июле 2022 года, оказались бракованными. Проблема была решена 27 июля 2022 года.

Конечному пользователю требуется объект диаграммы, который отображает по месяцу, какие изготовленные продукты были дефектными, а какие бездефектными, а также стоимость продуктов, изготовленных в этом месяце.

Скрипт загрузки

```
Products:
Load
*
Inline
[
product_id,manufacture_date,cost_price
8188,'1/19/2022',37.23
8189,'1/7/2022',17.17
8190,'2/28/2022',88.27
8191,'2/5/2022',57.42
8192,'3/16/2022',53.80
8193,'4/1/2022',82.06
8194,'5/7/2022',40.39
8195,'5/16/2022',87.21
8196,'6/15/2022',95.93
8197,'6/26/2022',45.89
8198,'7/9/2022',36.23
8199,'7/22/2022',25.66
8200,'7/23/2022',82.77
8201,'7/27/2022',69.98
8202,'8/2/2022',76.11
8203,'8/8/2022',25.12
8204,'8/19/2022',46.23
8205,'9/26/2022',84.21
8206,'10/14/2022',96.24
8207,'10/29/2022',67.67
];
```

Результаты

Загрузите данные и откройте лист. Создайте новую таблицу и добавьте это поле как измерение:

```
=monthname(manufacture_date)
```

Создайте следующие меры:

- `=sum(cost_price)`
- `=if(only(inmonth(manufacture_date,makedate(2022,07,01),0)),'Defective','Faultless')`

1. Задайте параметру **Формат чисел** меры значение **Денежный**.
2. В области **Вид** выключите параметр **Итоги**.

Результирующая таблица

monthname (manufacture_date)	=if(only(inmonth(manufacture_date,makedate (2022,07,01),0)),'Defective','Faultless')	sum(cost_ price)
Jan 2022	Бездефектный	$54.40
Feb 2022	Бездефектный	$145.69
Mar 2022	Бездефектный	$53.80
Apr 2022	Бездефектный	$82.06
May 2022	Бездефектный	$127.60
Jun 2022	Бездефектный	$141.82
Jul 2022	Дефектный	$214.64
Aug 2022	Бездефектный	$147.46
Sep 2022	Бездефектный	$84.21
Oct 2022	Бездефектный	$163.91

Функция `inmonth()` возвращает логическое значение при проверки дат производства каждого продукта. Для любого изделия, изготовленного в июле 2022 года, функция `inmonth()` возвращает булево значение True и ставит отметку «Дефектный». Любое изделие, которое возвращает значение False, то есть изготовлено в другой день, отмечается как «Бездефектный».

inmonths

Эта функция определяет, находится ли метка времени базовой даты в части месяца, двухмесячного периода, квартала, трети года (четыре месяца) или полугодия. Также можно проследить, находится ли метка времени в предыдущем или в последующем временном периоде.

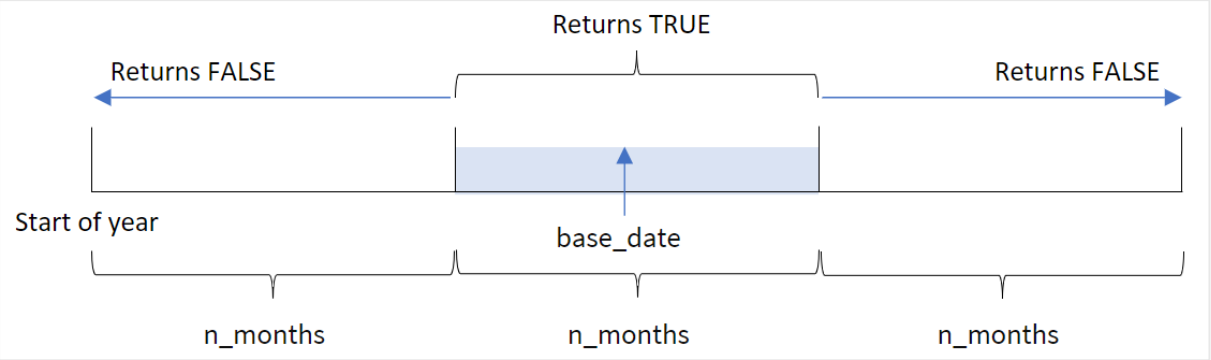
Синтаксис:

```
InMonths(n_months, timestamp, base_date, period_no [, first_month_of_year])
```

Возвращаемые типы данных: Булево значение

В Qlik Sense логическое значение «истина» представлено как -1, а «ложь» – как 0.

Схема функции `inmonths()`



Функция `inmonths()` делит год на сегменты на основе предоставленного аргумента `n_months`. Затем она определяет, относится ли каждая оцениваемая метка времени к тому же сегменту, что аргумент `base_date`. Однако если передан аргумент `period_no`, функция определяет, попадают ли метки времени в предыдущий или следующий период относительно `base_date`.

В качестве аргументов `n_month` в функции доступны следующие отрезки года.

Аргументы `n_month`

Период	Количество месяцев
месяц	1
два месяца	2
квартал	3
четыре месяца	4
полгода	6

Когда это следует использовать

Функция `inmonths()` возвращает результат в виде булева значения. Обычно этот тип функции используется в качестве условия в `if expression`. Используя функцию `inmonths()`, можно выбрать период, который требуется оценить. Например, это дает возможность идентифицировать продукты, произведенные в месяце, квартале или полугодии определенного периода.

Возвращаемые типы данных: Булево значение

В Qlik Sense логическое значение «истина» представлено как -1, а «ложь» – как 0.

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 (четыре месяца) или 6 (полугодие).

Аргумент	Описание
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**.

Можно использовать следующие значения, чтобы задать первый месяц года в аргументе `first_month_of_year`:

значения first_month_of_year

Месяц	Значение
Февраль	2
Март	3
Апрель	4
Мау	5
Июнь	6
Июль	7
Август	8
Сентябрь	9
Октябрь	10
Ноябрь	11
Декабрь	12

Региональные настройки

Если не указано иное, в примерах, приведенных в данном разделе, используется следующий формат даты: ММ/ДД/ГГГГ. Формат даты указан в операторе `SET DateFormat` скрипта загрузки данных. В вашей системе может быть установлен другой формат даты по умолчанию в зависимости от региональных настроек и других факторов. Можно изменить формат в примерах в соответствии с потребностями. Или можно изменить форматы в скрипте загрузки в соответствии с этими примерами.

Региональные настройки по умолчанию в приложениях основаны на системных региональных настройках компьютера или сервера, где установлено ПО Qlik Sense. Если на сервере Qlik Sense, к которому обращается пользователь, выбран шведский язык, то редактор загрузки данных будет

использовать шведские региональные настройки для даты, времени и валюты. Эти параметры регионального формата не связаны с языком, отображаемым в интерфейсе пользователя Qlik Sense. Qlik Sense будет отображаться на языке, который используется в браузере.

Примеры функции

Пример	Результат
`inmonths(4, '01/25/2013', '04/25/2013', 0)`	Возвращает TRUE. Поскольку значение timestamp, 01/25/2013, находится в четырехмесячном периоде с 01/01/2013 по 04/30/2013, в котором находится значение `base_date`, 04/25/2013.
`inmonths(4, '05/25/2013', '04/25/2013', 0)`	Возвращает FALSE. Поскольку 05/25/2013 находится за пределами периода, указанного в предыдущем примере.
`inmonths(4, '11/25/2012', '02/01/2013', -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 09/01/2012 по 12/31/2012.
`inmonths(4, '05/25/2006', '03/01/2006', 0, 3)`	Возвращает TRUE. Возвращает `first_month_of_year`, так как значение равно `3`, в результате чего период поиска длится с 03/01/2006 по 07/30/2006, а не с 01/01/2006 по 04/30/2006.

Пример 1. Без дополнительных аргументов

Скрипт загрузки и результаты

Обзор

Откройте редактор загрузки данных и добавьте приведенный ниже скрипт загрузки на новую вкладку.

Скрипт загрузки содержит следующее:

- Набор данных, содержащий набор транзакций за 2022 год, загруженный в таблицу под именем `Transactions`.
- Предыдущая загрузка с дополнительной переменной '`in_months`', которая определяет, совершены ли транзакции в том же квартале, к которому относится 15 мая 2022 года.

Скрипт загрузки

```
SET DateFormat='MM/DD/YYYY';

Transactions:
    Load
       *,
       inmonths(3,date,'05/15/2022', 0) as in_months
       ;
Load
```

```
*
Inline
[
id,date,amount
8188,'2/19/2022',37.23
8189,'3/7/2022',17.17
8190,'3/30/2022',88.27
8191,'4/5/2022',57.42
8192,'4/16/2022',53.80
8193,'5/1/2022',82.06
8194,'5/7/2022',40.39
8195,'5/22/2022',87.21
8196,'6/15/2022',95.93
8197,'6/26/2022',45.89
8198,'7/9/2022',36.23
8199,'7/22/2022',25.66
8200,'7/23/2022',82.77
8201,'7/27/2022',69.98
8202,'8/2/2022',76.11
8203,'8/8/2022',25.12
8204,'8/19/2022',46.23
8205,'9/26/2022',84.21
8206,'10/14/2022',96.24
8207,'10/29/2022',67.67
];
```

Результаты

Загрузите данные и откройте лист. Создайте новую таблицу и добавьте эти поля как измерения:

- `date`
- `in_months`

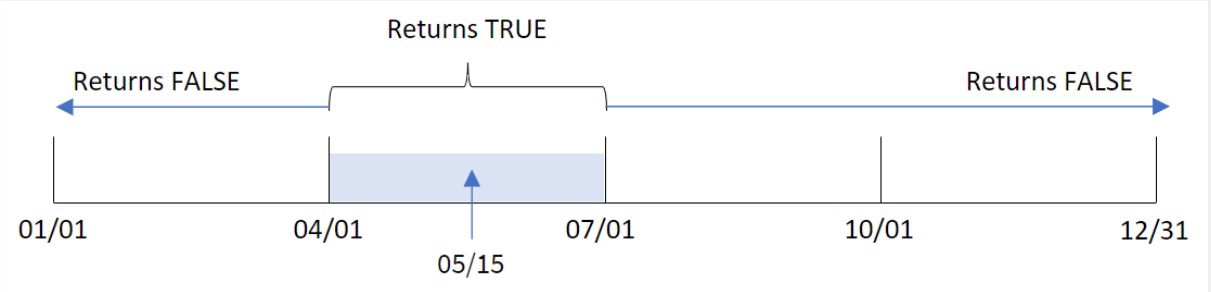
Результирующая таблица

date	in_months
2/19/2022	0
3/7/2022	0
3/30/2022	0
4/5/2022	-1
4/16/2022	-1
5/1/2022	-1
5/7/2022	-1
5/22/2022	-1
6/15/2022	-1
6/26/2022	-1

date	in_months
7/9/2022	0
7/22/2022	0
7/23/2022	0
7/27/2022	0
8/2/2022	0
8/8/2022	0
8/19/2022	0
9/26/2022	0
10/14/2022	0
10/29/2022	0

Поле `in_months` создано в предыдущем операторе load с помощью функции `inmonths()`. Первым предоставлен аргумент `3`, который делит год на квартальные сегменты. Второй аргумент определяет, какое поле оценивается. В данном примере это поле даты. Третий аргумент – это жестко закодированная дата 15 мая, которая является `base_date`, а `period_no` = `0` является заключительным аргументом.

Диаграмма функции `inmonths()` с квартальными сегментами



Май попадает во второй квартал года. Поэтому все транзакции, которые совершаются с 1 апреля по 30 июня будут возвращать булев результат TRUE. Это можно проверить в таблице результатов.

Пример 2. Скрипт period_no

Скрипт загрузки и результаты

Обзор

Откройте редактор загрузки данных и добавьте приведенный ниже скрипт загрузки на новую вкладку.

Скрипт загрузки содержит следующее:

- Набор данных, содержащий набор транзакций за 2022 год, загруженный в таблицу под именем `Transactions`.
- Предыдущая загрузка с дополнительной переменной '`previous_quarter`', которая определяет, совершены ли транзакции в квартале до 15 мая 2022 года.

Скрипт загрузки

```
SET DateFormat='MM/DD/YYYY';

Transactions:
    Load
       *,
       inmonths(3,date,'05/15/2022', -1) as previous_quarter
       ;
Load
*
Inline
[
id,date,amount
8188,'2/19/2022',37.23
8189,'3/7/2022',17.17
8190,'3/30/2022',88.27
8191,'4/5/2022',57.42
8192,'4/16/2022',53.80
8193,'5/1/2022',82.06
8194,'5/7/2022',40.39
8195,'5/22/2022',87.21
8196,'6/15/2022',95.93
8197,'6/26/2022',45.89
8198,'7/9/2022',36.23
8199,'7/22/2022',25.66
8200,'7/23/2022',82.77
8201,'7/27/2022',69.98
8202,'8/2/2022',76.11
8203,'8/8/2022',25.12
8204,'8/19/2022',46.23
8205,'9/26/2022',84.21
8206,'10/14/2022',96.24
8207,'10/29/2022',67.67
];
```

Результаты

Загрузите данные и откройте лист. Создайте новую таблицу и добавьте эти поля как измерения:

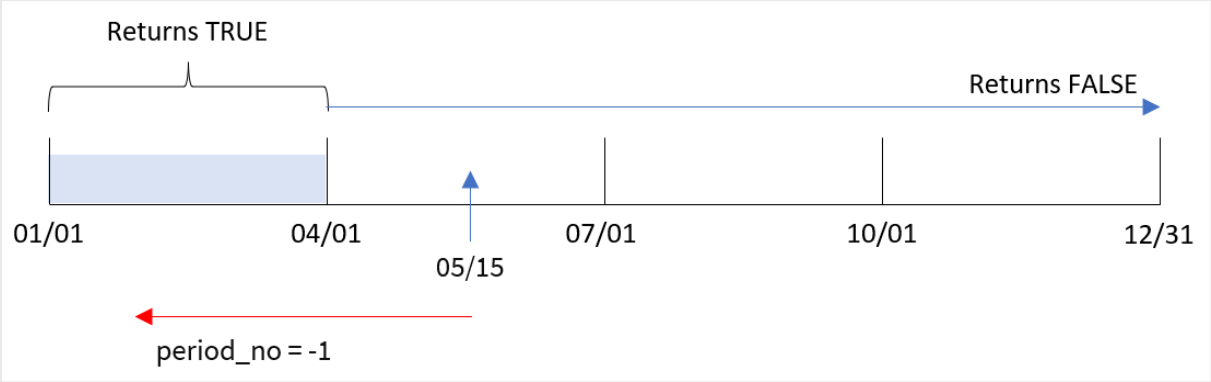
- `date`
- `previous_quarter`

Результирующая таблица

date	previous quarter
2/19/2022	-1
3/7/2022	-1
3/30/2022	-1
4/5/2022	0
4/16/2022	0
5/1/2022	0
5/7/2022	0
5/22/2022	0
6/15/2022	0
6/26/2022	0
7/9/2022	0
7/22/2022	0
7/23/2022	0
7/27/2022	0
8/2/2022	0
8/8/2022	0
8/19/2022	0
9/26/2022	0
10/14/2022	0
10/29/2022	0

Функция оценивает, совершены ли транзакции в первый квартал года, используя `-1` в качестве аргумента `period_no` в функции `inmonths()`. 15 мая является `base_date` и попадает во второй квартал года (апрель-июнь).

Диаграмма функции `inmonths()` с квартальными сегментами и `period_no = -1`



Таким образом, любая транзакция, совершенная в период с января по март, возвращает булев результат TRUE.

Пример 3. Аргумент first_month_of_year

Скрипт загрузки и результаты

Обзор

Откройте редактор загрузки данных и добавьте приведенный ниже скрипт загрузки на новую вкладку.

Скрипт загрузки содержит следующее:

- Набор данных, содержащий набор транзакций за 2022 год, загруженный в таблицу под именем `Transactions`.
- Предыдущая загрузка с дополнительной переменной '`in_months`', которая определяет, совершены ли транзакции в том же квартале, к которому относится 15 мая 2022 года.

В этом примере политика организации устанавливает март в качестве первого месяца финансового года.

Скрипт загрузки

```
SET DateFormat='MM/DD/YYYY';

Transactions:
    Load
       *,
       inmonths(3,date,'05/15/2022', 0, 3) as in_months
       ;
Load
*
Inline
[
id,date,amount
8188,'2/19/2022',37.23
8189,'3/7/2022',17.17
```

```
8190,'3/30/2022',88.27
8191,'4/5/2022',57.42
8192,'4/16/2022',53.80
8193,'5/1/2022',82.06
8194,'5/7/2022',40.39
8195,'5/22/2022',87.21
8196,'6/15/2022',95.93
8197,'6/26/2022',45.89
8198,'7/9/2022',36.23
8199,'7/22/2022',25.66
8200,'7/23/2022',82.77
8201,'7/27/2022',69.98
8202,'8/2/2022',76.11
8203,'8/8/2022',25.12
8204,'8/19/2022',46.23
8205,'9/26/2022',84.21
8206,'10/14/2022',96.24
8207,'10/29/2022',67.67
];
```

Результаты

Загрузите данные и откройте лист. Создайте новую таблицу и добавьте эти поля как измерения:

- `date`
- `in_months`

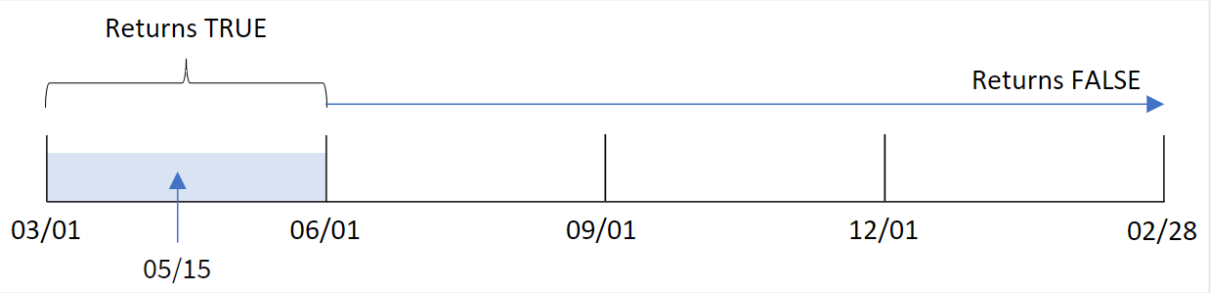
Результирующая таблица

date	in_months
2/19/2022	0
3/7/2022	-1
3/30/2022	-1
4/5/2022	-1
4/16/2022	-1
5/1/2022	-1
5/7/2022	-1
5/22/2022	-1
6/15/2022	0
6/26/2022	0
7/9/2022	0
7/22/2022	0
7/23/2022	0

date	in_months
7/27/2022	0
8/2/2022	0
8/8/2022	0
8/19/2022	0
9/26/2022	0
10/14/2022	0
10/29/2022	0

Так как используется `3` в качестве аргумента `first_month_of_year` в функции `inmonths()`, функция начинает год 1 марта. Затем функция `inmonths()` делит год на кварталы: Mar-May (март-май), Jun-Aug (июнь-август), Sep-Nov (сентябрь-ноябрь), Dec-Feb (декабрь-февраль). Таким образом, 15 мая попадает в первый квартал года (март-май).

Диаграмма функции `inmonths()`, где в качестве первого месяца года задан март.



Любая транзакция, совершенная в эти месяцы, будет возвращать булев результат TRUE.

Пример 4. Пример объекта диаграммы

Скрипт загрузки и выражение диаграммы

Обзор

Используется тот же набор данных и сценарий, что в первом примере.

Однако в этом примере в приложение загружается неизмененный набор данных. Вычисление, которое определяет, совершены ли транзакции в том квартале, в который попадает 15 мая 2022 года, создается как мера в диаграмме приложения.

Скрипт загрузки

```
SET DateFormat='MM/DD/YYYY';

Transactions:
Load
*
```

```
Inline
[
id,date,amount
8188,'2/19/2022',37.23
8189,'3/7/2022',17.17
8190,'3/30/2022',88.27
8191,'4/5/2022',57.42
8192,'4/16/2022',53.80
8193,'5/1/2022',82.06
8194,'5/7/2022',40.39
8195,'5/22/2022',87.21
8196,'6/15/2022',95.93
8197,'6/26/2022',45.89
8198,'7/9/2022',36.23
8199,'7/22/2022',25.66
8200,'7/23/2022',82.77
8201,'7/27/2022',69.98
8202,'8/2/2022',76.11
8203,'8/8/2022',25.12
8204,'8/19/2022',46.23
8205,'9/26/2022',84.21
8206,'10/14/2022',96.24
8207,'10/29/2022',67.67
];
```

Результаты

Загрузите данные и откройте лист. Создайте новую таблицу и добавьте это поле как измерение:

- `date`

Чтобы рассчитать, совершены ли транзакции в том же квартале, к которому относится 15 мая, создайте следующую меру:

```
=inmonths(3,date,'05/15/2022', 0)
```

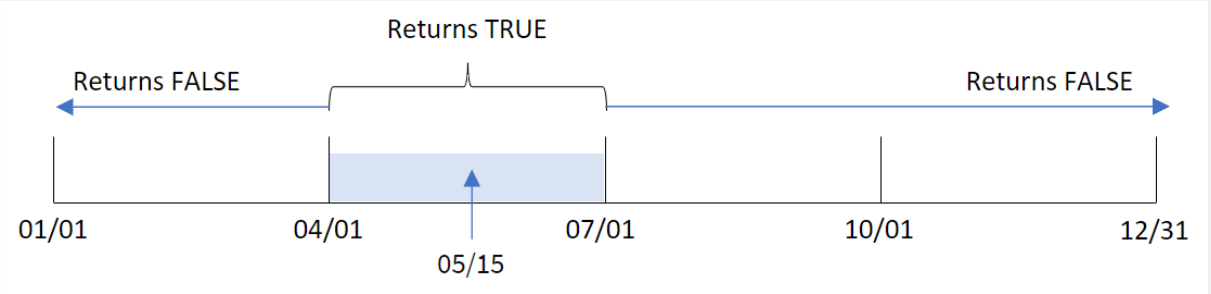
Результирующая таблица

date	=inmonths(3,date,'05/15/2022', 0)
2/19/2022	0
3/7/2022	0
3/30/2022	0
4/5/2022	-1
4/16/2022	-1
5/1/2022	-1
5/7/2022	-1
5/22/2022	-1
6/15/2022	-1

date	=inmonths(3,date,'05/15/2022', 0)
6/26/2022	-1
7/9/2022	0
7/22/2022	0
7/23/2022	0
7/27/2022	0
8/2/2022	0
8/8/2022	0
8/19/2022	0
9/26/2022	0
10/14/2022	0
10/29/2022	0

Поле `in_months` создается в диаграмме с помощью функции `inmonths()`. Первым предоставлен аргумент `3`, который делит год на кварталы. Второй аргумент определяет, какое поле оценивается. В данном примере это поле даты. Третий аргумент – это жестко закодированная дата 15 мая, которая является `base_date`, а `period_no` = `0` является заключительным аргументом.

Диаграмма функции `inmonths()` с квартальными сегментами



Май попадает во второй квартал года. Поэтому все транзакции, которые совершаются с 1 апреля по 30 июня будут возвращать булев результат TRUE. Это можно проверить в таблице результатов.

Пример 5. Сценарий

Скрипт загрузки и результаты

Обзор

Откройте редактор загрузки данных и добавьте приведенный ниже скрипт загрузки на новую вкладку.

Скрипт загрузки содержит следующее:

- Набор данных, загруженный в таблицу под именем `Products`.
- Данная таблица содержит следующие поля:
    - product ID (ИД продукта)
    - product type (тип продукта)
    - manufacture date (дата изготовления)
    - cost price (себестоимость)

Конечному пользователю нужна диаграмма, отображающая по типу продукта стоимость продуктов, произведенных в первом сегмента 2021 года. Пользователю требуется возможность определить продолжительность этого сегмента.

Скрипт загрузки

```
SET vPeriod = 1;

Products:
Load
*
Inline
[
product_id,product_type,manufacture_date,cost_price
8188,product A,'2/19/2022',37.23
8189,product D,'3/7/2022',17.17
8190,product C,'3/30/2022',88.27
8191,product B,'4/5/2022',57.42
8192,product D,'4/16/2022',53.80
8193,product D,'5/1/2022',82.06
8194,product A,'5/7/2022',40.39
8195,product B,'5/22/2022',87.21
8196,product C,'6/15/2022',95.93
8197,product B,'6/26/2022',45.89
8198,product C,'7/9/2022',36.23
8199,product D,'7/22/2022',25.66
8200,product D,'7/23/2022',82.77
8201,product A,'7/27/2022',69.98
8202,product A,'8/2/2022',76.11
8203,product B,'8/8/2022',25.12
8204,product B,'8/19/2022',46.23
8205,product B,'9/26/2022',84.21
8206,product C,'10/14/2022',96.24
8207,product D,'10/29/2022',67.67
];
```

Результаты

Загрузите данные и откройте лист.

В начале скрипта загрузки создана переменная `vPeriod`, которая будет привязана к элементу управления вводом переменной.

Выполните следующие действия.

1. На панели ресурсов щелкните **Пользовательские объекты**.
2. Выберите **Qlik Dashboard bundle** и создайте объект **Ввод переменной**.
3. Введите заголовок для объекта диаграммы.
4. В разделе **Переменная** выберите **vPeriod** в качестве имени и задайте для отображения объект **Раскрывающийся список**.
5. В списке **Значения** выберите **Динамическое**. Введите следующее: `='1~month|2~bi-month|3~quarter|4~tertial|6~half-year'.`
6. Добавьте на лист новую таблицу.
7. На панели свойств в разделе **Данные** можно добавить измерение `product_type`.
8. Добавьте следующее выражение в качестве меры: `=sum(if(inmonths($(vPeriod),manufacture_date,makedate(2022,01,01),0),cost_price,0))`
9. Задайте параметру **Формат чисел** меры значение **Денежный**.

Результирующая таблица

product_type	=sum(if(inmonths($(vPeriod),manufacture_date,makedate(2022,01,01),0),cost_price,0))
product A	$88.27
product B	$37.23
product C	$17.17
product D	$0.00

Функция `inmonths()` использует пользовательский ввод в качестве аргумента для определения размера начального сегмента года. Функция передает дату изготовления каждого продукта в качестве второго аргумента функции `inmonths()`. Так как 1 января используется в качестве третьего аргумента функции `inmonths()`, продукты, дата изготовления которых попадает в начальный сегмент года, возвращают булево значение TRUE и поэтому функция sum будет суммировать стоимость этих продуктов.

inmonthstodate

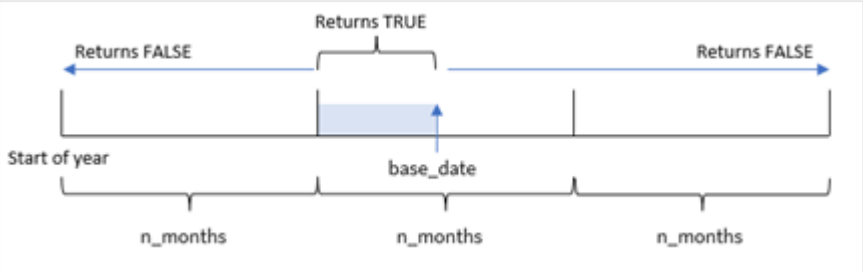
Эта функция определяет, находится ли метка времени в части месяца, двухмесячного периода, квартала, трети года (четыре месяца) или полугодия до последней миллисекунды, указанной в поле `base_date`, включительно. Также можно проследить, находится ли метка времени в предыдущем или в последующем временном периоде.

Синтаксис:

```
InMonths (n_months, timestamp, base_date, period_no[, first_month_of_year ])
```

Возвращаемые типы данных: Булево значение

Схема функции `inmonthstodate`.



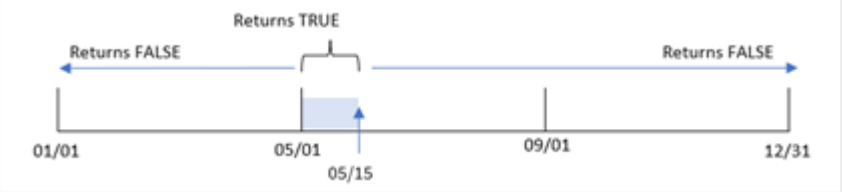
Аргументы

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 (четыре месяца) 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**.

В функции `inmonthstodate()` `base_date` играет роль конечной точки определенного сегмента года, к которому он относится.

Например, если год разбит на сегменты по четыре месяца (треть года) и в качестве `base_date` задано 15 мая, любая метка времени с начала января до конца апреля будет возвращать булев результат FALSE. Даты с 1 по 15 мая будут возвращать значение TRUE. Остальные метки времени будут возвращать FALSE.

Диаграмма диапазона булевых результатов для функции `inmonthstodate`.



В качестве аргументов `n_month` в функции доступны следующие отрезки года.

Аргументы `n_month`

Период	Количество месяцев
месяц	1
два месяца	2
квартал	3
треть года	4
полгода	6

Когда это следует использовать

Функция `inmonthstodate()` возвращает результат в виде булева значения. Обычно этот тип функции используется в качестве условия в `if expression`. Используя функцию `inmonthstodate()`, можно выбрать период, который требуется оценить. Например, предоставление входной переменной, которая позволяет идентифицировать продукты, произведенные в месяце, квартале или полугодии определенного периода, вплоть до определенной даты.

Региональные настройки

Если не указано иное, в примерах, приведенных в данном разделе, используется следующий формат даты: ММ/ДД/ГГГГ. Формат даты указан в операторе `SET DateFormat` скрипта загрузки данных. В вашей системе может быть установлен другой формат даты по умолчанию в зависимости от региональных настроек и других факторов. Можно изменить формат в примерах в соответствии с потребностями. Или можно изменить форматы в скрипте загрузки в соответствии с этими примерами.

Региональные настройки по умолчанию в приложениях основаны на системных региональных настройках компьютера или сервера, где установлено ПО Qlik Sense. Если на сервере Qlik Sense, к которому обращается пользователь, выбран шведский язык, то редактор загрузки данных будет использовать шведские региональные настройки для даты, времени и валюты. Эти параметры регионального формата не связаны с языком, отображаемым в интерфейсе пользователя Qlik Sense. Qlik Sense будет отображаться на языке, который используется в браузере.

Примеры функции

Пример	Результат
`inmonthstodate (4, '01/25/2013', '04/25/2013', 0)`	Возвращает True, поскольку значение timestamp, 01/25/2013, находится в четырехмесячном периоде 01/01/2013 по 04/25/2013, в котором находится значение base_date, 04/25/2013.
`inmonthstodate (4, '04/26/2013', '04/25/2006', 0)`	Возвращает False, поскольку 04/26/2013 находится за пределами периода, указанного в предыдущем примере.

Пример	Результат
`inmonthstodate (4, '09/25/2005', '02/01/2006', -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 02/01/2006.
`inmonthstodate (4, '04/25/2006', '06/01/2006', 0, 3)`	Возвращает True, поскольку для значения first_month_of_year задано 3, вследствие чего период поиска будет составлять промежуток с 03/01/2006 по 06/01/2006 вместо промежутка с 05/01/2006 по 06/01/2006.

Пример 1. Без дополнительных аргументов

Скрипт загрузки и результаты

Обзор

Откройте редактор загрузки данных и добавьте приведенный ниже скрипт загрузки на новую вкладку.

Скрипт загрузки содержит следующее:

- Набор данных, содержащий набор транзакций за 2022 год, загруженный в таблицу под именем `Transactions`.
- Поле даты, предоставленное в формате `(MM/DD/YYYY)` системной переменной `DateFormat`.
- Предыдущий оператор load, который содержит следующее:
  - Функция `inmonthstodate()`, заданная в качестве поля ‘`in_months_to_date`’. Она определяет, какие транзакции совершены в квартале до 15 мая 2022 года.

Скрипт загрузки

```
SET DateFormat='MM/DD/YYYY';

Transactions:
    Load
    *,
    inmonthstodate(3,date,'05/15/2022', 0) as in_months_to_date
    ;
Load
*
Inline
[
id,date,amount
8188,'1/19/2022',37.23
8189,'1/7/2022',17.17
8190,'2/28/2022',88.27
8191,'2/5/2022',57.42
8192,'3/16/2022',53.80
8193,'4/1/2022',82.06
8194,'5/7/2022',40.39
```

```
8195,'5/16/2022',87.21
8196,'6/15/2022',95.93
8197,'6/26/2022',45.89
8198,'7/9/2022',36.23
8199,'7/22/2022',25.66
8200,'7/23/2022',82.77
8201,'7/27/2022',69.98
8202,'8/2/2022',76.11
8203,'8/8/2022',25.12
8204,'8/19/2022',46.23
8205,'9/26/2022',84.21
8206,'10/14/2022',96.24
8207,'10/29/2022',67.67
];
```

Результаты

Загрузите данные и откройте лист. Создайте новую таблицу и добавьте эти поля как измерения:

- `date`
- `in_months_to_date`

Результирующая таблица

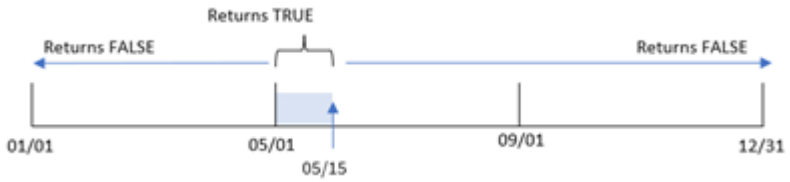
date	in_months_to_date
1/7/2022	0
1/19/2022	0
2/5/2022	0
2/28/2022	0
3/16/2022	0
4/1/2022	-1
5/7/2022	-1
5/16/2022	0
6/15/2022	0
6/26/2022	0
7/9/2022	0
7/22/2022	0
7/23/2022	0
7/27/2022	0
8/2/2022	0
8/8/2022	0
8/19/2022	0

date	in_months_to_date
9/26/2022	0
10/14/2022	0
10/29/2022	0

Поле `in_months_to_date` создано в предыдущем операторе load с помощью функции `inmonthstodate ()`.

Первым предоставлен аргумент `3`, который делит год на квартальные сегменты. Второй аргумент определяет, какое поле оценивается. Третий аргумент – это жестко закодированная дата 15 мая, которая представляет собой `base_date`, определяющий конечную границу сегмента. Заключительный аргумент – `period_no = 0`.

Диаграмма функции `inmonthstodate` без дополнительных аргументов



Любая транзакция, совершенная в период с 1 апреля по 15 мая, возвращает булев результат TRUE. Для транзакций, даты которых не укладываются в этот период, возвращается значение FALSE.

Пример 2. Скрипт period_no

Скрипт загрузки и результаты

Обзор

Используется тот же набор данных и сценарий, что в первом примере.

Однако в этом примере стоит задача создать поле `previous_qtr_to_date`, которое определяет, совершены ли транзакции за квартал до 15 мая.

Скрипт загрузки

```
SET DateFormat='MM/DD/YYYY';

Transactions:
    Load
    *,
    inmonthstodate(3,date,'05/15/2022', -1) as previous_qtr_to_date
    ;
Load
*
Inline
[
```

```
id,date,amount
8188,'1/19/2022',37.23
8189,'1/7/2022',17.17
8190,'2/28/2022',88.27
8191,'2/5/2022',57.42
8192,'3/16/2022',53.80
8193,'4/1/2022',82.06
8194,'5/7/2022',40.39
8195,'5/16/2022',87.21
8196,'6/15/2022',95.93
8197,'6/26/2022',45.89
8198,'7/9/2022',36.23
8199,'7/22/2022',25.66
8200,'7/23/2022',82.77
8201,'7/27/2022',69.98
8202,'8/2/2022',76.11
8203,'8/8/2022',25.12
8204,'8/19/2022',46.23
8205,'9/26/2022',84.21
8206,'10/14/2022',96.24
8207,'10/29/2022',67.67
];
```

Результаты

Загрузите данные и откройте лист. Создайте новую таблицу и добавьте эти поля как измерения:

- `date`
- `previous_qtr_to_date`

Результирующая таблица

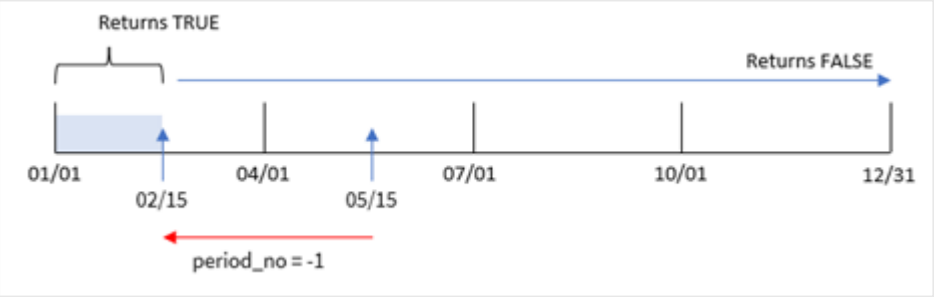
date	previous_qtr_to_date
1/7/2022	-1
1/19/2022	-1
2/5/2022	-1
2/28/2022	0
3/16/2022	0
4/1/2022	0
5/7/2022	0
5/16/2022	0
6/15/2022	0
6/26/2022	0
7/9/2022	0

date	previous_qtr_to_date
7/22/2022	0
7/23/2022	0
7/27/2022	0
8/2/2022	0
8/8/2022	0
8/19/2022	0
9/26/2022	0
10/14/2022	0
10/29/2022	0

Используя `-1` в качестве аргумента `period_no`, функция `inmonthstodate()` сдвигает границы сопоставляемого сегмента года на один квартал.

15 мая приходится на второй квартал года, поэтому этот сегмент изначально соответствует периоду с 1 апреля по 15 мая. Аргумент `period_no` сдвигает этот сегмент на три месяца назад. Границы диапазона дат переносятся на 1 января и 15 февраля.

Диаграмма функции `inmonthstodate` с аргументом period_no = -1.



Таким образом, любая транзакция, совершенная в период с 1 января по 15 февраля, возвращает булев результат TRUE.

Пример 3. Аргумент first_month_of_year

Скрипт загрузки и результаты

Обзор

Используется тот же набор данных и сценарий, что в первом примере.

В этом примере политика организации устанавливает март в качестве первого месяца финансового года.

Создайте поле in_months_to_date, которое определяет, какие транзакции совершены в том же квартале до 15 мая 2022 года.

Скрипт загрузки

```
SET DateFormat='MM/DD/YYYY';

Transactions:
    Load
    *,
    inmonthstodate(3,date,'05/15/2022', 0,3) as in_months_to_date
    ;
Load
*
Inline
[
id,date,amount
8188,'1/19/2022',37.23
8189,'1/7/2022',17.17
8190,'2/28/2022',88.27
8191,'2/5/2022',57.42
8192,'3/16/2022',53.80
8193,'4/1/2022',82.06
8194,'5/7/2022',40.39
8195,'5/16/2022',87.21
8196,'6/15/2022',95.93
8197,'6/26/2022',45.89
8198,'7/9/2022',36.23
8199,'7/22/2022',25.66
8200,'7/23/2022',82.77
8201,'7/27/2022',69.98
8202,'8/2/2022',76.11
8203,'8/8/2022',25.12
8204,'8/19/2022',46.23
8205,'9/26/2022',84.21
8206,'10/14/2022',96.24
8207,'10/29/2022',67.67
];
```

Результаты

Загрузите данные и откройте лист. Создайте новую таблицу и добавьте эти поля как измерения:

- date
- in_months_to_date

Результирующая таблица

date	previous_qtr_to_date
1/7/2022	0
1/19/2022	0

date	previous_qtr_to_date
2/5/2022	0
2/28/2022	0
3/16/2022	-1
4/1/2022	-1
5/7/2022	-1
5/16/2022	0
6/15/2022	0
6/26/2022	0
7/9/2022	0
7/22/2022	0
7/23/2022	0
7/27/2022	0
8/2/2022	0
8/8/2022	0
8/19/2022	0
9/26/2022	0
10/14/2022	0
10/29/2022	0

Так как используется `3` в качестве аргумента `first_month_of_year`, функция `inmonthstodate()` начинает год 1 марта и делит его на кварталы на основе первого переданного аргумента. Следовательно, имеются следующие квартальные сегменты:

- Mar-May (март-май)
- Jun-Aug (июнь-август)
- Sep-Nov (сентябрь-ноябрь)
- Dec-Feb (декабрь-февраль)

Аргумент `base_date` = 15 мая затем сегментирует квартал с марта по май, устанавливая для него конечную границу 15 мая.

Диаграмма функции `inmonthstodate`, где в качестве первого месяца года задан март.

Таким образом, для любой транзакции, совершенной между 1 марта и 15 мая, возвращается булев результат TRUE, а для транзакций с датами вне этих границ – значение FALSE.

Пример 4. Пример диаграммы

Скрипт загрузки и выражение диаграммы

Обзор

Используется тот же набор данных и сценарий, что в первом примере.

В этом примере в приложение загружается неизмененный набор данных. Поставлена задача создать расчет, определяющий, произошли ли транзакции в квартале, к которому относится 15 мая, в качестве меры в диаграмме приложения.

Скрипт загрузки

```
SET DateFormat='MM/DD/YYYY';

Transactions:
Load
*
Inline
[
id,date,amount
8188,'1/19/2022',37.23
8189,'1/7/2022',17.17
8190,'2/28/2022',88.27
8191,'2/5/2022',57.42
8192,'3/16/2022',53.80
8193,'4/1/2022',82.06
8194,'5/7/2022',40.39
8195,'5/16/2022',87.21
8196,'6/15/2022',95.93
8197,'6/26/2022',45.89
8198,'7/9/2022',36.23
8199,'7/22/2022',25.66
8200,'7/23/2022',82.77
8201,'7/27/2022',69.98
8202,'8/2/2022',76.11
8203,'8/8/2022',25.12
8204,'8/19/2022',46.23
8205,'9/26/2022',84.21
```

```
8206,'10/14/2022',96.24
8207,'10/29/2022',67.67
];
```

Результаты

Загрузите данные и откройте лист. Создайте новую таблицу и добавьте это поле как измерение:

```
date
```

Чтобы рассчитать, совершены ли транзакции в том же квартале, к которому относится 15 мая, создайте следующую меру:

```
=inmonthstodate(3,date,'05/15/2022', 0)
```

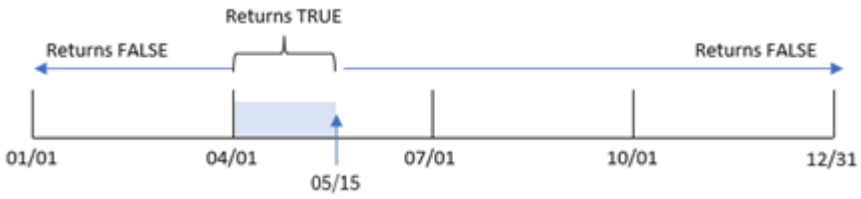
Результирующая таблица

date	=inmonthstodate(3,date,'05/15/2022', 0)
1/7/2022	0
1/19/2022	0
2/5/2022	0
2/28/2022	0
3/16/2022	0
4/1/2022	-1
5/7/2022	-1
5/16/2022	0
6/15/2022	0
6/26/2022	0
7/9/2022	0
7/22/2022	0
7/23/2022	0
7/27/2022	0
8/2/2022	0
8/8/2022	0
8/19/2022	0
9/26/2022	0
10/14/2022	0
10/29/2022	0

Мера ‘`in_months_to_date` создается в диаграмме с помощью функции `inmonthstodate()`.

Первым предоставлен аргумент `3`, который делит год на квартальные сегменты. Второй аргумент определяет, какое поле оценивается. Третий аргумент – это жестко закодированная дата 15 мая, которая представляет собой `base_date`, определяющий конечную границу этого сегмента. Заключительный аргумент – `period_no = 0`.

Диаграмма функции `inmonthstodate` с квартальными сегментами.



Любая транзакция, совершенная в период с 1 апреля по 15 мая, возвращает булев результат TRUE. Даты транзакций за пределами этого сегмента будут возвращать FALSE.

Пример 5. Сценарий

Скрипт загрузки и результаты

Обзор

В этом примере набор данных загружается в таблицу под именем `Sales`. Данная таблица содержит следующие поля:

- Product ID
- Product Type
- Sales date (Дата продажи)
- Sales price (Стоимость продажи)

Конечному пользователю нужна диаграмма, в которой по типу продукта отображаются объемы продаж продукта за период, предшествующий 24 декабря 2022 года. Пользователю требуется возможность определить продолжительность этого периода.

Скрипт загрузки

```
SET vPeriod = 1;

Products:
Load
*
Inline
[
product_id,product_type,sales_date,sales_price
8188,product A,'9/19/2022',37.23
8189,product D,'10/27/2022',17.17
8190,product C,'10/30/2022',88.27
8191,product B,'10/31/2022',57.42
8192,product D,'11/16/2022',53.80
8193,product D,'11/28/2022',82.06
```

```
8194,product A,'12/2/2022',40.39
8195,product B,'12/5/2022',87.21
8196,product C,'12/15/2022',95.93
8197,product B,'12/16/2022',45.89
8198,product C,'12/19/2022',36.23
8199,product D,'12/22/2022',25.66
8200,product D,'12/23/2022',82.77
8201,product A,'12/24/2022',69.98
8202,product A,'12/24/2022',76.11
8203,product B,'12/26/2022',25.12
8204,product B,'12/27/2022',46.23
8205,product B,'12/27/2022',84.21
8206,product C,'12/28/2022',96.24
8207,product D,'12/29/2022',67.67
];
```

Результаты

Загрузите данные и откройте лист.

В начале скрипта загрузки создана переменная `vPeriod`, которая будет привязана к элементу управления вводом переменной.

Выполните следующие действия.

1. На панели ресурсов щелкните **Пользовательские объекты**.
2. Выберите **Qlik Dashboard bundle** и добавьте **Ввод переменной** на лист.
3. Введите заголовок для диаграммы.
4. В разделе **Переменная** выберите **vPeriod** в качестве имени и задайте для отображения объект **Раскрывающийся список**.
5. В списке **Значения** выберите **Динамическое**. Введите следующее:
   `='1~month|2~bi-month|3~quarter|4~tertial|6~half-year'.`
6. Добавьте на лист новую таблицу.
7. На панели свойств в разделе **Данные** можно добавить измерение `product_type`.
8. Добавьте следующее выражение в качестве меры:
   `=sum(if(inmonthstodate($(vPeriod),sales_date,makedate(2022,12,24),0),sales_price,0))`
9. Задайте параметру **Формат чисел** меры значение **Денежный**.

Результирующая таблица

product_type	=sum(if(inmonthstodate($(vPeriod),sales_date,makedate(2022,12,24),0),sales_price,0))
product A	$186.48
product B	$190.52
product C	$220.43
product D	$261.46

Функция `inmonthstodate()` использует пользовательский ввод в качестве аргумента для определения размера начального сегмента года.

Функция передает дату продажи каждого продукта в качестве второго аргумента функции `inmonthstodate()`. В качестве третьего аргумента в функции `inmonthstodate()` используется 24 декабря, поэтому продукты, проданные в указанный период до 24 декабря включительно, возвращают булево значение TRUE. Функция sum суммирует продажи этих продуктов.

inmonthtodate

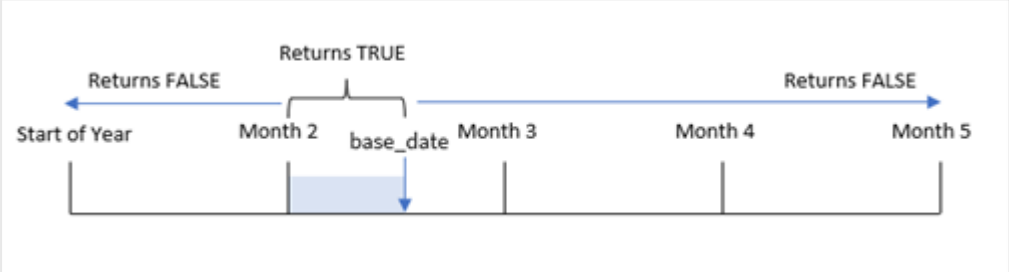
Возвращает значение True, если значение **date** находится в пределах части месяца, включающей значение, заданное в поле **basedate** до последней миллисекунды, указанной в поле **basedate**, включительно.

Синтаксис:

```
InMonthToDate (timestamp, base_date, period_no)
```

Возвращаемые типы данных: Булево значение

Схема функции `inmonthtodate`.



Функция `inmonthtodate()` идентифицирует выбранный месяц как сегмент. Начальной границей является начало месяца. В качестве конечной границы можно указать более позднюю дату месяца. Затем функция определяет, попадает набор дат в этот сегмент или нет, и возвращает булево значение TRUE или FALSE.

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.

Когда это следует использовать

Функция `inmonthtodate()` возвращает результат в виде булева значения. Обычно этот тип функции используется в качестве условия в `if expression`. Функция `inmonthtodate()` возвращает агрегирование или вычисление в зависимости от того, выпадает ли дата на указанный месяц, включая рассматриваемую дату.

Например, функцию `inmonthtodate()` можно использовать для идентификации всего оборудования, изготовленного в указанном месяце до определенной даты.

Региональные настройки

Если не указано иное, в примерах, приведенных в данном разделе, используется следующий формат даты: MM/ДД/ГГГГ. Формат даты указан в операторе `SET DateFormat` скрипта загрузки данных. В вашей системе может быть установлен другой формат даты по умолчанию в зависимости от региональных настроек и других факторов. Можно изменить формат в примерах в соответствии с потребностями. Или можно изменить форматы в скрипте загрузки в соответствии с этими примерами.

Региональные настройки по умолчанию в приложениях основаны на системных региональных настройках компьютера или сервера, где установлено ПО Qlik Sense. Если на сервере Qlik Sense, к которому обращается пользователь, выбран шведский язык, то редактор загрузки данных будет использовать шведские региональные настройки для даты, времени и валюты. Эти параметры регионального формата не связаны с языком, отображаемым в интерфейсе пользователя Qlik Sense. Qlik Sense будет отображаться на языке, который используется в браузере.

Примеры функции

Пример	Результат
`inmonthtodate ('01/25/2013', '25/01/2013', 0)`	Возвращает True
`inmonthtodate ('01/25/2013', '24/01/2013', 0)`	Возвращает False
`inmonthtodate ('01/25/2013', '28/02/2013', -1)`	Возвращает True

Пример 1. Без дополнительных аргументов

Скрипт загрузки и результаты

Обзор

Откройте редактор загрузки данных и добавьте приведенный ниже скрипт загрузки на новую вкладку.

Скрипт загрузки содержит следующее:

- Набор данных, содержащий набор транзакций за 2022 год, загруженный в таблицу под именем `Transactions`.
- Поле даты, предоставленное в формате системной переменной `DateFormat` (`MM/DD/YYYY`).

- Предыдущий оператор load, который содержит следующее:
    - Функция `inmonthtodate()`, заданная как поле `in_month_to_date`. Она определяет, какие транзакции совершены с 1 по 26 июля 2022 года.

Скрипт загрузки

```
SET DateFormat='MM/DD/YYYY';

Transactions:
    Load
    *,
    inmonthtodate(date,'07/26/2022', 0) as in_month_to_date
    ;
Load
*
Inline
[
id,date,amount
8188,'1/19/2022',37.23
8189,'1/7/2022',17.17
8190,'2/28/2022',88.27
8191,'2/5/2022',57.42
8192,'3/16/2022',53.80
8193,'4/1/2022',82.06
8194,'5/7/2022',40.39
8195,'5/16/2022',87.21
8196,'6/15/2022',95.93
8197,'6/26/2022',45.89
8198,'7/9/2022',36.23
8199,'7/22/2022',25.66
8200,'7/23/2022',82.77
8201,'7/27/2022',69.98
8202,'8/2/2022',76.11
8203,'8/8/2022',25.12
8204,'8/19/2022',46.23
8205,'9/26/2022',84.21
8206,'10/14/2022',96.24
8207,'10/29/2022',67.67
];
```

Результаты

Загрузите данные и откройте лист. Создайте новую таблицу и добавьте эти поля как измерения:

- `date`
- `in_month_to_date`

Результирующая таблица

date	in_month_to_date
1/7/2022	0

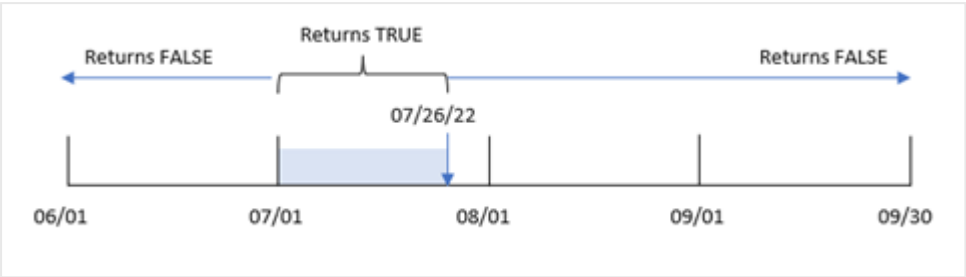
date	in_month_to_date
1/19/2022	0
2/5/2022	0
2/28/2022	0
3/16/2022	0
4/1/2022	0
5/7/2022	0
5/16/2022	0
6/15/2022	0
6/26/2022	0
7/9/2022	-1
7/22/2022	-1
7/23/2022	-1
7/27/2022	0
8/2/2022	0
8/8/2022	0
8/19/2022	0
9/26/2022	0
10/14/2022	0
10/29/2022	0

Поле `in_month_to_date` создано в предыдущем операторе load с помощью функции `inmonthtodate()`.

Первый аргумент определяет, какое поле оценивается. Второй аргумент – это жестко закодированная дата, 26 июля, которая выступает в роли `base_date`. Этот аргумент `base_date` определяет сегментированный месяц, а также конечную границу оцениваемого сегмента.

`period_no = 0` – это последний аргумент, означающий, что функция не сравнивает месяцы, которые следуют до или после сегментированного месяца.

Диаграмма функции `inmonthtodate` без дополнительных аргументов



В результате все транзакции, совершенные в период с 1 по 26 июля, возвращают булево значение TRUE. Все транзакции, совершенные в июле после 26 числа, возвращают булево значение FALSE, как и транзакции, совершенные в другие месяцы года.

Пример 2. Скрипт period_no

Скрипт загрузки и результаты

Обзор

Используется тот же набор данных и сценарий, что в первом примере.

В этом примере стоит задача создать поле `six_months_prior`, которое определяет, какие транзакции совершены за шесть полных месяцев до периода с 1 по 26 июля.

Скрипт загрузки

```
SET DateFormat='MM/DD/YYYY';

Transactions:
    Load
    *,
    inmonthtodate(date,'07/26/2022', -6) as six_months_prior
    ;
Load
*
Inline
[
id,date,amount
8188,'1/19/2022',37.23
8189,'1/7/2022',17.17
8190,'2/28/2022',88.27
8191,'2/5/2022',57.42
8192,'3/16/2022',53.80
8193,'4/1/2022',82.06
8194,'5/7/2022',40.39
8195,'5/16/2022',87.21
8196,'6/15/2022',95.93
8197,'6/26/2022',45.89
8198,'7/9/2022',36.23
8199,'7/22/2022',25.66
8200,'7/23/2022',82.77
```

```
8201,'7/27/2022',69.98
8202,'8/2/2022',76.11
8203,'8/8/2022',25.12
8204,'8/19/2022',46.23
8205,'9/26/2022',84.21
8206,'10/14/2022',96.24
8207,'10/29/2022',67.67
];
```

Результаты

Загрузите данные и откройте лист. Создайте новую таблицу и добавьте эти поля как измерения:

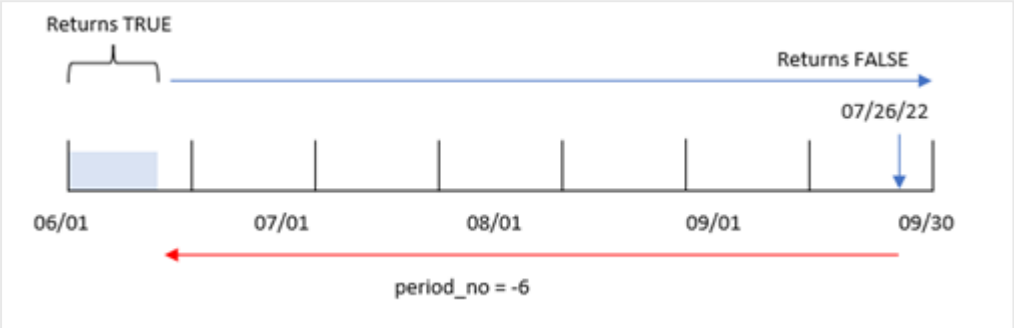
- `date`
- `six_months_prior`

Результирующая таблица

date	six_months_prior
1/7/2022	-1
1/19/2022	-1
2/5/2022	0
2/28/2022	0
3/16/2022	0
4/1/2022	0
5/7/2022	0
5/16/2022	0
6/15/2022	0
6/26/2022	0
7/9/2022	0
7/22/2022	0
7/23/2022	0
7/27/2022	0
8/2/2022	0
8/8/2022	0
8/19/2022	0
9/26/2022	0
10/14/2022	0
10/29/2022	0

Используя `-6` в качестве аргумента `period_no`, функция `inmonthtodate()` сдвигает границы сравниваемого сегмента месяца на шесть месяцев. Сначала сегмент месяца начинается 1 июля и заканчивается 26 июля. Затем `period_no` сдвигает этот сегмент назад на шесть месяцев, в результате чего новыми границами сегмента становятся 1 января и 26 января.

Диаграмма функции `inmonthtodate` с аргументом period_no = -6.



В результате все транзакции, совершенные в период с 1 по 26 января, возвращают булево значение TRUE.

Пример 3. Пример диаграммы

Скрипт загрузки и выражение диаграммы

Обзор

Используется тот же набор данных и сценарий, что в первом примере.

В этом примере в приложение загружается неизмененный набор данных. Цель — создать расчет, определяющий, произошли ли транзакции с 1 по 26 июля, как меру в диаграмме приложения.

Скрипт загрузки

```
SET DateFormat='MM/DD/YYYY';

Transactions:
Load
*
Inline
[
id,date,amount
8188,'1/19/2022',37.23
8189,'1/7/2022',17.17
8190,'2/28/2022',88.27
8191,'2/5/2022',57.42
8192,'3/16/2022',53.80
8193,'4/1/2022',82.06
8194,'5/7/2022',40.39
8195,'5/16/2022',87.21
8196,'6/15/2022',95.93
8197,'6/26/2022',45.89
8198,'7/9/2022',36.23
```

```
8199,'7/22/2022',25.66
8200,'7/23/2022',82.77
8201,'7/27/2022',69.98
8202,'8/2/2022',76.11
8203,'8/8/2022',25.12
8204,'8/19/2022',46.23
8205,'9/26/2022',84.21
8206,'10/14/2022',96.24
8207,'10/29/2022',67.67
];
```

Результаты

Загрузите данные и откройте лист. Создайте новую таблицу и добавьте это поле как измерение:

```
date
```

Чтобы рассчитать, совершены ли транзакции с 1 по 26 июля, создайте следующую меру:

```
=inmonthtodate(date,'07/26/2022', 0)
```

Результирующая таблица

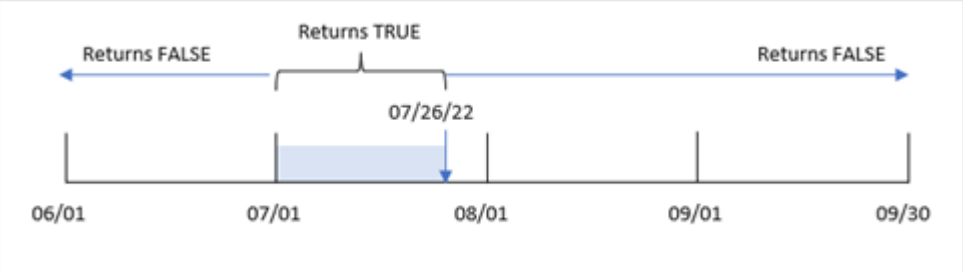
date	=inmonthtodate(date,'07/26/2022', 0)
1/7/2022	0
1/19/2022	0
2/5/2022	0
2/28/2022	0
3/16/2022	0
4/1/2022	0
5/7/2022	0
5/16/2022	0
6/15/2022	0
6/26/2022	0
7/9/2022	-1
7/22/2022	-1
7/23/2022	-1
7/27/2022	0
8/2/2022	0
8/8/2022	0
8/19/2022	0

date	=inmonthtodate(date,'07/26/2022', 0)
9/26/2022	0
10/14/2022	0
10/29/2022	0

Мера для поля `in_month_to_date` создается в диаграмме с помощью функции `inmonthtodate()`.

Первый аргумент определяет, какое поле оценивается. Второй аргумент – это жестко закодированная дата, 26 июля, которая выступает в роли `base_date`. Этот аргумент base_date определяет сегментированный месяц, а также конечную границу оцениваемого сегмента. Заключительный аргумент – `period_no` = 0. Это значит, что функция не сравнивает месяцы, которые следуют до или после сегментированного месяца.

Диаграмма функции `inmonthtodate` без дополнительных аргументов



В результате все транзакции, совершенные в период с 1 по 26 июля, возвращают булево значение TRUE. Все транзакции, совершенные в июле после 26 числа, возвращают булево значение FALSE, как и транзакции, совершенные в другие месяцы года.

Пример 4. Сценарий

Скрипт загрузки и результаты

Обзор

В этом примере набор данных загружается в таблицу под именем `Products`. Данная таблица содержит следующие поля:

- ИД продукта
- Время изготовления
- Себестоимость

Из-за ошибки оборудования продукты, произведенные в июле 2022 года, оказались бракованными. Проблема была решена 27 июля 2022 года.

Конечному пользователю требуется объект диаграммы, который отображает по месяцу, какие изготовленные продукты были дефектными, а какие бездефектными, а также стоимость продуктов, изготовленных в этом месяце.

Скрипт загрузки

```
Products:
Load
*
Inline
[
product_id,manufacture_date,cost_price
8188,'1/19/2022',37.23
8189,'1/7/2022',17.17
8190,'2/28/2022',88.27
8191,'2/5/2022',57.42
8192,'3/16/2022',53.80
8193,'4/1/2022',82.06
8194,'5/7/2022',40.39
8195,'5/16/2022',87.21
8196,'6/15/2022',95.93
8197,'6/26/2022',45.89
8198,'7/9/2022',36.23
8199,'7/22/2022',25.66
8200,'7/23/2022',82.77
8201,'7/27/2022',69.98
8202,'8/2/2022',76.11
8203,'8/8/2022',25.12
8204,'8/19/2022',46.23
8205,'9/26/2022',84.21
8206,'10/14/2022',96.24
8207,'10/29/2022',67.67
];
```

Результаты

Загрузите данные и откройте лист. Создайте новую таблицу и добавьте эти поля как измерения:

- `=monthname(manufacture_date)`
- `=if(Inmonthtodate(manufacture_date,makedate(2022,07,26),0),'Defective','Faultless')`

Чтобы вычислить совокупную стоимость продуктов, создайте следующую меру:

```
=sum(cost_price)
```

Задайте параметру **Формат чисел** меры значение **Денежный**.

Результирующая таблица

monthname (manufacture_date)	if(Inmonthtodate(manufacture_date,makedate (2022,07,26),0),'Defective','Faultless')	Sum(cost_ price)
Jan 2022	Бездефектный	$54.40
Feb 2022	Бездефектный	$145.69
Mar 2022	Бездефектный	$53.80

monthname (manufacture_date)	if(Inmonthtodate(manufacture_date,makedate (2022,07,26),0),'Defective','Faultless')	Sum(cost_ price)
Apr 2022	Бездефектный	$82.06
May 2022	Бездефектный	$127.60
Jun 2022	Бездефектный	$141.82
Jul 2022	Дефектный	$144.66
Jul 2022	Бездефектный	$69.98
Aug 2022	Бездефектный	$147.46
Sep 2022	Бездефектный	$84.21
Oct 2022	Бездефектный	$163.91

Функция `inmonthtodate()` возвращает булево значение при проверке дат производства каждого продукта.

Для дат, которые возвращают булево значение TRUE, продукт отмечается как «Дефектный». Любой продукт, которое возвращает значение FALSE, то есть изготовлен не с 1 по 26 июля включительно, отмечается как «Бездефектный».

inquarter

Эта функция возвращает значение True, если поле **timestamp** находится в пределах квартала, включающего значение, указанное в поле **base_date**.

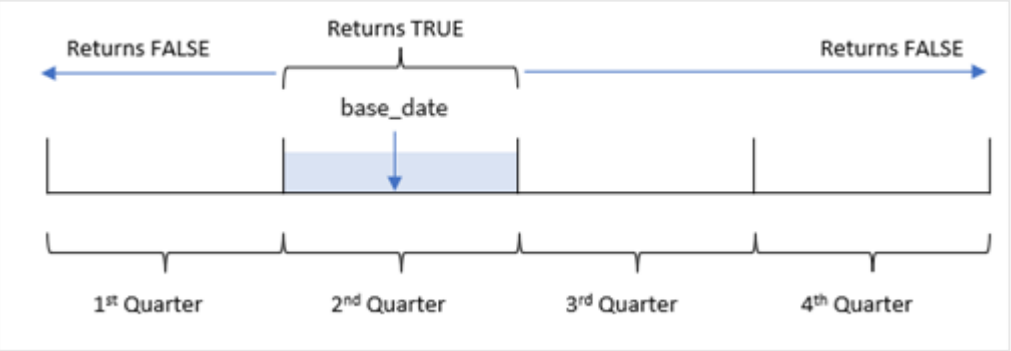
Синтаксис:

```
InQuarter (timestamp, base_date, period_no[, first_month_of_year])
```

Возвращаемые типы данных: Булево значение

В Qlik Sense логическое значение «истина» представлено как -1, а «ложь» – как 0.

Диаграмма диапазона функции `inquarter()`



Другими словами, функция `inquarter()` делит год на четыре одинаковых квартала с 1 января по 31 декабря. С помощью аргумента `first_month_of_year` можно изменить месяц, который приложение считает первым, и сегментирование на кварталы изменится с учетом этого аргумента. Функция `base_date` определяет, какой квартал необходимо использовать для сравнения. Наконец, функция возвращает булев результат при сравнении значений даты с этим квартальным сегментом.

Когда это следует использовать

Функция `inquarter()` возвращает результат в виде булева значения. Обычно этот тип функции используется в качестве условия в `if expression`. Она возвращает агрегирование или вычисление в зависимости от того, попадает ли дата в указанный квартал.

Например, с помощью функции `inquarter()` можно идентифицировать все оборудование, изготовленное в квартальном сегменте, на основании даты производства оборудования.

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**.

Можно использовать следующие значения, чтобы задать первый месяц года в аргументе `first_month_of_year`:

значения first_month_of_year

Месяц	Значение
Февраль	2
Март	3
Апрель	4
Мау	5
Июнь	6
Июль	7
Август	8
Сентябрь	9
Октябрь	10

Месяц	Значение
Ноябрь	11
Декабрь	12

Региональные настройки

Если не указано иное, в примерах, приведенных в данном разделе, используется следующий формат даты: ММ/ДД/ГГГГ. Формат даты указан в операторе `SET DateFormat` скрипта загрузки данных. В вашей системе может быть установлен другой формат даты по умолчанию в зависимости от региональных настроек и других факторов. Можно изменить формат в примерах в соответствии с потребностями. Или можно изменить форматы в скрипте загрузки в соответствии с этими примерами.

Региональные настройки по умолчанию в приложениях основаны на системных региональных настройках компьютера или сервера, где установлено ПО Qlik Sense. Если на сервере Qlik Sense, к которому обращается пользователь, выбран шведский язык, то редактор загрузки данных будет использовать шведские региональные настройки для даты, времени и валюты. Эти параметры регионального формата не связаны с языком, отображаемым в интерфейсе пользователя Qlik Sense. Qlik Sense будет отображаться на языке, который используется в браузере.

Примеры функции

Пример	Результат
inquarter ('01/25/2013', '01/01/2013', 0)	Возвращает TRUE
inquarter ('01/25/2013', '04/01/2013', 0)	Возвращает FALSE
inquarter ('01/25/2013', '01/01/2013', -1)	Возвращает FALSE
inquarter ('12/25/2012', '01/01/2013', -1)	Возвращает TRUE
inquarter ('01/25/2013', '03/01/2013', 0, 3)	Возвращает FALSE
inquarter ('03/25/2013', '03/01/2013', 0, 3)	Возвращает TRUE

Пример 1. Без дополнительных аргументов

Скрипт загрузки и результаты

Обзор

Откройте редактор загрузки данных и добавьте приведенный ниже скрипт загрузки на новую вкладку.

Скрипт загрузки содержит следующее:

- Набор данных, содержащий набор транзакций за 2022 год, загруженный в таблицу под именем `Transactions`.

- Предыдущая загрузка с функцией `inquarter()`, которая задана в качестве поля `in_quarter` и определяет, какие транзакции совершены в том же квартале, к которому относится 15 мая 2022 года.

Скрипт загрузки

```
SET DateFormat='MM/DD/YYYY';

Transactions:
    Load
       *,
       inquarter (date,'05/15/2022', 0) as in_quarter
       ;
Load
*
Inline
[
id,date,amount
8188,'1/19/2022',37.23
8189,'1/7/2022',17.17
8190,'2/28/2022',88.27
8191,'2/5/2022',57.42
8192,'3/16/2022',53.80
8193,'4/1/2022',82.06
8194,'5/7/2022',40.39
8195,'5/16/2022',87.21
8196,'6/15/2022',95.93
8197,'6/26/2022',45.89
8198,'7/9/2022',36.23
8199,'7/22/2022',25.66
8200,'7/23/2022',82.77
8201,'7/27/2022',69.98
8202,'8/2/2022',76.11
8203,'8/8/2022',25.12
8204,'8/19/2022',46.23
8205,'9/26/2022',84.21
8206,'10/14/2022',96.24
8207,'10/29/2022',67.67
];
```

Результаты

Загрузите данные и откройте лист. Создайте новую таблицу и добавьте эти поля как измерения:

- `date`
- `in_quarter`

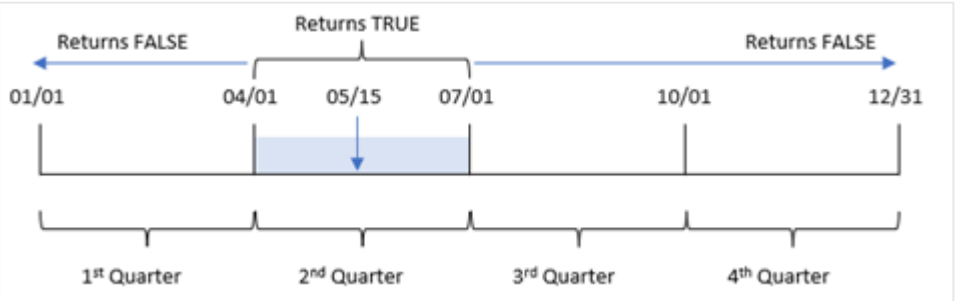
Результирующая таблица

date	in_quarter
1/7/2022	0

date	in_quarter
1/19/2022	0
2/5/2022	0
2/28/2022	0
3/16/2022	0
4/1/2022	-1
5/7/2022	-1
5/16/2022	-1
6/15/2022	-1
6/26/2022	-1
7/9/2022	0
7/22/2022	0
7/23/2022	0
7/27/2022	0
8/2/2022	0
8/8/2022	0
8/19/2022	0
9/26/2022	0
10/14/2022	0
10/29/2022	0

Поле `in_quarter` создано в предыдущем операторе load с помощью функции `inquarter()`. Первый аргумент определяет, какое поле оценивается. Второй аргумент — это жестко закодированная дата, 15 мая, которая определяет, какой квартал необходимо использовать для сравнения. `period_no = 0` — это последний аргумент, означающий, что функция `inquarter()` не сравнивает кварталы, которые следуют до или после сегментированного квартала.

Диаграмма функции `inquarter()` с базовой датой 15 мая



Любая транзакция, совершенная в период с 1 апреля по 30 июня, возвращает булев результат TRUE.

Пример 2. Скрипт period_no

Скрипт загрузки и результаты

Обзор

Откройте редактор загрузки данных и добавьте приведенный ниже скрипт загрузки на новую вкладку.

Скрипт загрузки содержит следующее:

- Набор данных, содержащий набор транзакций за 2022 год, загруженный в таблицу под именем `Transactions`.
- Предыдущая загрузка с функцией `inquarter()`, которая задана в качестве поля `previous_quarter` и определяет, какие транзакции совершены в квартале, предшествующем кварталу, к которому относится 15 мая 2022 года.

Скрипт загрузки

```
SET DateFormat='MM/DD/YYYY';

Transactions:
    Load
       *,
       inquarter (date,'05/15/2022', -1) as previous_qtr
       ;
Load
*
Inline
[
id,date,amount
8188,'1/19/2022',37.23
8189,'1/7/2022',17.17
8190,'2/28/2022',88.27
8191,'2/5/2022',57.42
8192,'3/16/2022',53.80
8193,'4/1/2022',82.06
8194,'5/7/2022',40.39
8195,'5/16/2022',87.21
8196,'6/15/2022',95.93
8197,'6/26/2022',45.89
8198,'7/9/2022',36.23
8199,'7/22/2022',25.66
8200,'7/23/2022',82.77
8201,'7/27/2022',69.98
8202,'8/2/2022',76.11
8203,'8/8/2022',25.12
8204,'8/19/2022',46.23
8205,'9/26/2022',84.21
8206,'10/14/2022',96.24
```

```
8207,'10/29/2022',67.67
];
```

Результаты

Загрузите данные и откройте лист. Создайте новую таблицу и добавьте эти поля как измерения:

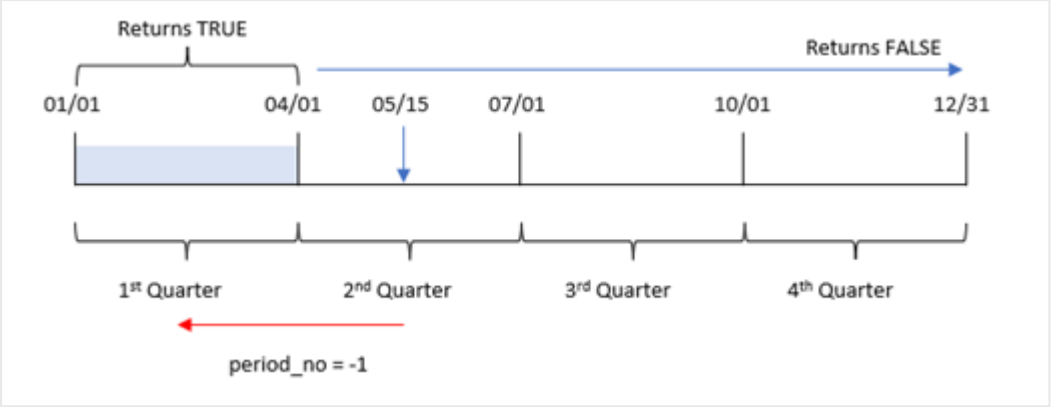
- `date`
- `previous_qtr`

Результирующая таблица

date	previous_qtr
1/7/2022	-1
1/19/2022	-1
2/5/2022	-1
2/28/2022	-1
3/16/2022	-1
4/1/2022	0
5/7/2022	0
5/16/2022	0
6/15/2022	0
6/26/2022	0
7/9/2022	0
7/22/2022	0
7/23/2022	0
7/27/2022	0
8/2/2022	0
8/8/2022	0
8/19/2022	0
9/26/2022	0
10/14/2022	0
10/29/2022	0

Используя `-1` в качестве аргумента `period_no`, функция `inquarter()` сдвигает границы сравниваемого квартала на один полный квартал назад. 15 мая приходится на второй квартал года, поэтому этот сегмент изначально соответствует периоду с 1 апреля по 30 июня. `period_no` сдвигает этот сегмент на три месяца назад, в результате чего границы диапазона дат переносятся на 1 января и 30 марта.

Диаграмма функции `inquarter()` с базовой датой 15 мая



Таким образом, любая транзакция, совершенная в период с 1 января по 30 марта, возвращает булев результат TRUE.

Пример 3. Аргумент first_month_of_year

Скрипт загрузки и результаты

Обзор

Откройте редактор загрузки данных и добавьте приведенный ниже скрипт загрузки на новую вкладку.

Скрипт загрузки содержит следующее:

- Набор данных, содержащий набор транзакций за 2022 год, загруженный в таблицу под именем `Transactions`.
- Предыдущая загрузка с функцией `inquarter()`, которая задана в качестве поля `in_quarter` и определяет, какие транзакции совершены в том же квартале, к которому относится 15 мая 2022 года.

Однако в этом примере политика организации устанавливает март в качестве первого месяца финансового года.

Скрипт загрузки

```
SET DateFormat='MM/DD/YYYY';

Transactions:
    Load
       *,
       inquarter (date,'05/15/2022', 0, 3) as in_quarter
       ;
Load
*
Inline
[
id,date,amount
```

```
8188,'1/19/2022',37.23
8189,'1/7/2022',17.17
8190,'2/28/2022',88.27
8191,'2/5/2022',57.42
8192,'3/16/2022',53.80
8193,'4/1/2022',82.06
8194,'5/7/2022',40.39
8195,'5/16/2022',87.21
8196,'6/15/2022',95.93
8197,'6/26/2022',45.89
8198,'7/9/2022',36.23
8199,'7/22/2022',25.66
8200,'7/23/2022',82.77
8201,'7/27/2022',69.98
8202,'8/2/2022',76.11
8203,'8/8/2022',25.12
8204,'8/19/2022',46.23
8205,'9/26/2022',84.21
8206,'10/14/2022',96.24
8207,'10/29/2022',67.67
];
```

Результаты

Загрузите данные и откройте лист. Создайте новую таблицу и добавьте эти поля как измерения:

- `date`
- `previous_qtr`

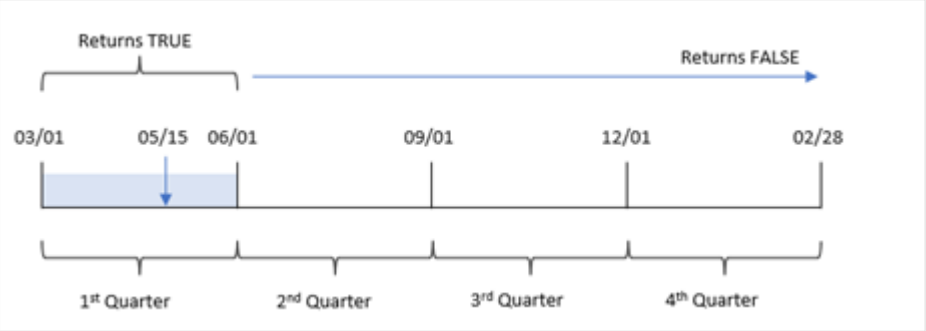
Результирующая таблица

date	previous_qtr
1/7/2022	0
1/19/2022	0
2/5/2022	0
2/28/2022	0
3/16/2022	-1
4/1/2022	-1
5/7/2022	-1
5/16/2022	-1
6/15/2022	0
6/26/2022	0
7/9/2022	0
7/22/2022	0

date	previous_qtr
7/23/2022	0
7/27/2022	0
8/2/2022	0
8/8/2022	0
8/19/2022	0
9/26/2022	0
10/14/2022	0
10/29/2022	0

Так как используется `3` в качестве аргумента `first_month_of_year`, функция `inquarter()` начинает год 1 марта, а затем делит год на кварталы. Таким образом, рассматриваются кварталы Mar-May (март-май), Jun-Aug (июнь-август), Sep-Nov (сентябрь-ноябрь), Dec-Feb (декабрь-февраль). Аргумент `base_date` = 15 мая задает квартал Mar-May в качестве квартала сравнения для функции.

Диаграмма функции `inquarter()`, где в качестве первого месяца года задан март.



Таким образом, любая транзакция, совершенная в период с 1 марта по 31 мая, возвращает булев результат TRUE.

Пример 4. Пример объекта диаграммы

Скрипт загрузки и выражение диаграммы

Обзор

Откройте редактор загрузки данных и добавьте приведенный ниже скрипт загрузки на новую вкладку.

Скрипт загрузки содержит следующее:

- Набор данных, содержащий набор транзакций за 2022 год, загруженный в таблицу под именем `Transactions`.

- Предыдущая загрузка с функцией `inquarter()`, которая задана в качестве поля `in_quarter` и определяет, какие транзакции совершены в том же квартале, к которому относится 15 мая 2022 года.

Скрипт загрузки

```
SET DateFormat='MM/DD/YYYY';

Transactions:
Load
*
Inline
[
id,date,amount
8188,'1/19/2022',37.23
8189,'1/7/2022',17.17
8190,'2/28/2022',88.27
8191,'2/5/2022',57.42
8192,'3/16/2022',53.80
8193,'4/1/2022',82.06
8194,'5/7/2022',40.39
8195,'5/16/2022',87.21
8196,'6/15/2022',95.93
8197,'6/26/2022',45.89
8198,'7/9/2022',36.23
8199,'7/22/2022',25.66
8200,'7/23/2022',82.77
8201,'7/27/2022',69.98
8202,'8/2/2022',76.11
8203,'8/8/2022',25.12
8204,'8/19/2022',46.23
8205,'9/26/2022',84.21
8206,'10/14/2022',96.24
8207,'10/29/2022',67.67
];
```

Результаты

Загрузите данные и откройте лист. Создайте новую таблицу и добавьте это поле как измерение:

- `date`

Создайте следующую меру, чтобы рассчитать, совершены ли транзакции в том же квартале, к которому относится 15 мая:

```
=inquarter(date,'05/15/2022', 0)
```

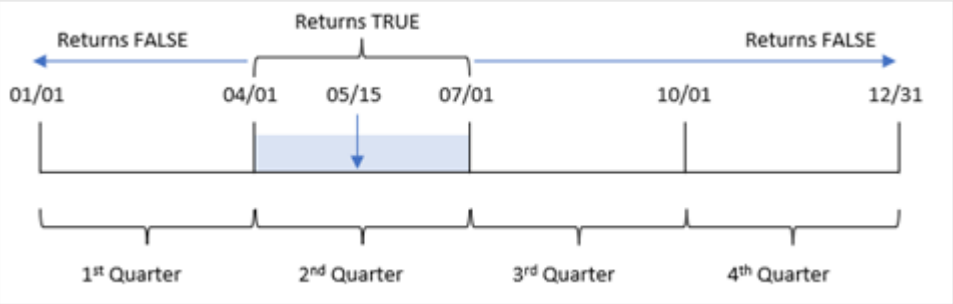
Результирующая таблица

date	in_quarter
1/7/2022	0
1/19/2022	0

date	in_quarter
2/5/2022	0
2/28/2022	0
3/16/2022	0
4/1/2022	-1
5/7/2022	-1
5/16/2022	-1
6/15/2022	-1
6/26/2022	-1
7/9/2022	0
7/22/2022	0
7/23/2022	0
7/27/2022	0
8/2/2022	0
8/8/2022	0
8/19/2022	0
9/26/2022	0
10/14/2022	0
10/29/2022	0

Мера `in_quarter` создается в диаграмме с помощью функции `inquarter()`. Первый аргумент определяет, какое поле оценивается. Второй аргумент — это жестко закодированная дата, 15 мая, которая определяет, какой квартал необходимо использовать для сравнения. `period_no = 0` — это последний аргумент, означающий, что функция `inquarter()` не сравнивает кварталы, которые следуют до или после сегментированного квартала.

Диаграмма функции `inquarter()` с базовой датой 15 мая



Любая транзакция, совершенная в период с 1 апреля по 30 июня, возвращает булев результат TRUE.

Пример 5. Сценарий

Скрипт загрузки и результаты

Обзор

Откройте редактор загрузки данных и добавьте приведенный ниже скрипт загрузки на новую вкладку.

Скрипт загрузки содержит следующее:

- Набор данных, загруженный в таблицу под именем `Products`.
- Данная таблица содержит следующие поля:
    - product ID (ИД продукта)
    - product type (тип продукта)
    - manufacture date (дата изготовления)
    - cost price (себестоимость)

Установлено, что вследствие ошибки оборудования продукты, изготовленные в течение недели, на которую выпадает 15 мая 2022 года, являются дефектными. Конечному пользователю требуется диаграмма, которая отображает по имени квартала, какие изготовленные продукты были дефектными, а какие бездефектными, а также стоимость продуктов, изготовленных в этом квартале.

Скрипт загрузки

```
Products:
Load
*
Inline
[
product_id,manufacture_date,cost_price
8188,'1/19/2022',37.23
8189,'1/7/2022',17.17
8190,'2/28/2022',88.27
8191,'2/5/2022',57.42
8192,'3/16/2022',53.80
8193,'4/1/2022',82.06
8194,'5/7/2022',40.39
8195,'5/16/2022',87.21
8196,'6/15/2022',95.93
8197,'6/26/2022',45.89
8198,'7/9/2022',36.23
8199,'7/22/2022',25.66
8200,'7/23/2022',82.77
8201,'7/27/2022',69.98
8202,'8/2/2022',76.11
8203,'8/8/2022',25.12
8204,'8/19/2022',46.23
8205,'9/26/2022',84.21
8206,'10/14/2022',96.24
```

```
8207,'10/29/2022',67.67
];
```

Результаты

Загрузите данные и откройте лист. Создайте новую таблицу и добавьте это поле как измерение:

```
=quartername(manufacture_date)
```

Создайте следующие меры:

- `=if(only(InQuarter(manufacture_date,makedate(2022,05,15),0)),'Defective','Faultless')`, чтобы определить, какие из продуктов дефектные, а какие нет, с использованием функции `inquarter()`:
- `=sum(cost_price)`, чтобы показать суммарную стоимость всех продуктов.

Выполните следующие действия.

1. Задайте параметру **Формат чисел** меры значение **Денежный**.
2. В области **Вид** выключите параметр **Итоги**.

Результирующая таблица

quartername (manufacture_date)	=if(only(InQuarter(manufacture_date,makedate (2022,05,15),0)),'Defective','Faultless')	Sum(cost_ price)
Jan-Mar 2022	Бездефектный	253.89
Apr-Jun 2022	Дефектный	351.48
Jul-Sep 2022	Бездефектный	446.31
Oct-Dec 2022	Бездефектный	163.91

Функция `inquarter()` возвращает булево значение при проверке дат производства каждого продукта. Для любого продукта, изготовленного в течение квартала, к которому относится 15 мая, функция `inquarter()` возвращает булево значение TRUE и ставит отметку «Дефектный». Любой продукт, который возвращает значение FALSE, то есть изготовлен в другом квартале, отмечается как «Бездефектный».

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.

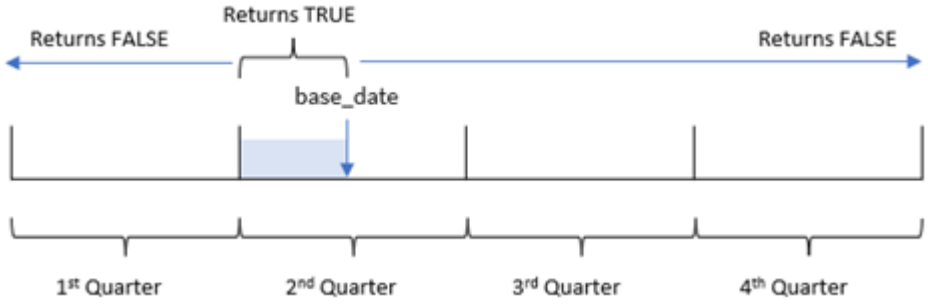
Синтаксис:

```
InQuarterToDate (timestamp, base_date, period_no [, first_month_of_year])
```

Возвращаемые типы данных: Булево значение

> *В Qlik Sense логическое значение «истина» представлено как -1, а «ложь» – как 0.*

Диаграмма функции inquartertodate



Функция `inquartertodate()` делит год на четыре равных квартала между 1 января и 31 декабря (или определяемым пользователем началом года и соответствующей ему датой окончания). Используя `base_date`, функция затем сегментирует конкретный квартал, а `base_date` определяет как квартал, так и максимально допустимую дату для этого сегмента квартала. Наконец, функция возвращает булев результат при сравнении предписанных значений даты с этим сегментом.

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**.

Когда это следует использовать

Функция `inquartertodate()` возвращает результат в виде логического значения. Обычно этот тип функции используется в качестве условия в выражении `if`. Функция `inquartertodate()` возвращает агрегирование или расчет в зависимости от того, попадает ли проверяемая дата в квартал до рассматриваемой даты включительно.

Например, функцию `inquartertodate()` можно использовать для идентификации всего оборудования, изготовленного в указанном квартале до определенной даты.

Примеры функции

Пример	Результат
`inquartertodate ('01/25/2013', '03/25/2013', 0)`	Возвращает `TRUE`, так как значение`timestamp`, 01/25/2013, попадает в трехмесячный период с 01/01/2013 по 03/25/2013, в который попадает значение `base_date`, 03/25/2013.
`inquartertodate ('04/26/2013', '03/25/2013', 0)`	Возвращает `FALSE`, так как 04/26/2013 находится за пределами периода, указанного в предыдущем примере.
`inquartertodate ('02/25/2013', '06/09/2013', -1)`	Возвращает `TRUE`, так как значение `period_no`, -1, сдвигает период поиска назад на один трехмесячный период (один квартал года). Таким образом, период поиска длится с 01/01/2013 по 03/09/2013.
`inquartertodate ('03/25/2006', '04/15/2006', 0, 2)`	Возвращает `TRUE`, так как значение `first_month_of_year`равно 2, в результате чего период поиска длится с 02/01/2006 по 04/15/2006, а не с 04/01/2006 по 04/15/2006.

Региональные настройки

Если не указано иное, в примерах, приведенных в данном разделе, используется следующий формат даты: ММ/ДД/ГГГГ. Формат даты указан в операторе `SET DateFormat` скрипта загрузки данных. В вашей системе может быть установлен другой формат даты по умолчанию в зависимости от региональных настроек и других факторов. Можно изменить формат в примерах в соответствии с потребностями. Или можно изменить форматы в скрипте загрузки в соответствии с этими примерами.

Региональные настройки по умолчанию в приложениях основаны на системных региональных настройках компьютера или сервера, где установлено ПО Qlik Sense. Если на сервере Qlik Sense, к которому обращается пользователь, выбран шведский язык, то редактор загрузки данных будет использовать шведские региональные настройки для даты, времени и валюты. Эти параметры регионального формата не связаны с языком, отображаемым в интерфейсе пользователя Qlik Sense. Qlik Sense будет отображаться на языке, который используется в браузере.

Пример 1. Без дополнительных аргументов

Скрипт загрузки и результаты

Обзор

Откройте редактор загрузки данных и добавьте приведенный ниже скрипт загрузки на новую вкладку.

Скрипт загрузки содержит следующее:

- Набор данных, содержащий набор транзакций за 2022 год, загружается в таблицу под именем `Transactions`.

- Поле даты было предоставлено в формате системной переменной `DateFormat` (MM/DD/YYYY).
- Создание поля `in_quarter_to_date`, которое определяет, какие транзакции совершены в квартале до 15 мая 2022 года.

Скрипт загрузки

```
SET DateFormat='MM/DD/YYYY';

Transactions:
    Load
        *,
        inquartertodate(date,'05/15/2022', 0) as in_quarter_to_date
        ;
Load
*
Inline
[
id,date,amount
8188,'1/19/2022',37.23
8189,'1/7/2022',17.17
8190,'2/28/2022',88.27
8191,'2/5/2022',57.42
8192,'3/16/2022',53.80
8193,'4/1/2022',82.06
8194,'5/7/2022',40.39
8195,'5/16/2022',87.21
8196,'6/15/2022',95.93
8197,'6/26/2022',45.89
8198,'7/9/2022',36.23
8199,'7/22/2022',25.66
8200,'7/23/2022',82.77
8201,'7/27/2022',69.98
8202,'8/2/2022',76.11
8203,'8/8/2022',25.12
8204,'8/19/2022',46.23
8205,'9/26/2022',84.21
8206,'10/14/2022',96.24
8207,'10/29/2022',67.67
];
```

Результаты

Загрузите данные и откройте лист. Создайте новую таблицу и добавьте эти поля как измерения:

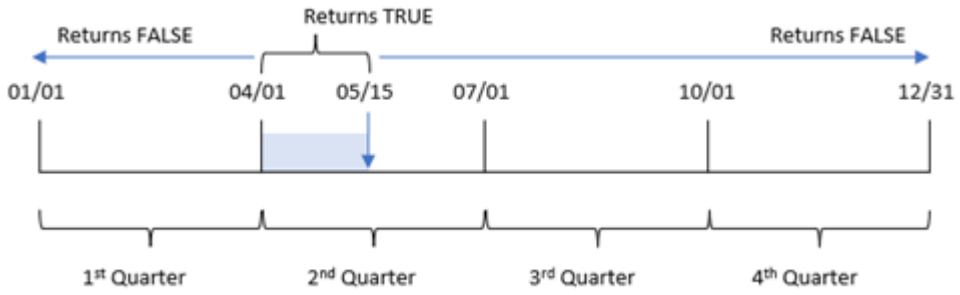
- `date`
- `in_quarter_to_date`

Результирующая таблица

date	in_quarter_to_date
1/7/2022	0
1/19/2022	0
2/5/2022	0
2/28/2022	0
3/16/2022	0
4/1/2022	-1
5/7/2022	-1
5/16/2022	0
6/15/2022	0
6/26/2022	0
7/9/2022	0
7/22/2022	0
7/23/2022	0
7/27/2022	0
8/2/2022	0
8/8/2022	0
8/19/2022	0
9/26/2022	0
10/14/2022	0
10/29/2022	0

Поле `in_quarter_to_date` создано в предыдущем операторе load с помощью функции `inquartertodate()`. Первый предоставленный аргумент определяет, какое поле оценивается. Второй аргумент – это жестко закодированная дата «15 мая», которая представляет собой `base_date`, определяющий, какой квартал сегментировать, и определяющий конечную границу этого сегмента. `period_no` = 0 – это последний аргумент, означающий, что функция не сравнивает кварталы, предшествующие или следующие за сегментированным кварталом.

Диаграмма функции inquartertodate без дополнительных аргументов



Любая транзакция, совершенная в период с 1 апреля по 15 мая, возвращает булев результат TRUE. Для транзакций с датами от 16 мая и позже возвращается FALSE, как и для транзакций до 1 апреля.

Пример 2. Скрипт period_no

Скрипт загрузки и результаты

Обзор

Откройте редактор загрузки данных и добавьте приведенный ниже скрипт загрузки на новую вкладку.

Скрипт загрузки содержит следующее:

- Используется тот же набор данных и сценарий, что в первом примере.
- Создание поля `previous_qtr_to_date`, которое определяет, какие транзакции совершены в течение полного квартала до 15 мая 2022 года.

Скрипт загрузки

```
SET DateFormat='MM/DD/YYYY';

Transactions:
    Load
        *,
        inquartertodate(date,'05/15/2022', -1) as previous_qtr_to_date
        ;
Load
*
Inline
[
id,date,amount
8188,'1/19/2022',37.23
8189,'1/7/2022',17.17
8190,'2/28/2022',88.27
8191,'2/5/2022',57.42
8192,'3/16/2022',53.80
8193,'4/1/2022',82.06
8194,'5/7/2022',40.39
```

```
8195,'5/16/2022',87.21
8196,'6/15/2022',95.93
8197,'6/26/2022',45.89
8198,'7/9/2022',36.23
8199,'7/22/2022',25.66
8200,'7/23/2022',82.77
8201,'7/27/2022',69.98
8202,'8/2/2022',76.11
8203,'8/8/2022',25.12
8204,'8/19/2022',46.23
8205,'9/26/2022',84.21
8206,'10/14/2022',96.24
8207,'10/29/2022',67.67
];
```

Результаты

Загрузите данные и откройте лист. Создайте новую таблицу и добавьте эти поля как измерения:

- `date`
- `previous_qtr_to_date`

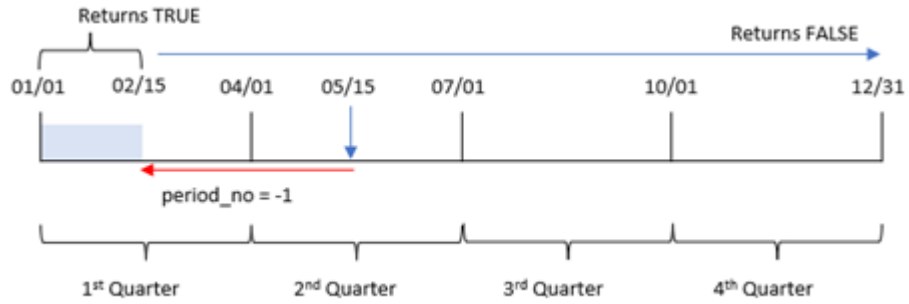
Результирующая таблица

date	previous_qtr_to_date
1/7/2022	-1
1/19/2022	-1
2/5/2022	-1
2/28/2022	0
3/16/2022	0
4/1/2022	0
5/7/2022	0
5/16/2022	0
6/15/2022	0
6/26/2022	0
7/9/2022	0
7/22/2022	0
7/23/2022	0
7/27/2022	0
8/2/2022	0
8/8/2022	0
8/19/2022	0

date	previous_qtr_to_date
9/26/2022	0
10/14/2022	0
10/29/2022	0

Значение `period_no` равное -1 указывает, что функция `inquartertodate ()` сравнивает сегмент входного квартала с предыдущим кварталом. 15 мая приходится на второй квартал года, поэтому этот сегмент изначально соответствует периоду с 1 апреля по 15 мая. Затем `period_no` смещает этот сегмент на три месяца назад, в результате чего границы диапазона дат переносятся на 1 января и 15 февраля.

Диаграмма функции inquartertodate, пример с аргументом period_no



Таким образом, любая транзакция, совершенная в период с 1 января по 15 февраля, возвращает булев результат `TRUE`.

Пример 3. Аргумент first_month_of_year

Скрипт загрузки и результаты

Обзор

Откройте редактор загрузки данных и добавьте приведенный ниже скрипт загрузки на новую вкладку.

Скрипт загрузки содержит следующее:

- Используется тот же набор данных и сценарий, что в первом примере.
- Создание поля `in_quarter_to_date`, которое определяет, какие транзакции совершены в том же квартале до 15 мая 2022 года.

В этом примере мы устанавливаем март в качестве первого месяца финансового года.

Скрипт загрузки

```
SET DateFormat='MM/DD/YYYY';
```

```
Transactions:
    Load
        *,
        inquartertodate(date,'05/15/2022', 0,3) as in_quarter_to_date
        ;
Load
*
Inline
[
id,date,amount
8188,'1/19/2022',37.23
8189,'1/7/2022',17.17
8190,'2/28/2022',88.27
8191,'2/5/2022',57.42
8192,'3/16/2022',53.80
8193,'4/1/2022',82.06
8194,'5/7/2022',40.39
8195,'5/16/2022',87.21
8196,'6/15/2022',95.93
8197,'6/26/2022',45.89
8198,'7/9/2022',36.23
8199,'7/22/2022',25.66
8200,'7/23/2022',82.77
8201,'7/27/2022',69.98
8202,'8/2/2022',76.11
8203,'8/8/2022',25.12
8204,'8/19/2022',46.23
8205,'9/26/2022',84.21
8206,'10/14/2022',96.24
8207,'10/29/2022',67.67
];
```

Результаты

Загрузите данные и откройте лист. Создайте новую таблицу и добавьте эти поля как измерения:

- `date`
- `in_quarter_to_date`

Результирующая таблица

date	in_quarter_to_date
1/7/2022	0
1/19/2022	0
2/5/2022	0
2/28/2022	0
3/16/2022	-1
4/1/2022	-1

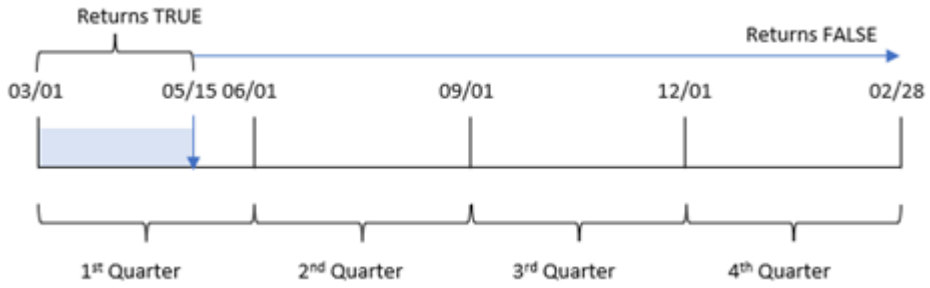
date	in_quarter_to_date
5/7/2022	-1
5/16/2022	0
6/15/2022	0
6/26/2022	0
7/9/2022	0
7/22/2022	0
7/23/2022	0
7/27/2022	0
8/2/2022	0
8/8/2022	0
8/19/2022	0
9/26/2022	0
10/14/2022	0
10/29/2022	0

Так как используется 3 в качестве аргумента [`first_month_of_year` в функции `inquartertodate()`, функция начинает год 1 марта, а затем делит год на кварталы. Следовательно, имеются следующие квартальные сегменты:

- с марта по май
- с июня по август
- с сентября по ноябрь
- с декабря по февраль

`base_date` от 15 мая затем сегментирует квартал с марта по май, устанавливая для него конечную границу 15 мая.

Диаграмма функции inquartertodate, пример с аргументом first_month_of_year



Таким образом, для любой транзакции, совершенной между 1 марта и 15 мая, возвращается булев результат TRUE, а для транзакций с датами вне этих границ – значение FALSE.

Пример 4. Пример объекта диаграммы

Скрипт загрузки и выражение диаграммы

Обзор

Откройте редактор загрузки данных и добавьте приведенный ниже скрипт загрузки на новую вкладку.

Скрипт загрузки содержит тот же набор данных и сценарий, что в первом примере. Однако в этом примере в приложение загружается неизмененный набор данных. Расчет, который определяет, совершены ли транзакции в том квартале, в который попадает 15 мая, создается как мера в объекте диаграммы в приложении.

Скрипт загрузки

```
SET DateFormat='MM/DD/YYYY';

Transactions:
Load
*
Inline
[
id,date,amount
8188,'1/19/2022',37.23
8189,'1/7/2022',17.17
8190,'2/28/2022',88.27
8191,'2/5/2022',57.42
8192,'3/16/2022',53.80
8193,'4/1/2022',82.06
8194,'5/7/2022',40.39
8195,'5/16/2022',87.21
8196,'6/15/2022',95.93
8197,'6/26/2022',45.89
8198,'7/9/2022',36.23
8199,'7/22/2022',25.66
8200,'7/23/2022',82.77
8201,'7/27/2022',69.98
8202,'8/2/2022',76.11
8203,'8/8/2022',25.12
8204,'8/19/2022',46.23
8205,'9/26/2022',84.21
8206,'10/14/2022',96.24
8207,'10/29/2022',67.67
];
```

Результаты

Загрузите данные и откройте лист. Создайте новую таблицу и добавьте это поле как измерение:`date`.

Создайте следующую меру:

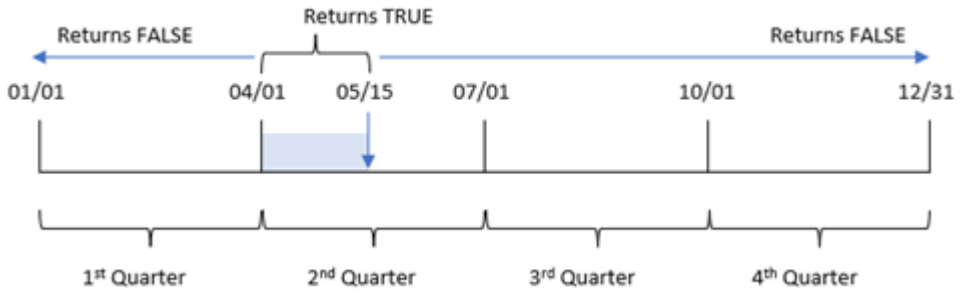
```
=inquartertodate(date,'05/15/2022', 0)
```

Результирующая таблица

date	=inquartertodate(date,'05/15/2022', 0)
1/7/2022	0
1/19/2022	0
2/5/2022	0
2/28/2022	0
3/16/2022	0
4/1/2022	-1
5/7/2022	-1
5/16/2022	0
6/15/2022	0
6/26/2022	0
7/9/2022	0
7/22/2022	0
7/23/2022	0
7/27/2022	0
8/2/2022	0
8/8/2022	0
8/19/2022	0
9/26/2022	0
10/14/2022	0
10/29/2022	0

Мера `in_quarter_to_date` создается в объекте диаграммы с помощью функции `inquartertodate()`. Первый аргумент – это оцениваемое поле даты. Второй аргумент – это жестко закодированная дата «15 мая», которая представляет собой `base_date`, определяющий, какой квартал сегментировать, и определяющий конечную границу этого сегмента. `period_no` = 0 – это последний аргумент, означающий, что функция не сравнивает кварталы, предшествующие или следующие за сегментированным кварталом.

Диаграмма функции inquartertodate, пример с объектом диаграммы



Любая транзакция, совершенная в период с 1 апреля по 15 мая, возвращает булев результат `TRUE`. Для транзакций с датами от 16 мая и позже возвращается `FALSE`, как и для транзакций до 1 апреля.

Пример 5. Сценарий

Скрипт загрузки и выражение диаграммы

Обзор

Откройте редактор загрузки данных и добавьте приведенный ниже скрипт загрузки на новую вкладку.

Скрипт загрузки содержит следующее:

- Набор данных, загруженный в таблицу, называется `Products`
- Информация об идентификаторе продукта, дате изготовления и себестоимости.

15 мая 2022 г. в производственном процессе была обнаружена и устранена ошибка оборудования. Продукты, изготовленные в этом квартале до этой даты, будут дефектными. Конечному пользователю требуется объект диаграммы, который отображает по названию квартала, какие изготовленные продукты были дефектными, а какие бездефектными, а также стоимость продуктов, изготовленных в этом квартале до указанной даты.

Скрипт загрузки

```
Products:
Load
*
Inline
[
product_id,manufacture_date,cost_price
8188,'1/19/2022',37.23
8189,'1/7/2022',17.17
8190,'2/28/2022',88.27
8191,'2/5/2022',57.42
8192,'3/16/2022',53.80
8193,'4/1/2022',82.06
8194,'5/7/2022',40.39
8195,'5/16/2022',87.21
```

```
8196,'6/15/2022',95.93
8197,'6/26/2022',45.89
8198,'7/9/2022',36.23
8199,'7/22/2022',25.66
8200,'7/23/2022',82.77
8201,'7/27/2022',69.98
8202,'8/2/2022',76.11
8203,'8/8/2022',25.12
8204,'8/19/2022',46.23
8205,'9/26/2022',84.21
8206,'10/14/2022',96.24
8207,'10/29/2022',67.67
];
```

Результаты

Выполните следующие действия.

1. Загрузите данные и откройте лист. Создайте новую таблицу. Создайте измерение для отображения названий кварталов:
   `=quartername(manufacture_date)`
2. Затем создайте измерение, чтобы определить, какие из продуктов дефектные, а какие нет:
   `=if(inquartertodate(manufacture_date,makedate(2022,05,15),0),'Defective','Faultless')`
3. Создайте меру для суммирования `cost_price` продуктов:
   `=sum(cost_price)`
4. Задайте параметру меры **Формат чисел** значение **Денежный**.

Результирующая таблица

quartername (manufacture_date)	if(inquartertodate(manufacture_date,makedate (2022,05,15),0),'Defective','Faultless')	Sum(cost_ price)
Jan-Mar 2022	Бездефектный	$253.89
Apr-Jun 2022	Бездефектный	$229.03
Apr-Jun 2022	Дефектный	$122.45
Jul-Sep 2022	Бездефектный	$446.31
Oct-Dec 2022	Бездефектный	$163.91

Функция `inquartertodate()` возвращает логическое значение при проверки дат производства каждого продукта. Продукты, для которых возвращается булево значение `TRUE`, помечаются как `'Defective'`. Продукты, для которых возвращается значение `FALSE` и которые, следовательно, не произведены в квартале до 15 мая включительно, помечаются как `'Faultless'`.

inweek

Эта функция возвращает значение True, если поле **timestamp** находится в пределах недели, включающей значение, указанное в поле **base_date**.

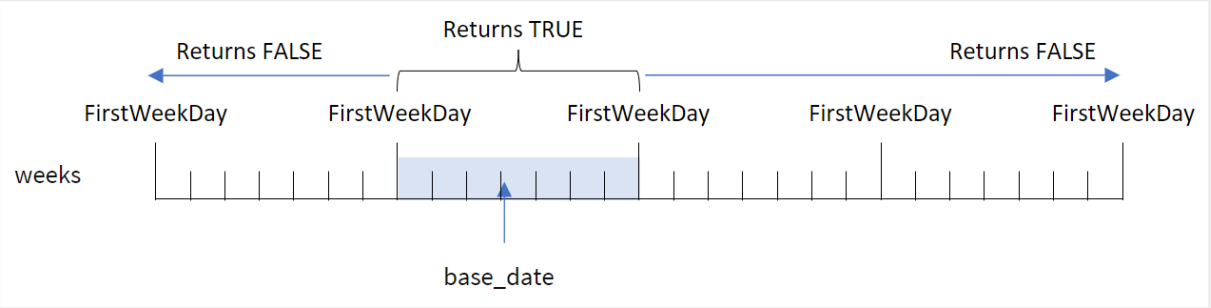
Синтаксис:

```
InWeek (timestamp, base_date, period_no[, first_week_day])
```

Возвращаемые типы данных: Булево значение

В Qlik Sense логическое значение «истина» представлено как -1, а «ложь» – как 0.

Диаграмма диапазона функции `inweek()`



Функция `inweek()` использует аргумент `base_date` для определения того, в какой семидневный период попадает дата. Начало недели определяется системной переменной `FirstWeekDay`. Однако также можно изменить первый день недели, используя аргумент `first_week_day` в функции `inweek()`.

После выбора недели функция возвращает булевы результаты при сравнении предписанных значений даты с этим сегментом.

Когда это следует использовать

Функция `inweek()` возвращает результат в виде булева значения. Обычно этот тип функции используется в качестве условия в `if expression`. Функция `inweek()` возвращает агрегирование или вычисление в зависимости от того, попадает ли проверяемая дата в неделю, к которой относится дата, переданная в аргументе `base_date`.

Например, функцию `inweek()` можно использовать для идентификации всего оборудования, изготовленного в течение указанной недели.

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 воскресенье (согласно системной переменной FirstWeekDay), начиная с полуночи между субботой и воскресеньем. Параметр **first_week_day** заменяет переменную **FirstWeekDay**. Чтобы задать другой день в качестве начала недели, укажите флаг от 0 до 6.

значения first_week_day

День	Значение
Понедельник	0
Вторник	1
Среда	2
Четверг	3
Пятница	4
Суббота	5
Воскресенье	6

Региональные настройки

Если не указано иное, в примерах, приведенных в данном разделе, используется следующий формат даты: ММ/ДД/ГГГГ. Формат даты указан в операторе `SET DateFormat` скрипта загрузки данных. В вашей системе может быть установлен другой формат даты по умолчанию в зависимости от региональных настроек и других факторов. Можно изменить формат в примерах в соответствии с потребностями. Или можно изменить форматы в скрипте загрузки в соответствии с этими примерами.

Региональные настройки по умолчанию в приложениях основаны на системных региональных настройках компьютера или сервера, где установлено ПО Qlik Sense. Если на сервере Qlik Sense, к которому обращается пользователь, выбран шведский язык, то редактор загрузки данных будет использовать шведские региональные настройки для даты, времени и валюты. Эти параметры регионального формата не связаны с языком, отображаемым в интерфейсе пользователя Qlik Sense. Qlik Sense будет отображаться на языке, который используется в браузере.

Примеры функции

Пример	Результат
`inweek ('01/12/2006', '01/14/2006', 0)`	Возвращает TRUE
`inweek ('01/12/2006', '01/20/2006', 0 )`	Возвращает FALSE
`inweek ('01/12/2006', '01/14/2006', -1 )`	Возвращает FALSE

Пример	Результат
`inweek ('01/07/2006', '01/14/2006', -1)`	Возвращает TRUE
`inweek ('01/12/2006', '01/09/2006', 0, 3)`	Возвращает FALSE, поскольку для аргумента `first_week_day` указано значение 3 (четверг), в результате чего 01/12/2006 становится первым днем недели после недели, содержащей дату 01/09/2006.

Эти темы помогут вам в работе с этой функцией:

Связанные темы

Тема	Флаг по умолчанию / значение	Описание
FirstWeekDay (page 225)	6 / воскресенье	Определяет день начала каждой недели.

Пример 1. Без дополнительных аргументов

Скрипт загрузки и результаты

Обзор

Откройте редактор загрузки данных и добавьте приведенный ниже скрипт загрузки на новую вкладку.

Скрипт загрузки содержит следующее:

- Набор данных, содержащий набор транзакций за январь 2022 года, загруженный в таблицу под именем `Transactions`.
- Системная переменная `FirstWeekDay`, для которой задано значение `6` (воскресенье).
- Предыдущая загрузка, которая содержит следующее:
  - Функция `inweek()`, которая задана как поле `in_week` и определяет, какие транзакции совершены в течение недели, к которой относится 14 января 2022 года.
  - Функция `weekday()`, которая задана как поле `week_day` и показывает, какой день недели соответствует каждой дате.

Скрипт загрузки

```
SET FirstWeekDay=6;
SET DateFormat='MM/DD/YYYY';

Transactions:
    Load
       *,
       weekday(date) as week_day,
```

```
       inweek(date,'01/14/2022', 0) as in_week
       ;
Load
*
Inline
[
id,date,amount
8188,'01/02/2022',37.23
8189,'01/05/2022',17.17
8190,'01/06/2022',88.27
8191,'01/08/2022',57.42
8192,'01/09/2022',53.80
8193,'01/10/2022',82.06
8194,'01/11/2022',40.39
8195,'01/12/2022',87.21
8196,'01/13/2022',95.93
8197,'01/14/2022',45.89
8198,'01/15/2022',36.23
8199,'01/16/2022',25.66
8200,'01/17/2022',82.77
8201,'01/18/2022',69.98
8202,'01/26/2022',76.11
8203,'01/27/2022',25.12
8204,'01/28/2022',46.23
8205,'01/29/2022',84.21
8206,'01/30/2022',96.24
8207,'01/31/2022',67.67
];
```

Результаты

Загрузите данные и откройте лист. Создайте новую таблицу и добавьте эти поля как измерения:

- date
- week_day
- in_week

Результирующая таблица

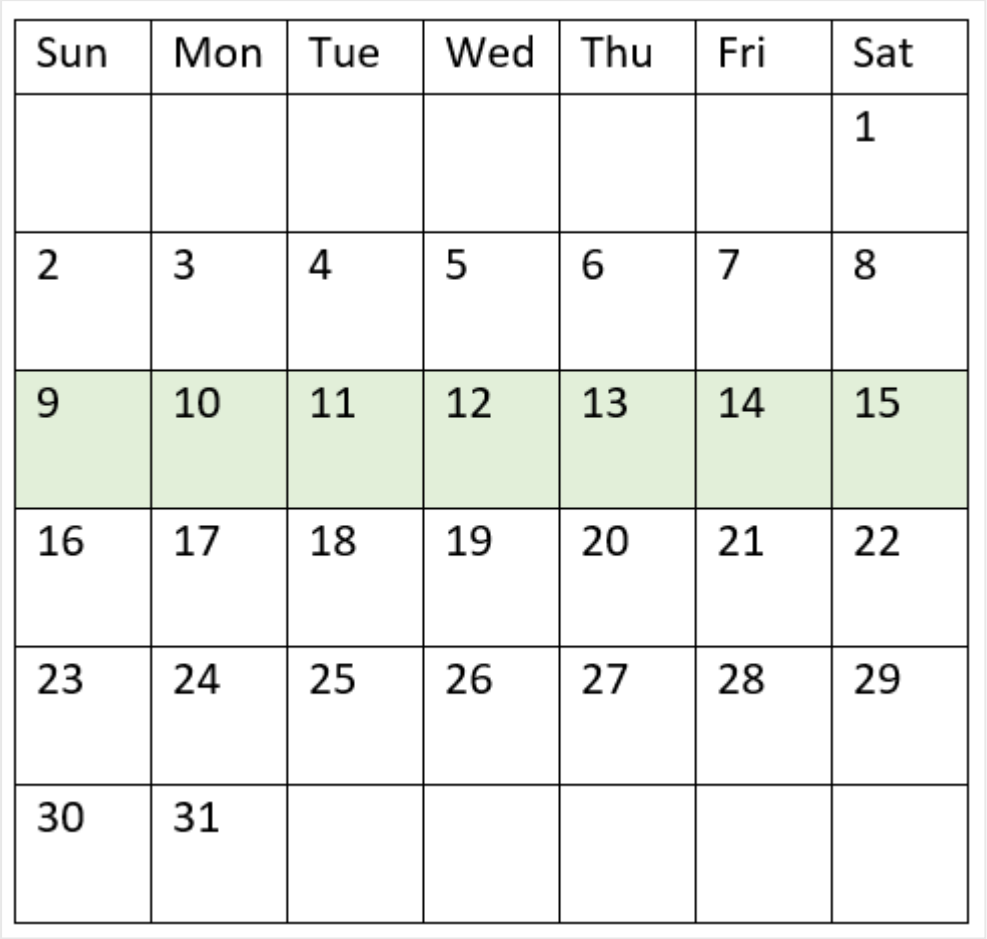
date	week_day	in_week
01/02/2022	Sun	0
01/05/2022	Wed	0
01/06/2022	Thu	0
01/08/2022	Sat	0
01/09/2022	Sun	-1
01/10/2022	Mon	-1
01/11/2022	Tue	-1

date	week_day	in_week
01/12/2022	Wed	-1
01/13/2022	Thu	-1
01/14/2022	Fri	-1
01/15/2022	Sat	-1
01/16/2022	Sun	0
01/17/2022	Mon	0
01/18/2022	Tue	0
01/26/2022	Wed	0
01/27/2022	Thu	0
01/28/2022	Fri	0
01/29/2022	Sat	0
01/30/2022	Sun	0
01/31/2022	Mon	0

Поле `in_week` создано в предыдущем операторе load с помощью функции `inweek()`. Первый аргумент определяет, какое поле оценивается. Второй аргумент — это жестко закодированная дата 14 января, которая выступает в роли `base_date`. Аргумент `base_date` работает с системной переменной `FirstWeekDay` для идентификации сравниваемой недели. `period_no = 0` означает, что функция не сравнивает недели, следующие до или после сегментированной недели; это последний аргумент.

Системная переменная `FirstWeekDay` определяет, что недели начинаются в воскресенье и заканчиваются в субботу. Таким образом, январь будет разбит на недели в соответствии с приведенной ниже диаграммой, где даты между 9 и 15 января будут действительным периодом для расчета `inweek()`:

Диаграмма календаря с выделенным диапазоном функции `inweek()`



Sun	Mon	Tue	Wed	Thu	Fri	Sat
						1
2	3	4	5	6	7	8
9	10	11	12	13	14	15
16	17	18	19	20	21	22
23	24	25	26	27	28	29
30	31					

Любая транзакция, совершенная в период с 9 по 15 января, возвращает булев результат `TRUE`.

Пример 2. Аргумент period_no

Скрипт загрузки и результаты

Обзор

Откройте редактор загрузки данных и добавьте приведенный ниже скрипт загрузки на новую вкладку.

Скрипт загрузки содержит следующее:

- Тот же набор данных, содержащий набор транзакций за 2022 год, загруженный в таблицу под именем `Transactions`.
- Системная переменная `FirstWeekDay`, для которой задано значение `6` (воскресенье).
- Предыдущая загрузка, которая содержит следующее:

- Функция `inweek ()`, которая задана как поле `prev_week` и определяет, какие транзакции совершены за неделю до недели, к которой относится 14 января 2022 года.
- Функция `weekday()`, которая задана как поле `week_day`, и показывает, какой день недели соответствует каждой дате.

Скрипт загрузки

```
SET FirstWeekDay=6;
SET DateFormat='MM/DD/YYYY';

Transactions:
    Load
       *,
       weekday(date) as week_day,
       inweek(date,'01/14/2022', -1) as prev_week
       ;
Load
*
Inline
[
id,date,amount
8188,'01/02/2022',37.23
8189,'01/05/2022',17.17
8190,'01/06/2022',88.27
8191,'01/08/2022',57.42
8192,'01/09/2022',53.80
8193,'01/10/2022',82.06
8194,'01/11/2022',40.39
8195,'01/12/2022',87.21
8196,'01/13/2022',95.93
8197,'01/14/2022',45.89
8198,'01/15/2022',36.23
8199,'01/16/2022',25.66
8200,'01/17/2022',82.77
8201,'01/18/2022',69.98
8202,'01/26/2022',76.11
8203,'01/27/2022',25.12
8204,'01/28/2022',46.23
8205,'01/29/2022',84.21
8206,'01/30/2022',96.24
8207,'01/31/2022',67.67
];
```

Результаты

Загрузите данные и откройте лист. Создайте новую таблицу и добавьте эти поля как измерения:

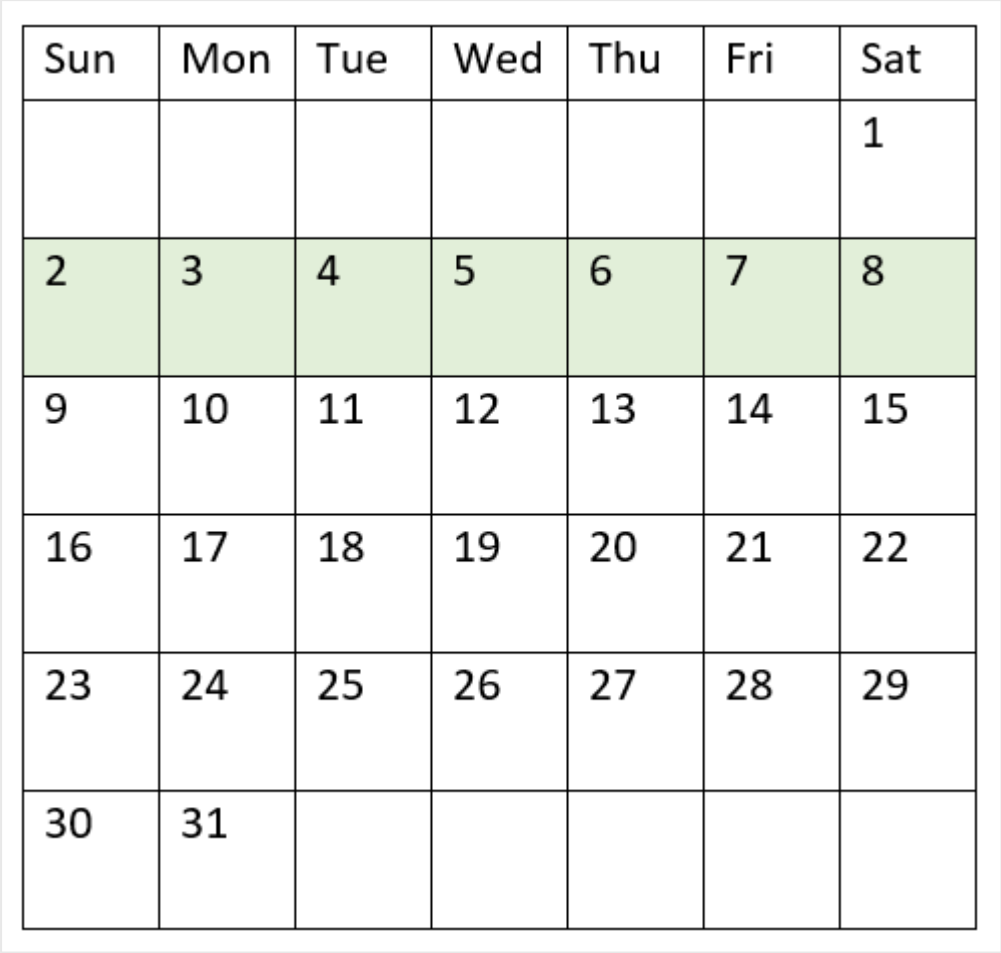
- `date`
- `week_day`
- `prev_week`

Результирующая таблица

date	week_day	prev_week
01/02/2022	Sun	-1
01/05/2022	Wed	-1
01/06/2022	Thu	-1
01/08/2022	Sat	-1
01/09/2022	Sun	0
01/10/2022	Mon	0
01/11/2022	Tue	0
01/12/2022	Wed	0
01/13/2022	Thu	0
01/14/2022	Fri	0
01/15/2022	Sat	0
01/16/2022	Sun	0
01/17/2022	Mon	0
01/18/2022	Tue	0
01/26/2022	Wed	0
01/27/2022	Thu	0
01/28/2022	Fri	0
01/29/2022	Sat	0
01/30/2022	Sun	0
01/31/2022	Mon	0

Используя `-1` в качестве аргумента `period_no`, функция `inweek()` сдвигает границы сравниваемой недели на семь дней назад. При `period_no = 0` границами недели являются 9 и 15 января. Но в этом примере `period_no = -1` сдвигает начальную и конечную границы этого сегмента на одну неделю назад. Границы диапазона дат переносятся на 2 и 8 января.

Диаграмма календаря с выделенным диапазоном функции `inweek()`



Sun	Mon	Tue	Wed	Thu	Fri	Sat
						1
2	3	4	5	6	7	8
9	10	11	12	13	14	15
16	17	18	19	20	21	22
23	24	25	26	27	28	29
30	31					

Таким образом, любая транзакция, совершенная в период с 2 по 8 января, возвращает булев результат TRUE.

Пример 3. Аргумент first_week_day

Скрипт загрузки и результаты

Обзор

Откройте редактор загрузки данных и добавьте приведенный ниже скрипт загрузки на новую вкладку.

Скрипт загрузки содержит следующее:

- Тот же набор данных, содержащий набор транзакций за 2022 год, загруженный в таблицу под именем `Transactions`.
- Системная переменная `FirstWeekDay`, для которой задано значение `6` (воскресенье).
- Предыдущая загрузка, которая содержит следующее:

- Функция `inweek()`, которая задана как поле `in_week` и определяет, какие транзакции совершены в течение недели, к которой относится 14 января 2022 года.
- Функция `weekday()`, которая задана как поле `week_day` и показывает, какой день недели соответствует каждой дате.

Скрипт загрузки

```
SET FirstWeekDay=6;
SET DateFormat='MM/DD/YYYY';

Transactions:
    Load
       *,
       weekday(date) as week_day,
       inweek(date,'01/14/2022', 0, 0) as in_week
       ;
Load
*
Inline
[
id,date,amount
8188,'01/02/2022',37.23
8189,'01/05/2022',17.17
8190,'01/06/2022',88.27
8191,'01/08/2022',57.42
8192,'01/09/2022',53.80
8193,'01/10/2022',82.06
8194,'01/11/2022',40.39
8195,'01/12/2022',87.21
8196,'01/13/2022',95.93
8197,'01/14/2022',45.89
8198,'01/15/2022',36.23
8199,'01/16/2022',25.66
8200,'01/17/2022',82.77
8201,'01/18/2022',69.98
8202,'01/26/2022',76.11
8203,'01/27/2022',25.12
8204,'01/28/2022',46.23
8205,'01/29/2022',84.21
8206,'01/30/2022',96.24
8207,'01/31/2022',67.67
];
```

Результаты

Загрузите данные и откройте лист. Создайте новую таблицу и добавьте эти поля как измерения:

- `date`
- `week_day`
- `in_week`

Результирующая таблица

date	week_day	in_week
01/02/2022	Sun	0
01/05/2022	Wed	0
01/06/2022	Thu	0
01/08/2022	Sat	0
01/09/2022	Sun	0
01/10/2022	Mon	-1
01/11/2022	Tue	-1
01/12/2022	Wed	-1
01/13/2022	Thu	-1
01/14/2022	Fri	-1
01/15/2022	Sat	-1
01/16/2022	Sun	-1
01/17/2022	Mon	0
01/18/2022	Tue	0
01/26/2022	Wed	0
01/27/2022	Thu	0
01/28/2022	Fri	0
01/29/2022	Sat	0
01/30/2022	Sun	0
01/31/2022	Mon	0

Использование `0` в качестве аргумента в функции `first_week_day` аргумент функции `inweek()` заменяет системную переменную `FirstWeekDay` и устанавливает понедельник в качестве первого дня недели.

Диаграмма календаря с выделенным диапазоном функции `inweek()`

Sun	Mon	Tue	Wed	Thu	Fri	Sat
						1
2	3	4	5	6	7	8
9	10	11	12	13	14	15
16	17	18	19	20	21	22
23	24	25	26	27	28	29
30	31					

Таким образом, любая транзакция, совершенная в период с 10 по 16 января, возвращает булев результат TRUE.

Пример 4. Пример объекта диаграммы

Скрипт загрузки и выражение диаграммы

Обзор

Используется тот же набор данных и сценарий, что в первом примере.

Однако в этом примере в приложение загружается неизмененный набор данных. Создайте меру в таблице результатов, чтобы определить, какие транзакции совершены в течение недели, к которой относится 14 января 2022 года.

Скрипт загрузки

```
SET FirstWeekDay=6;
SET DateFormat='MM/DD/YYYY';

Transactions:
```

```
Load
*
Inline
[
id,date,amount
8188,'01/02/2022',37.23
8189,'01/05/2022',17.17
8190,'01/06/2022',88.27
8191,'01/08/2022',57.42
8192,'01/09/2022',53.80
8193,'01/10/2022',82.06
8194,'01/11/2022',40.39
8195,'01/12/2022',87.21
8196,'01/13/2022',95.93
8197,'01/14/2022',45.89
8198,'01/15/2022',36.23
8199,'01/16/2022',25.66
8200,'01/17/2022',82.77
8201,'01/18/2022',69.98
8202,'01/26/2022',76.11
8203,'01/27/2022',25.12
8204,'01/28/2022',46.23
8205,'01/29/2022',84.21
8206,'01/30/2022',96.24
8207,'01/31/2022',67.67
];
```

Результаты

Загрузите данные и откройте лист. Создайте новую таблицу и добавьте это поле как измерение:

- `date`

Создайте следующие меры:

- `=inweek (date,'01/14/2022',0)`, чтобы вычислить, какие транзакции совершены в течение недели, к которой относится 14 января.
- `=weekday(date)`, чтобы показать, какой день недели соответствует каждой дате.

Результирующая таблица

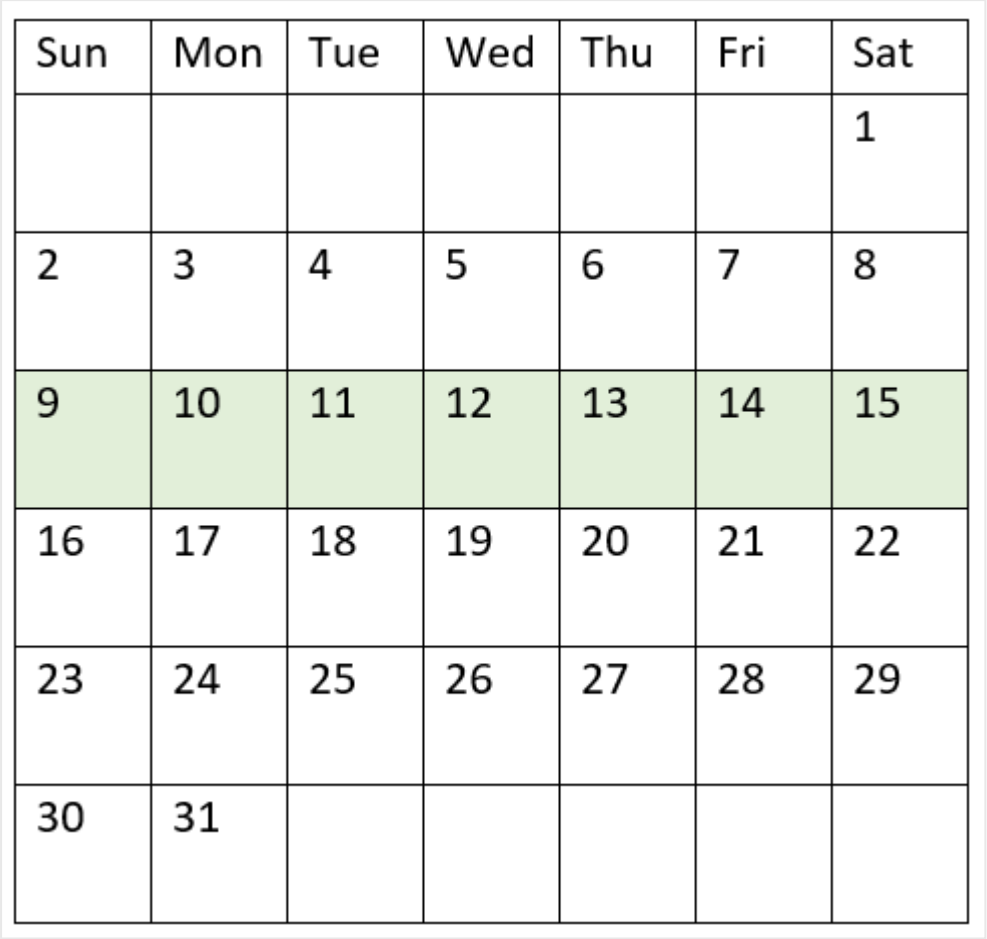
date	week_day	=inweek (date,'01/14/2022',0)
01/02/2022	Sun	0
01/05/2022	Wed	0
01/06/2022	Thu	0
01/08/2022	Sat	0
01/09/2022	Sun	-1
01/10/2022	Mon	-1

date	week_day	=inweek (date,'01/14/2022',0)
01/11/2022	Tue	-1
01/12/2022	Wed	-1
01/13/2022	Thu	-1
01/14/2022	Fri	-1
01/15/2022	Sat	-1
01/16/2022	Sun	0
01/17/2022	Mon	0
01/18/2022	Tue	0
01/26/2022	Wed	0
01/27/2022	Thu	0
01/28/2022	Fri	0
01/29/2022	Sat	0
01/30/2022	Sun	0
01/31/2022	Mon	0

Мера `in_week` создается в диаграмме с помощью функции `inweek()`. Первый аргумент определяет, какое поле оценивается. Второй аргумент – это жестко закодированная дата 14 января, которая выступает в роли `base_date`. Аргумент `base_date` работает с системной переменной `FirstWeekDay` для идентификации сравниваемой недели. Заключительный аргумент – `period_no = 0`.

Системная переменная `FirstWeekDay` определяет, что недели начинаются в воскресенье и заканчиваются в субботу. Таким образом, январь будет разбит на недели в соответствии с приведенной ниже диаграммой, где даты между 9 и 15 января будут действительным периодом для расчета `inweek()`:

Диаграмма календаря с выделенным диапазоном функции `inweek()`



Sun	Mon	Tue	Wed	Thu	Fri	Sat
						1
2	3	4	5	6	7	8
9	10	11	12	13	14	15
16	17	18	19	20	21	22
23	24	25	26	27	28	29
30	31					

Любая транзакция, совершенная в период с 9 по 15 января, возвращает булев результат `TRUE`.

Пример 5. Сценарий

Скрипт загрузки и результаты

Обзор

Откройте редактор загрузки данных и добавьте приведенный ниже скрипт загрузки на новую вкладку.

Скрипт загрузки содержит следующее:

- Набор данных, загруженный в таблицу под именем `Products`.
- Данная таблица содержит следующие поля:
  - product ID (ИД продукта)
  - product type (тип продукта)
  - manufacture date (дата изготовления)
  - cost price (себестоимость)

Установлено, что вследствие ошибки оборудования продукты, изготовленные в течение недели, на которую выпадает 12 января, являются дефектными. Конечному пользователю требуется диаграмма, которая отображает по неделе, какие изготовленные продукты были дефектными, а какие бездефектными, а также стоимость продуктов, изготовленных на этой неделе.

Скрипт загрузки

```
Products:
Load
*
Inline
[
product_id,manufacture_date,cost_price
8188,'01/02/2022',37.23
8189,'01/05/2022',17.17
8190,'01/06/2022',88.27
8191,'01/08/2022',57.42
8192,'01/09/2022',53.80
8193,'01/10/2022',82.06
8194,'01/11/2022',40.39
8195,'01/12/2022',87.21
8196,'01/13/2022',95.93
8197,'01/14/2022',45.89
8198,'01/15/2022',36.23
8199,'01/16/2022',25.66
8200,'01/17/2022',82.77
8201,'01/18/2022',69.98
8202,'01/26/2022',76.11
8203,'01/27/2022',25.12
8204,'01/28/2022',46.23
8205,'01/29/2022',84.21
8206,'01/30/2022',96.24
8207,'01/31/2022',67.67
];
```

Результаты

Загрузите данные и откройте лист. Создайте новую таблицу и добавьте это поле как измерение:

- `=weekname(manufacture_date)`

Создайте следующие меры:

- `=if(only(inweek(manufacture_date,makedate(2022,01,12),0)),'Defective','Faultless')`, чтобы определить, какие из продуктов дефектные, а какие нет, с использованием функции `inweek()`:
- `=sum(cost_price)`, чтобы показать суммарную стоимость всех продуктов.

Выполните следующие действия.

1. Задайте параметру **Формат чисел** меры значение **Денежный**.
2. В области **Вид** выключите параметр **Итоги**.

Результирующая таблица

weekname (manufacture_date)	=if(only(inweek(manufacture_date,makedate (2022,01,12),0)), 'Defective','Faultless')	Sum(cost_ price)
2022/02	Бездефектный	200.09
2022/03	Дефектный	441.51
2022/04	Бездефектный	178.41
2022/05	Бездефектный	231.67
2022/06	Бездефектный	163.91

Функция `inweek()` возвращает логическое значение при проверке дат производства каждого продукта. Для любого продукта, изготовленного 12 января, функция `inweek()` возвращает булево значение TRUE и ставит отметку «Дефектный». Любой продукт, который возвращает значение FALSE, то есть изготовлен в другую неделю, отмечается как «Бездефектный».

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.

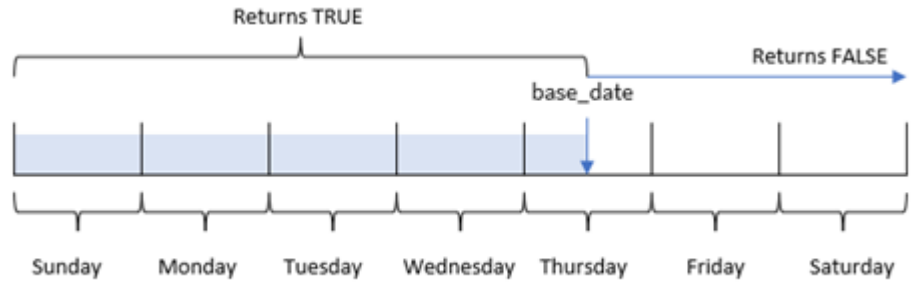
Синтаксис:

```
InWeekToDate (timestamp, base_date, period_no [, first_week_day])
```

Возвращаемые типы данных: Булево значение

> *В Qlik Sense логическое значение «истина» представлено как -1, а «ложь» – как 0.*

Диаграмма функции inweektodate



Функция inweektodate() использует параметр base_date для определения дату максимальной границы сегмента недели, а также соответствующей даты начала недели, которая основана на системной переменной FirstWeekDay (или определяемом пользователем параметре first_week_day). После определения сегмента недели функция возвращает булевы результаты при сравнении предписанных значений даты с этим сегментом.

Когда это следует использовать

Функция inweektodate() возвращает результат в виде логического значения. Обычно этот тип функции используется в качестве условия в выражении if. Функция возвращает агрегирование или расчет в зависимости от того, попадает ли проверяемая дата в рассматриваемую неделю до указанной даты включительно.

Например, функцию inweektodate() можно использовать для расчета всех продаж, совершенных в течение указанной недели до определенной даты.

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 воскресенье (согласно системной переменной FirstWeekDay), начиная с полуночи между субботой и воскресеньем. Параметр **first_week_day** заменяет переменную **FirstWeekDay**. Чтобы задать другой день в качестве начала недели, укажите флаг от 0 до 6. Чтобы неделя начиналась в понедельник и заканчивалась в воскресенье, используйте флаг 0 для понедельника, 1 для вторника, 2 для среды, 3 для четверга, 4 для пятницы, 5 для субботы и 6 для воскресенья.

Примеры функции

Пример	Взаимодействие
`inweektodate ('01/12/2006', '01/12/2006', 0)`	Возвращает TRUE.
`inweektodate ('01/12/2006', '01/11/2006', 0)`	Возвращает FALSE.

Пример	Взаимодействие
`inweektodate ('01/12/2006', '01/18/2006', -1)`	Возвращает `FALSE`. Поскольку для элемента `period_no` указано значение -1, дата вступления в силу `timestamp` измеряется на основе 01/11/2006.
`inweektodate ('01/11/2006', '01/12/2006', 0, 3 )`	Возвращает `FALSE`, поскольку для элемента `first_week_day` указано значение `3` (четверг), в результате чего элемент 01/12/2006 становится первым днем недели после недели с элементом 01/12/2006.

Эти темы помогут вам в работе с этой функцией:

Связанные темы

Тема	Флаг по умолчанию / значение	Описание
FirstWeekDay (page 225)	6 / воскресенье	Определяет день начала каждой недели.

Региональные настройки

Если не указано иное, в примерах, приведенных в данном разделе, используется следующий формат даты: ММ/ДД/ГГГГ. Формат даты указан в операторе `SET DateFormat` скрипта загрузки данных. В вашей системе может быть установлен другой формат даты по умолчанию в зависимости от региональных настроек и других факторов. Можно изменить формат в примерах в соответствии с потребностями. Или можно изменить форматы в скрипте загрузки в соответствии с этими примерами.

Региональные настройки по умолчанию в приложениях основаны на системных региональных настройках компьютера или сервера, где установлено ПО Qlik Sense. Если на сервере Qlik Sense, к которому обращается пользователь, выбран шведский язык, то редактор загрузки данных будет использовать шведские региональные настройки для даты, времени и валюты. Эти параметры регионального формата не связаны с языком, отображаемым в интерфейсе пользователя Qlik Sense. Qlik Sense будет отображаться на языке, который используется в браузере.

Пример 1. Без дополнительных аргументов

Скрипт загрузки и результаты

Обзор

Откройте редактор загрузки данных и добавьте приведенный ниже скрипт загрузки на новую вкладку.

Скрипт загрузки содержит следующее:

- Набор данных, содержащий набор транзакций за январь 2022 года, загружается в таблицу под именем `Transactions`.

- Поле данных представлено в формате `TimestampFormat='M/D/YYYY h:mm:ss[.fff]'`.
- Создание поля `in_week_to_date`, которое определяет, какие транзакции совершены в течение недели до 14 января 2022 года.
- Создание дополнительного поля под именем `weekday` с использованием функции `weekday()`. Это новое поле создано, чтобы показать, какой день недели соответствует каждой дате.

Скрипт загрузки

```
SET TimestampFormat='M/D/YYYY h:mm:ss[.fff]';
SET FirstWeekDay=6;
Transactions:
    Load
        *,
        weekday(date) as week_day,
        inweektodate(date,'01/14/2022', 0) as in_week_to_date
        ;
Load
*
Inline
[
id,date,amount
8188,'2022-01-02 12:22:06',37.23
8189,'2022-01-05 01:02:30',17.17
8190,'2022-01-06 15:36:20',88.27
8191,'2022-01-08 10:58:35',57.42
8192,'2022-01-09 08:53:32',53.80
8193,'2022-01-10 21:13:01',82.06
8194,'2022-01-11 00:57:13',40.39
8195,'2022-01-12 09:26:02',87.21
8196,'2022-01-13 15:05:09',95.93
8197,'2022-01-14 18:44:57',45.89
8198,'2022-01-15 06:10:46',36.23
8199,'2022-01-16 06:39:27',25.66
8200,'2022-01-17 10:44:16',82.77
8201,'2022-01-18 18:48:17',69.98
8202,'2022-01-26 04:36:03',76.11
8203,'2022-01-27 08:07:49',25.12
8204,'2022-01-28 12:24:29',46.23
8205,'2022-01-30 11:56:56',84.21
8206,'2022-01-30 14:40:19',96.24
8207,'2022-01-31 05:28:21',67.67
];
```

Результаты

Загрузите данные и откройте лист. Создайте новую таблицу и добавьте эти поля как измерения:

- `date`
- `week_day`
- `in_week_to_date`

Результирующая таблица

date	week_day	in_week_to_date
2022-01-02 12:22:06	Sun	0
2022-01-05 01:02:30	Wed	0
2022-01-06 15:36:20	Thu	0
2022-01-08 10:58:35	Sat	0
2022-01-09 08:53:32	Sun	-1
2022-01-10 21:13:01	Mon	-1
2022-01-11 00:57:13	Tue	-1
2022-01-12 09:26:02	Wed	-1
2022-01-13 15:05:09	Thu	-1
2022-01-14 18:44:57	Fri	-1
2022-01-15 06:10:46	Sat	0
2022-01-16 06:39:27	Sun	0
2022-01-17 10:44:16	Mon	0
2022-01-18 18:48:17	Tue	0
2022-01-26 04:36:03	Wed	0
2022-01-27 08:07:49	Thu	0
2022-01-28 12:24:29	Fri	0
2022-01-30 11:56:56	Sun	0
2022-01-30 14:40:19	Sun	0
2022-01-31 05:28:21	Mon	0

Поле `in_week_to_date` создано в предыдущем операторе load с помощью функции `inweektodate()`. Первый предоставленный аргумент определяет, какое поле оценивается. Второй аргумент — это жестко закодированная дата «14 мая», которая представляет собой `base_date`, определяющий, какую неделю сегментировать, и определяющий конечную границу этого сегмента. `period_no` = 0 — это последний аргумент, означающий, что функция не сравнивает недели, предшествующие или следующие за сегментированной недели.

Системная переменная `FirstWeekDay` определяет, что недели начинаются в воскресенье и заканчиваются в субботу. Таким образом, январь будет разбит на недели в соответствии с приведенной ниже диаграммой, где даты между 9 и 14 января будут действительным периодом для расчета `inweekdodate()` :

Диаграмма календаря с датами транзакций, которые вернут булев результат TRUE

Sun	Mon	Tue	Wed	Thur	Fri	Sat
						1
2	3	4	5	6	7	8
9	10	11	12	13	14	15
16	17	18	19	20	21	22
23	24	25	26	27	28	29
30	31					

Любая транзакция, совершенная в период с 9 апреля по 14 января, возвращает булев результат TRUE. Транзакции до и после дат возвращают логический результат FALSE.

Пример 2. Скрипт period_no

Скрипт загрузки и результаты

Обзор

Откройте редактор загрузки данных и добавьте приведенный ниже скрипт загрузки на новую вкладку.

Скрипт загрузки содержит следующее:

- Используется тот же набор данных и сценарий, что в первом примере.
- Создание поля `prev_week_to_date`, определяющего, какие транзакции совершены в течение полной недели до сегмента недели, который заканчивается 14 января 2022 года.
- Создание дополнительного поля под именем `weekday` с использованием функции `weekday()`. Это новое поле создано, чтобы показать, какой день недели соответствует каждой дате.

Скрипт загрузки

```
SET FirstWeekDay=6;
SET TimestampFormat='M/D/YYYY h:mm:ss[.fff]';
Transactions:
    Load
        *,
        weekday(date) as week_day,
        inweektodate(date,'01/14/2022', -1) as prev_week_to_date
        ;
Load
*
Inline
[
```

```
id,date,amount
8188,'2022-01-02 12:22:06',37.23
8189,'2022-01-05 01:02:30',17.17
8190,'2022-01-06 15:36:20',88.27
8191,'2022-01-08 10:58:35',57.42
8192,'2022-01-09 08:53:32',53.80
8193,'2022-01-10 21:13:01',82.06
8194,'2022-01-11 00:57:13',40.39
8195,'2022-01-12 09:26:02',87.21
8196,'2022-01-13 15:05:09',95.93
8197,'2022-01-14 18:44:57',45.89
8198,'2022-01-15 06:10:46',36.23
8199,'2022-01-16 06:39:27',25.66
8200,'2022-01-17 10:44:16',82.77
8201,'2022-01-18 18:48:17',69.98
8202,'2022-01-26 04:36:03',76.11
8203,'2022-01-27 08:07:49',25.12
8204,'2022-01-28 12:24:29',46.23
8205,'2022-01-30 11:56:56',84.21
8206,'2022-01-30 14:40:19',96.24
8207,'2022-01-31 05:28:21',67.67
];
```

Результаты

Загрузите данные и откройте лист. Создайте новую таблицу и добавьте эти поля как измерения:

- `date`
- `week_day`
- `prev_week_to_date`

Результирующая таблица

date	week_day	prev_week_to_date
2022-01-02 12:22:06	Sun	-1
2022-01-05 01:02:30	Wed	-1
2022-01-06 15:36:20	Thu	-1
2022-01-08 10:58:35	Sat	0
2022-01-09 08:53:32	Sun	0
2022-01-10 21:13:01	Mon	0
2022-01-11 00:57:13	Tue	0
2022-01-12 09:26:02	Wed	0
2022-01-13 15:05:09	Thu	0
2022-01-14 18:44:57	Fri	0
2022-01-15 06:10:46	Sat	0

date	week_day	prev_week_to_date
2022-01-16 06:39:27	Sun	0
2022-01-17 10:44:16	Mon	0
2022-01-18 18:48:17	Tue	0
2022-01-26 04:36:03	Wed	0
2022-01-27 08:07:49	Thu	0
2022-01-28 12:24:29	Fri	0
2022-01-30 11:56:56	Sun	0
2022-01-30 14:40:19	Sun	0
2022-01-31 05:28:21	Mon	0

Значение `period_no` равное -1 указывает, что функция `inweektodate ()` сравнивает сегмент входной недели с предыдущей неделей. Сегмент недели первоначально соответствует периоду с 9 по 14 января. Затем `period_no` смещает как начальную, так и конечную границу этого сегмента на одну неделю назад, в результате чего границы периода дат переносятся на 2 января и 7 января.

Диаграмма календаря с датами транзакций, которые вернут булев результат TRUE

Sun	Mon	Tue	Wed	Thu	Fri	Sat
						1
2	3	4	5	6	7	8
9	10	11	12	13	14	15
16	17	18	19	20	21	22
23	24	25	26	27	28	29
30	31					

Таким образом, любая транзакция, совершенная в период с 2 до 8 января (не включая 8 января), возвращает булев результат `TRUE`.

Пример 3. Аргумент first_week_day

Скрипт загрузки и результаты

Обзор

Откройте редактор загрузки данных и добавьте приведенный ниже скрипт загрузки на новую вкладку.

Скрипт загрузки содержит следующее:

- Используется тот же набор данных и сценарий, что в первом примере.
- Создание поля `in_week_to_date`, которое определяет, какие транзакции совершены в течение недели до 14 января 2022 года.
- Создание дополнительного поля под именем `weekday` с использованием функции `weekday()`. Это новое поле создано, чтобы показать, какой день недели соответствует каждой дате.

В этом примере мы используем понедельник в качестве первого дня недели.

Скрипт загрузки

```
SET FirstWeekDay=6;
SET TimestampFormat='M/D/YYYY h:mm:ss[.fff]';

Transactions:
    Load
        *,
        weekday(date) as week_day,
        inweektodate(date,'01/14/2022', 0, 0) as in_week_to_date
        ;
Load
*
Inline
[
id,date,amount
8188,'2022-01-02 12:22:06',37.23
8189,'2022-01-05 01:02:30',17.17
8190,'2022-01-06 15:36:20',88.27
8191,'2022-01-08 10:58:35',57.42
8192,'2022-01-09 08:53:32',53.80
8193,'2022-01-10 21:13:01',82.06
8194,'2022-01-11 00:57:13',40.39
8195,'2022-01-12 09:26:02',87.21
8196,'2022-01-13 15:05:09',95.93
8197,'2022-01-14 18:44:57',45.89
8198,'2022-01-15 06:10:46',36.23
8199,'2022-01-16 06:39:27',25.66
8200,'2022-01-17 10:44:16',82.77
8201,'2022-01-18 18:48:17',69.98
8202,'2022-01-26 04:36:03',76.11
8203,'2022-01-27 08:07:49',25.12
8204,'2022-01-28 12:24:29',46.23
8205,'2022-01-30 11:56:56',84.21
8206,'2022-01-30 14:40:19',96.24
8207,'2022-01-31 05:28:21',67.67
];
```

Результаты

Загрузите данные и откройте лист. Создайте новую таблицу и добавьте эти поля как измерения:

- date
- week_day
- in_week_to_date

Результирующая таблица

date	week_day	in_week_to_date
2022-01-02 12:22:06	Sun	0
2022-01-05 01:02:30	Wed	0
2022-01-06 15:36:20	Thu	0
2022-01-08 10:58:35	Sat	0
2022-01-09 08:53:32	Sun	0
2022-01-10 21:13:01	Mon	-1
2022-01-11 00:57:13	Tue	-1
2022-01-12 09:26:02	Wed	-1
2022-01-13 15:05:09	Thu	-1
2022-01-14 18:44:57	Fri	-1
2022-01-15 06:10:46	Sat	0
2022-01-16 06:39:27	Sun	0
2022-01-17 10:44:16	Mon	0
2022-01-18 18:48:17	Tue	0
2022-01-26 04:36:03	Wed	0
2022-01-27 08:07:49	Thu	0
2022-01-28 12:24:29	Fri	0
2022-01-30 11:56:56	Sun	0
2022-01-30 14:40:19	Sun	0
2022-01-31 05:28:21	Mon	0

При использовании 0 в качестве аргумента `inweektodate()` в функции `first_week_day` аргумент функции заменяет системную переменную `FirstWeekDay` и устанавливает понедельник в качестве первого дня недели.

Диаграмма календаря с датами транзакций, которые вернут булев результат TRUE

Mon	Tue	Wed	Thu	Fri	Sat	Sun
					1	2
3	4	5	6	7	8	9
10	11	12	13	14	15	16
17	18	19	20	21	22	17
24	25	26	27	28	29	30
31						

Таким образом, для любой транзакции, совершенной между 10 марта и 14 января, возвращается булев результат TRUE, а для транзакций с датами вне этих границ – значение FALSE.

Пример 4. Пример объекта диаграммы

Скрипт загрузки и выражение диаграммы

Обзор

Откройте редактор загрузки данных и добавьте приведенный ниже скрипт загрузки на новую вкладку.

Скрипт загрузки содержит тот же набор данных и сценарий, что в первом примере. Однако в этом примере в приложение загружается неизмененный набор данных. Расчет, который определяет, совершены ли транзакции на той неделе до 14 января 2022 года, создается как мера в объекте диаграммы в приложении.

Скрипт загрузки

```
SET DateFormat='MM/DD/YYYY';

Transactions:
Load
*
Inline
[
id,date,amount
8188,'2022-01-02 12:22:06',37.23
8189,'2022-01-05 01:02:30',17.17
8190,'2022-01-06 15:36:20',88.27
8191,'2022-01-08 10:58:35',57.42
8192,'2022-01-09 08:53:32',53.80
8193,'2022-01-10 21:13:01',82.06
```

```
8194,'2022-01-11 00:57:13',40.39
8195,'2022-01-12 09:26:02',87.21
8196,'2022-01-13 15:05:09',95.93
8197,'2022-01-14 18:44:57',45.89
8198,'2022-01-15 06:10:46',36.23
8199,'2022-01-16 06:39:27',25.66
8200,'2022-01-17 10:44:16',82.77
8201,'2022-01-18 18:48:17',69.98
8202,'2022-01-26 04:36:03',76.11
8203,'2022-01-27 08:07:49',25.12
8204,'2022-01-28 12:24:29',46.23
8205,'2022-01-30 11:56:56',84.21
8206,'2022-01-30 14:40:19',96.24
8207,'2022-01-31 05:28:21',67.67
];
```

Результаты

Выполните следующие действия.

1. Загрузите данные и откройте лист. Создайте новую таблицу и добавьте это поле как измерение: `date`.
2. Чтобы рассчитать, происходили ли транзакции на той же неделе до 14 января, создайте следующую меру:
   `=inweektodate(date,'01/14/2022',0)`
3. Чтобы показать, какой день недели соответствует каждой дате, создайте дополнительную меру.
   `=weekday(date)`

Результирующая таблица

date	week_day	in_week_to_date
2022-01-02 12:22:06	Sun	0
2022-01-05 01:02:30	Wed	0
2022-01-06 15:36:20	Thu	0
2022-01-08 10:58:35	Sat	0
2022-01-09 08:53:32	Sun	-1
2022-01-10 21:13:01	Mon	-1
2022-01-11 00:57:13	Tue	-1
2022-01-12 09:26:02	Wed	-1
2022-01-13 15:05:09	Thu	-1
2022-01-14 18:44:57	Fri	-1
2022-01-15 06:10:46	Sat	0

date	week_day	in_week_to_date
2022-01-16 06:39:27	Sun	0
2022-01-17 10:44:16	Mon	0
2022-01-18 18:48:17	Tue	0
2022-01-26 04:36:03	Wed	0
2022-01-27 08:07:49	Thu	0
2022-01-28 12:24:29	Fri	0
2022-01-30 11:56:56	Sun	0
2022-01-30 14:40:19	Sun	0
2022-01-31 05:28:21	Mon	0

Поле `in_week_to_date` создается в объекте диаграммы в качестве меры с помощью функции `inweektodate()`. Первый предоставленный аргумент определяет, какое поле оценивается. Второй аргумент – это жестко закодированная дата «14 мая», которая представляет собой `base_date`, определяющий, какую неделю сегментировать, и определяющий конечную границу этого сегмента. `period_no` = 0 – это последний аргумент, означающий, что функция не сравнивает недели, предшествующие или следующие за сегментированной недели.

Системная переменная `FirstWeekDay` определяет, что недели начинаются в воскресенье и заканчиваются в субботу. Таким образом, январь будет разбит на недели в соответствии с приведенной ниже диаграммой, где даты между 9 и 14 января будут действительным периодом для расчета `inweekdodate()`:

Диаграмма календаря с датами транзакций, которые вернут булев результат TRUE

Sun	Mon	Tue	Wed	Thu	Fri	Sat
						1
2	3	4	5	6	7	8
9	10	11	12	13	14	15
16	17	18	19	20	21	22
23	24	25	26	27	28	29
30	31					

Любая транзакция, совершенная в период с 9 апреля по 14 января, возвращает булев результат `TRUE`. Транзакции до и после дат возвращают булев результат `FALSE`.

Пример 5. Сценарий

Скрипт загрузки и выражение диаграммы

Обзор

Откройте редактор загрузки данных и добавьте приведенный ниже скрипт загрузки на новую вкладку.

Скрипт загрузки содержит следующее:

- Набор данных, загруженный в таблицу, называется `Products`
- Информация об идентификаторе продукта, дате изготовления и себестоимости.

Установлено, что вследствие ошибки оборудования изделия, изготовленные в течение недели, на которую выпадает 12 января, являются дефектными. Проблема была решена 13 января. Конечному пользователю требуется объект диаграммы, который отображает по неделе, какие изготовленные продукты были дефектными, а какие бездефектными, а также стоимость продуктов, изготовленных на этой неделе.

Скрипт загрузки

```
Products:
Load
*
Inline
[
product_id,manufacture_date,cost_price
8188,'2022-01-02 12:22:06',37.23
8189,'2022-01-05 01:02:30',17.17
8190,'2022-01-06 15:36:20',88.27
8191,'2022-01-08 10:58:35',57.42
8192,'2022-01-09 08:53:32',53.80
8193,'2022-01-10 21:13:01',82.06
8194,'2022-01-11 00:57:13',40.39
8195,'2022-01-12 09:26:02',87.21
8196,'2022-01-13 15:05:09',95.93
8197,'2022-01-14 18:44:57',45.89
8198,'2022-01-15 06:10:46',36.23
8199,'2022-01-16 06:39:27',25.66
8200,'2022-01-17 10:44:16',82.77
8201,'2022-01-18 18:48:17',69.98
8202,'2022-01-26 04:36:03',76.11
8203,'2022-01-27 08:07:49',25.12
8204,'2022-01-28 12:24:29',46.23
8205,'2022-01-30 11:56:56',84.21
8206,'2022-01-30 14:40:19',96.24
8207,'2022-01-31 05:28:21',67.67
];
```

Результаты

Выполните следующие действия.

1. Загрузите данные и откройте лист. Создайте новую таблицу. Создайте измерение для отображения названий недель:
   `=weekname(manufacture_date)`
2. Затем создайте измерение, чтобы определить, какие из продуктов дефектные, а какие нет:
   `=if(inWeektodate(manufacture_date,makedate(2022,01,12),0),'Defective','Faultless')`
3. Создайте меру для суммирования `cost_price` продуктов:
   `=sum(cost_price)`
4. Задайте параметру меры **Формат чисел** значение **Денежный**.

Результирующая таблица

weekname(manufacture_date)	if(inweektodate(manufacture_date,makedate (2022,01,12),0),'Defective','Faultless')	Sum(cost_price)
2022/02	Бездефектный	$200.09
2022/03	Дефектный	$263.46
2022/03	Бездефектный	$178.05
2022/04	Бездефектный	$178.41
2022/05	Бездефектный	$147.46
2022/06	Бездефектный	$248.12

Функция `inweektodate()` возвращает логическое значение при проверки дат производства каждого продукта. Продукты, для которых возвращается булево значение `TRUE`, помечаются как `'Defective'`. Продукты, для которых возвращается значение `FALSE` и которые, следовательно, не произведены в течение недели до 12 января, помечаются как `'Faultless'`.

inyear

Эта функция возвращает значение True, если поле **timestamp** находится в пределах года, включающего значение, указанное в поле **base_date**.

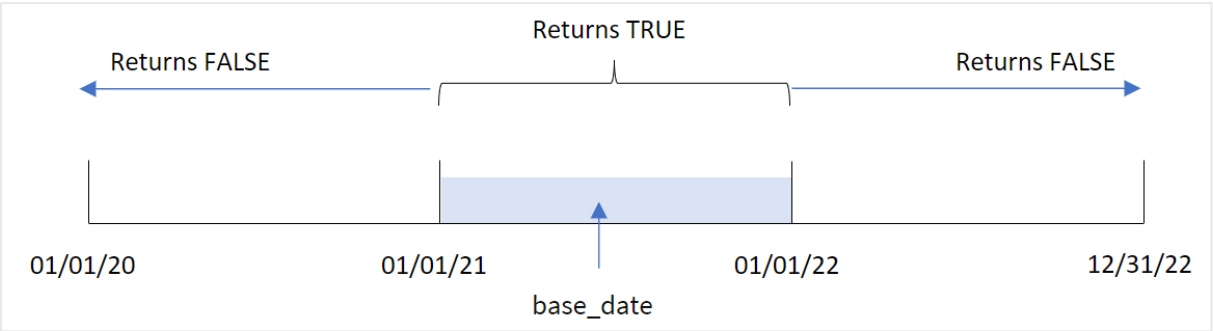
Синтаксис:

```
InYear (timestamp, base_date, period_no [, first_month_of_year])
```

Возвращаемые типы данных: Булево значение

В Qlik Sense логическое значение «истина» представлено как -1, а «ложь» – как 0.

Диаграмма диапазона функции `inyear()`



Функция `inyear()` возвращает булев результат сравнения выбранных значений дат с годом, заданным в аргументе `base_date`.

Когда это следует использовать

Функция `inyear()` возвращает результат в виде булева значения. Обычно этот тип функции используется в качестве условия в `if expression`. Она возвращает агрегацию или вычисление в зависимости от того, относится ли проверяемая дата к рассматриваемому году. Например, с помощью функции `inyear()` можно идентифицировать все продажи, совершенные в определенному году.

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**.

Можно использовать следующие значения, чтобы задать первый месяц года в аргументе `first_month_of_year`:

значения first_month_of_year

Месяц	Значение
Февраль	2
Март	3
Апрель	4
May	5

Месяц	Значение
Июнь	6
Июль	7
Август	8
Сентябрь	9
Октябрь	10
Ноябрь	11
Декабрь	12

Региональные настройки

Если не указано иное, в примерах, приведенных в данном разделе, используется следующий формат даты: ММ/ДД/ГГГГ. Формат даты указан в операторе `SET DateFormat` скрипта загрузки данных. В вашей системе может быть установлен другой формат даты по умолчанию в зависимости от региональных настроек и других факторов. Можно изменить формат в примерах в соответствии с потребностями. Или можно изменить форматы в скрипте загрузки в соответствии с этими примерами.

Региональные настройки по умолчанию в приложениях основаны на системных региональных настройках компьютера или сервера, где установлено ПО Qlik Sense. Если на сервере Qlik Sense, к которому обращается пользователь, выбран шведский язык, то редактор загрузки данных будет использовать шведские региональные настройки для даты, времени и валюты. Эти параметры регионального формата не связаны с языком, отображаемым в интерфейсе пользователя Qlik Sense. Qlik Sense будет отображаться на языке, который используется в браузере.

Примеры функции

Пример	Результат
`inyear ('01/25/2013', '01/01/2013', 0 )`	Возвращает TRUE
`inyear ('01/25/2012', '01/01/2013', 0)`	Возвращает FALSE
`inyear ('01/25/2013', '01/01/2013', -1)`	Возвращает FALSE
`inyear ('01/25/2012', '01/01/2013', -1 )`	Возвращает TRUE
`inyear ('01/25/2013', '01/01/2013', 0, 3)`	Возвращает TRUE Значения `base_date` и `first_month_of_year` указывают, что метка времени должна попадать в период с 01/03/2012 по 02/28/2013
`inyear ('03/25/2013', '07/01/2013', 0, 3 )`	Возвращает TRUE

Пример 1. Базовый пример

Скрипт загрузки и результаты

Обзор

Откройте редактор загрузки данных и добавьте приведенный ниже скрипт загрузки на новую вкладку.

Скрипт загрузки содержит следующее:

- Набор данных, содержащий набор транзакций с 2020 по 2022 год, загруженный в таблицу под именем `Transactions`.
- Предыдущая загрузка с функцией `inyear()`, которая задана в качестве поля `in_year` и определяет, какие транзакции совершены в том же году, к которому относится 26 июля 2021 года.

Скрипт загрузки

```
SET DateFormat='MM/DD/YYYY';
Transactions:
    Load
        *,
        inyear(date,'07/26/2021', 0) as in_year
        ;
Load
*
Inline
[
id,date,amount
8188,'01/13/2020',37.23
8189,'02/26/2020',17.17
8190,'03/27/2020',88.27
8191,'04/16/2020',57.42
8192,'05/21/2020',53.80
8193,'08/14/2020',82.06
8194,'10/07/2020',40.39
8195,'12/05/2020',87.21
8196,'01/22/2021',95.93
8197,'02/03/2021',45.89
8198,'03/17/2021',36.23
8199,'04/23/2021',25.66
8200,'05/04/2021',82.77
8201,'06/30/2021',69.98
8202,'07/26/2021',76.11
8203,'12/27/2021',25.12
8204,'06/06/2022',46.23
8205,'07/18/2022',84.21
8206,'11/14/2022',96.24
8207,'12/12/2022',67.67
];
```

Результаты

Загрузите данные и откройте лист. Создайте новую таблицу и добавьте эти поля как измерения:

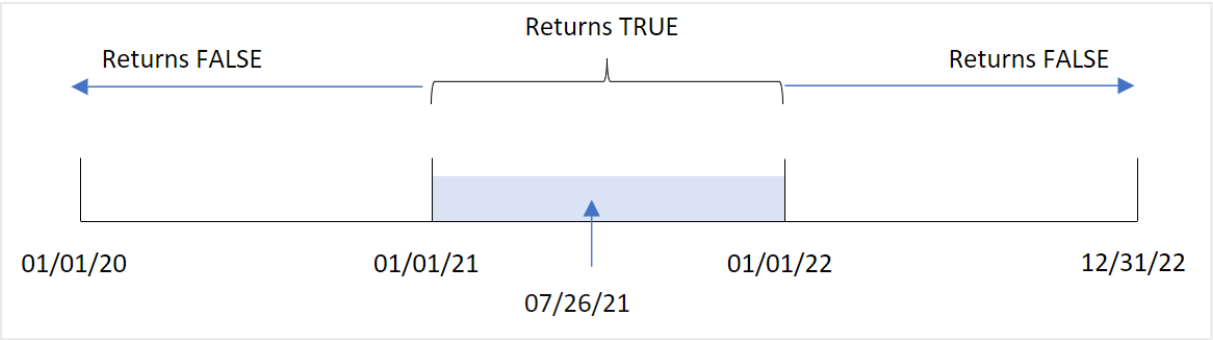
- `date`
- `in_year`

Результирующая таблица

date	in_year
01/13/2020	0
02/26/2020	0
03/27/2020	0
04/16/2020	0
05/21/2020	0
08/14/2020	0
10/07/2020	0
12/05/2020	0
01/22/2021	-1
02/03/2021	-1
03/17/2021	-1
04/23/2021	-1
05/04/2021	-1
06/30/2021	-1
07/26/2021	-1
12/27/2021	-1
06/06/2022	0
07/18/2022	0
11/14/2022	0
12/12/2022	0

Поле `in_year` создано в предыдущем операторе load с помощью функции `inyear()`. Первый аргумент определяет, какое поле оценивается. Второй аргумент — это жестко закодированная дата 26 июля 2021 года, которая представляет собой `base_date`, определяющий сравниваемый год. `period_no= 0` — это последний аргумент, означающий, что функция `inyear()` не сравнивает годы, которые следуют до или после рассматриваемого года.

Диаграмма диапазона функции `inyear()` с базовой датой 26 июля



Любая транзакция, совершенная в 2021 году, будет возвращать булев результат TRUE.

Пример 2. Аргумент period_no

Скрипт загрузки и результаты

Обзор

Откройте редактор загрузки данных и добавьте приведенный ниже скрипт загрузки на новую вкладку.

Скрипт загрузки содержит следующее:

- Набор данных, содержащий набор транзакций с 2020 по 2022 год, загруженный в таблицу под именем `Transactions`.
- Предыдущая загрузка с функцией `inyear()`, которая задана в качестве поля `previous_year` и определяет, какие транзакции совершены за 12 месяцев до года, к которому относится 26 июля 2021 года.

Скрипт загрузки

```
SET DateFormat='MM/DD/YYYY';
Transactions:
    Load
        *,
        inyear(date,'07/26/2021', -1) as previous_year
        ;
Load
*
Inline
[
id,date,amount
8188,'01/13/2020',37.23
8189,'02/26/2020',17.17
8190,'03/27/2020',88.27
8191,'04/16/2020',57.42
8192,'05/21/2020',53.80
8193,'08/14/2020',82.06
8194,'10/07/2020',40.39
```

```
8195,'12/05/2020',87.21
8196,'01/22/2021',95.93
8197,'02/03/2021',45.89
8198,'03/17/2021',36.23
8199,'04/23/2021',25.66
8200,'05/04/2021',82.77
8201,'06/30/2021',69.98
8202,'07/26/2021',76.11
8203,'12/27/2021',25.12
8204,'06/06/2022',46.23
8205,'07/18/2022',84.21
8206,'11/14/2022',96.24
8207,'12/12/2022',67.67
];
```

Результаты

Загрузите данные и откройте лист. Создайте новую таблицу и добавьте эти поля как измерения:

- `date`
- `previous_year`

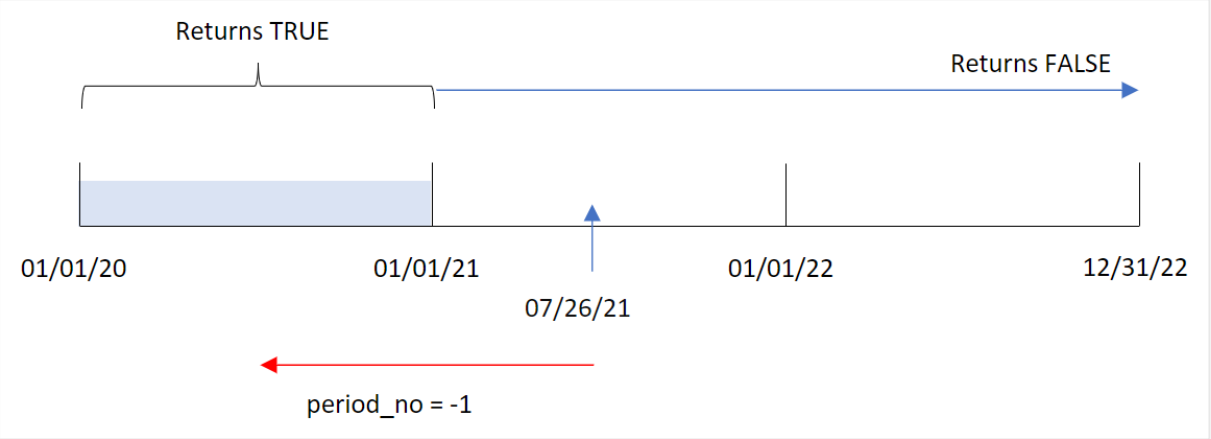
Результирующая таблица

date	previous_year
01/13/2020	-1
02/26/2020	-1
03/27/2020	-1
04/16/2020	-1
05/21/2020	-1
08/14/2020	-1
10/07/2020	-1
12/05/2020	-1
01/22/2021	0
02/03/2021	0
03/17/2021	0
04/23/2021	0
05/04/2021	0
06/30/2021	0
07/26/2021	0
12/27/2021	0
06/06/2022	0

date	previous_year
07/18/2022	0
11/14/2022	0
12/12/2022	0

При использовании `-1` в качестве аргумента `period_no` функции `inyear()` границы сравниваемого года сдвигаются на полный год назад. Сначала в качестве сравниваемого года был определен 2021 год. `period_no` сдвигает сравниваемый год на один, теперь это 2020 год.

Диаграмма диапазона функции `inyear()` с аргументом period_no = -1



Любая транзакция, совершенная в 2020 году, возвращает булев результат TRUE.

Пример 3. Аргумент first_month_of_year

Скрипт загрузки и результаты

Обзор

Откройте редактор загрузки данных и добавьте приведенный ниже скрипт загрузки на новую вкладку.

Скрипт загрузки содержит следующее:

- Набор данных, содержащий набор транзакций с 2020 по 2022 год, загруженный в таблицу под именем `Transactions`.
- Предыдущая загрузка с функцией `inyear()`, которая задана в качестве поля `in_year` и определяет, какие транзакции совершены в том же году, к которому относится 26 июля 2021 года.

Однако в этом примере политика организации устанавливает март в качестве первого месяца финансового года.

Скрипт загрузки

```
SET DateFormat='MM/DD/YYYY';
Transactions:
    Load
        *,
        inyear(date,'07/26/2021', 0, 3) as in_year
        ;
Load
*
Inline
[
id,date,amount
8188,'01/13/2020',37.23
8189,'02/26/2020',17.17
8190,'03/27/2020',88.27
8191,'04/16/2020',57.42
8192,'05/21/2020',53.80
8193,'08/14/2020',82.06
8194,'10/07/2020',40.39
8195,'12/05/2020',87.21
8196,'01/22/2021',95.93
8197,'02/03/2021',45.89
8198,'03/17/2021',36.23
8199,'04/23/2021',25.66
8200,'05/04/2021',82.77
8201,'06/30/2021',69.98
8202,'07/26/2021',76.11
8203,'12/27/2021',25.12
8204,'06/06/2022',46.23
8205,'07/18/2022',84.21
8206,'11/14/2022',96.24
8207,'12/12/2022',67.67
];
```

Результаты

Загрузите данные и откройте лист. Создайте новую таблицу и добавьте эти поля как измерения:

- `date`
- `in_year`

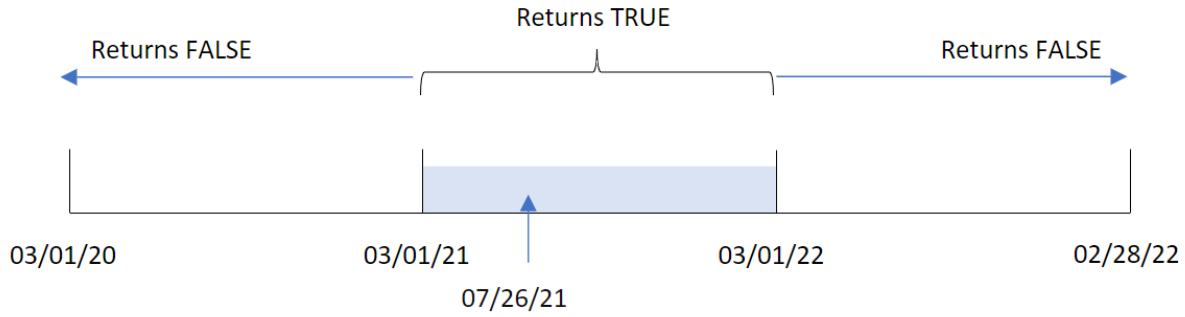
Результирующая таблица

date	in_year
01/13/2020	0
02/26/2020	0
03/27/2020	0
04/16/2020	0

date	in_year
05/21/2020	0
08/14/2020	0
10/07/2020	0
12/05/2020	0
01/22/2021	0
02/03/2021	0
03/17/2021	-1
04/23/2021	-1
05/04/2021	-1
06/30/2021	-1
07/26/2021	-1
12/27/2021	-1
06/06/2022	0
07/18/2022	0
11/14/2022	0
12/12/2022	0

При использовании `3` в качестве аргумента `first_month_of_year` функции `inyear()` год начинается с 1 марта и заканчивается в последний день февраля.

Диаграмма диапазона функции `inyear()`, где в качестве первого месяца года задан март.



Таким образом, любая транзакция, совершенная в период с 1 марта 2021 года до 1 марта 2022 года, возвращает булев результат TRUE.

Пример 4. Пример объекта диаграммы

Скрипт загрузки и выражение диаграммы

Обзор

Используется тот же набор данных и сценарий, что в первом примере.

Однако в этом примере в приложение загружается неизмененный набор данных. Расчет, который определяет, совершены ли транзакции в том же году, к которому относится 26 июля 2021 года, создается в качестве меры в объекте диаграммы в приложении.

Скрипт загрузки

```
SET DateFormat='MM/DD/YYYY';
Transactions:
Load
*
Inline
[
id,date,amount
8188,'01/13/2020',37.23
8189,'02/26/2020',17.17
8190,'03/27/2020',88.27
8191,'04/16/2020',57.42
8192,'05/21/2020',53.80
8193,'08/14/2020',82.06
8194,'10/07/2020',40.39
8195,'12/05/2020',87.21
8196,'01/22/2021',95.93
8197,'02/03/2021',45.89
8198,'03/17/2021',36.23
8199,'04/23/2021',25.66
8200,'05/04/2021',82.77
8201,'06/30/2021',69.98
8202,'07/26/2021',76.11
8203,'12/27/2021',25.12
8204,'06/06/2022',46.23
8205,'07/18/2022',84.21
8206,'11/14/2022',96.24
8207,'12/12/2022',67.67
];
```

Результаты

Загрузите данные и откройте лист. Создайте новую таблицу и добавьте это поле как измерение:

- `date`

Чтобы рассчитать, совершены ли транзакции в том же году, к которому относится 26 июля 2021 года, создайте следующую меру:

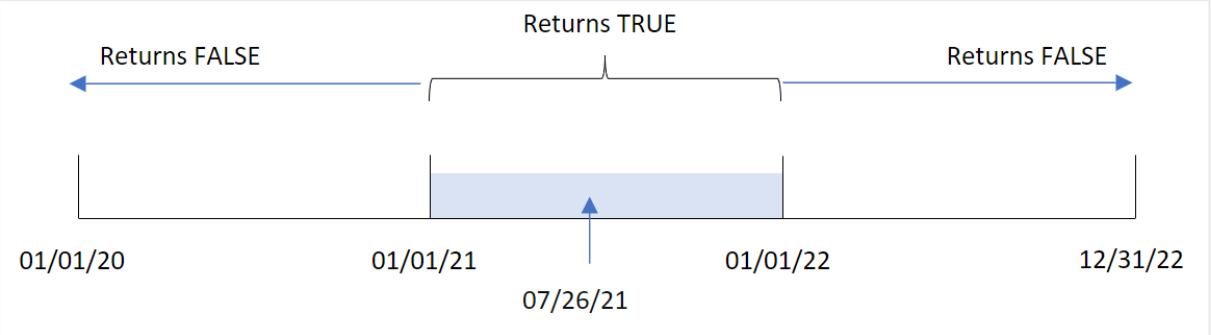
- =inyear(date,'07/26/2021', 0)

Результирующая таблица

date	=inyear(date,'07/26/2021',0)
01/13/2020	0
02/26/2020	0
03/27/2020	0
04/16/2020	0
05/21/2020	0
08/14/2020	0
10/07/2020	0
12/05/2020	0
01/22/2021	-1
02/03/2021	-1
03/17/2021	-1
04/23/2021	-1
05/04/2021	-1
06/30/2021	-1
07/26/2021	-1
12/27/2021	-1
06/06/2022	0
07/18/2022	0
11/14/2022	0
12/12/2022	0

Поле `in_year` создается в диаграмме с помощью функции `inyear()`. Первый аргумент определяет, какое поле оценивается. Второй аргумент – это жестко закодированная дата 26 июля 2021 года, которая представляет собой `base_date`, определяющий сравниваемый год. `period_no= 0` – это последний аргумент, означающий, что функция `inyear()` не сравнивает годы, которые следуют до или после рассматриваемого года.

Диаграмма диапазона функции `inyear()` с базовой датой 27 июля



Любая транзакция, совершенная в 2021 году, будет возвращать булев результат TRUE.

Пример 5. Сценарий

Скрипт загрузки и результаты

Обзор

Откройте редактор загрузки данных и добавьте приведенный ниже скрипт загрузки на новую вкладку.

Скрипт загрузки содержит следующее:

- Набор данных, загруженный в таблицу под именем `Products`.
- Данная таблица содержит следующие поля:
  - product ID (ИД продукта)
  - product type (тип продукта)
  - manufacture date (дата изготовления)
  - cost price (себестоимость)

Конечному пользователю нужен объект диаграммы, отображающий по типу продукта стоимость продуктов, произведенных в 2021 году.

Скрипт загрузки

```
Products:
Load
*
Inline
[
product_id,product_type,manufacture_date,cost_price
8188,product A,'01/13/2020',37.23
8189,product B,'02/26/2020',17.17
8190,product B,'03/27/2020',88.27
8191,product C,'04/16/2020',57.42
8192,product D,'05/21/2020',53.80
8193,product D,'08/14/2020',82.06
8194,product C,'10/07/2020',40.39
```

```
8195,product B,'12/05/2020',87.21
8196,product A,'01/22/2021',95.93
8197,product B,'02/03/2021',45.89
8198,product C,'03/17/2021',36.23
8199,product C,'04/23/2021',25.66
8200,product B,'05/04/2021',82.77
8201,product D,'06/30/2021',69.98
8202,product D,'07/26/2021',76.11
8203,product D,'12/27/2021',25.12
8204,product C,'06/06/2022',46.23
8205,product C,'07/18/2022',84.21
8206,product A,'11/14/2022',96.24
8207,product B,'12/12/2022',67.67
];
```

Результаты

Загрузите данные и откройте лист. Создайте новую таблицу и добавьте это поле как измерение:

- `product_type`

Чтобы вычислить сумму для каждого продукта, произведенного в 2021 году, создайте следующую меру:

- `=sum(if(InYear(manufacture_date,makedate(2021,01,01),0),cost_price,0))`

Выполните следующие действия.

1. Задайте параметру **Формат чисел** меры значение **Денежный**.
2. В области **Вид** выключите параметр **Итоги**.

Результирующая таблица

product_type	=sum(if(InYear(manufacture_date,makedate(2021,01,01),0),cost_price,0))
product A	$95.93
product B	$128.66
product C	$61.89
product D	$171.21

Функция `inyear()` возвращает логическое значение при проверке дат производства каждого продукта. Для любого продукта, изготовленного в 2021 году, функция `inyear()` возвращает булево значение TRUE и сумму `cost_price`.

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.

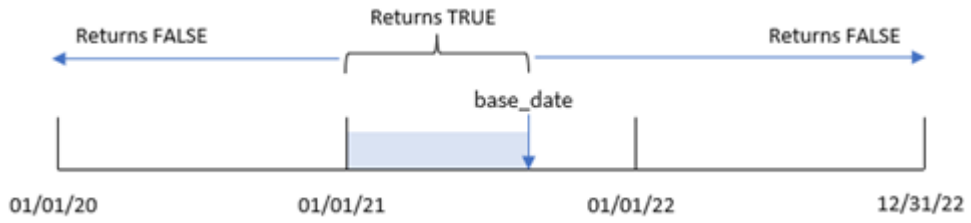
Синтаксис:

```
InYearToDate (timestamp, base_date, period_no[, first_month_of_year])
```

Возвращаемые типы данных: Булево значение

> *В Qlik Sense логическое значение «истина» представлено как -1, а «ложь» – как 0.*

Диаграмма функции inyeartodate



Функция `inyeartodate()` сегментирует конкретный период года, а `base_date` определяет максимально допустимую дату для этого сегмента года. Затем функция оценивает, попадает ли поле даты или значение в этот сегмент, и возвращает булев результат.

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**.

Когда это следует использовать

Функция `inyeartodate()` возвращает результат в виде логического значения. Обычно этот тип функции используется в качестве условия в выражении `if`. Это возвращает агрегирование или расчет в зависимости от того, попадает ли проверяемая дата в год до рассматриваемой даты включительно.

Например, функцию `inyeartodate()` можно использовать для идентификации всего оборудования, изготовленного в указанном году до определенной даты.

В этих примерах используется формат даты MM/DD/YYYY. Формат даты указан в операторе `SET DateFormat` в верхней части скрипта загрузки данных. Измените формат в примерах согласно своим пожеланиям.

Примеры функции

Пример	Результат
`inyeartodate ('01/25/2013', '02/01/2013', 0)`	Возвращает `TRUE`.
`inyeartodate ('01/25/2012', '01/01/2013', 0)`	Возвращает `FALSE`.
`inyeartodate ('01/25/2012', '02/01/2013', -1)`	Возвращает `TRUE`.
`inyeartodate ('11/25/2012', '01/31/2013', 0, 4)`	Возвращает `TRUE`. Значение `timestamp` выпадает на финансовый год, начинающийся в четвертом месяце, и до значения `base_date`.
`inyeartodate ('3/31/2013', '01/31/2013', 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.

Региональные настройки

Если не указано иное, в примерах, приведенных в данном разделе, используется следующий формат даты: ММ/ДД/ГГГГ. Формат даты указан в операторе `SET DateFormat` скрипта загрузки данных. В вашей системе может быть установлен другой формат даты по умолчанию в зависимости от региональных настроек и других факторов. Можно изменить формат в примерах в соответствии с потребностями. Или можно изменить форматы в скрипте загрузки в соответствии с этими примерами.

Региональные настройки по умолчанию в приложениях основаны на системных региональных настройках компьютера или сервера, где установлено ПО Qlik Sense. Если на сервере Qlik Sense, к которому обращается пользователь, выбран шведский язык, то редактор загрузки данных будет использовать шведские региональные настройки для даты, времени и валюты. Эти параметры регионального формата не связаны с языком, отображаемым в интерфейсе пользователя Qlik Sense. Qlik Sense будет отображаться на языке, который используется в браузере.

Пример 1. Без дополнительных аргументов

Скрипт загрузки и результаты

Обзор

Откройте редактор загрузки данных и добавьте приведенный ниже скрипт загрузки на новую вкладку.

Скрипт загрузки содержит следующее:

- Набор данных, содержащий набор транзакций с 2020 по 2022 год, загружается в таблицу под именем `Transactions`.
- Поле даты было предоставлено в формате системной переменной `DateFormat` (MM/DD/YYYY).
- Создание поля `in_year_to_date`, которое определяет, какие транзакции совершены в течение года до 26 июля 2021 года.

Скрипт загрузки

```
SET DateFormat='MM/DD/YYYY';

Transactions:
    Load
        *,
        inyeartodate(date,'07/26/2021', 0) as in_year_to_date
        ;
Load
*
Inline
[
id,date,amount
8188,'01/13/2020',37.23
8189,'02/26/2020',17.17
8190,'03/27/2020',88.27
8191,'04/16/2020',57.42
8192,'05/21/2020',53.80
8193,'06/14/2020',82.06
8194,'08/07/2020',40.39
8195,'09/05/2020',87.21
8196,'01/22/2021',95.93
8197,'02/03/2021',45.89
8198,'03/17/2021',36.23
8199,'04/23/2021',25.66
8200,'05/04/2021',82.77
8201,'06/30/2021',69.98
8202,'07/26/2021',76.11
8203,'07/27/2021',25.12
8204,'06/06/2022',46.23
8205,'07/18/2022',84.21
8206,'11/14/2022',96.24
8207,'12/12/2022',67.67
];
```

Результаты

Загрузите данные и откройте лист. Создайте новую таблицу и добавьте эти поля как измерения:

- `date`
- `in_year_to_date`

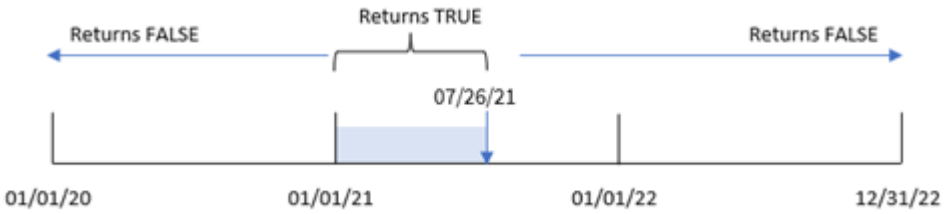
Результирующая таблица

date	in_year_to_date
01/13/2020	0
02/26/2020	0
03/27/2020	0
04/16/2020	0
05/21/2020	0
06/14/2020	0
08/07/2020	0
09/05/2020	0
01/22/2021	-1
02/03/2021	-1
03/17/2021	-1
04/23/2021	-1
05/04/2021	-1
06/30/2021	-1
07/26/2021	-1
07/27/2021	0
06/06/2022	0
07/18/2022	0
11/14/2022	0
12/12/2022	0

Поле `in_year_to_date` создано в предыдущем операторе load с помощью функции `inyeartodate()`. Первый предоставленный аргумент определяет, какое поле оценивается.

Второй аргумент – это жестко закодированная дата «26 июля 2021 года», которая представляет собой `base_date`, определяющий конечную границу этого сегмента года. `period_no` = 0 – это последний аргумент, означающий, что функция не сравнивает год, предшествующие или следующие за сегментированным годом.

Диаграмма функции inyeartodate без дополнительных аргументов



Любая транзакция, совершенная в период с 1 января по 26 июля, возвращает булев результат TRUE. Даты транзакций до 2021 года и после 26 июля 2021 года возвращают FALSE.

Пример 2. Скрипт period_no

Скрипт загрузки и результаты

Обзор

Откройте редактор загрузки данных и добавьте приведенный ниже скрипт загрузки на новую вкладку.

Скрипт загрузки содержит следующее:

- Используется тот же набор данных и сценарий, что в первом примере.
- Создание поля `previous_year_to_date`, определяющего, какие транзакции совершены в течение полного года до сегмента года, который заканчивается 26 июля 2021 года.

Скрипт загрузки

```
SET DateFormat='MM/DD/YYYY';

Transactions:
    Load
        *,
        inyeartodate(date,'07/26/2021', -1) as previous_year_to_date
        ;
Load
*
Inline
[
id,date,amount
8188,'01/13/2020',37.23
8189,'02/26/2020',17.17
8190,'03/27/2020',88.27
8191,'04/16/2020',57.42
8192,'05/21/2020',53.80
8193,'06/14/2020',82.06
8194,'08/07/2020',40.39
8195,'09/05/2020',87.21
8196,'01/22/2021',95.93
8197,'02/03/2021',45.89
```

```
8198,'03/17/2021',36.23
8199,'04/23/2021',25.66
8200,'05/04/2021',82.77
8201,'06/30/2021',69.98
8202,'07/26/2021',76.11
8203,'07/27/2021',25.12
8204,'06/06/2022',46.23
8205,'07/18/2022',84.21
8206,'11/14/2022',96.24
8207,'12/12/2022',67.67
];
```

Результаты

Загрузите данные и откройте лист. Создайте новую таблицу и добавьте эти поля как измерения:

- `date`
- `previous_year_to_date`

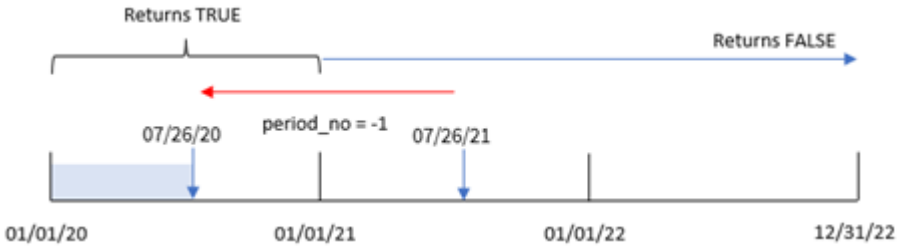
Результирующая таблица

date	previous_year_to_date
01/13/2020	-1
02/26/2020	-1
03/27/2020	-1
04/16/2020	-1
05/21/2020	-1
06/14/2020	-1
08/07/2020	0
09/05/2020	0
01/22/2021	0
02/03/2021	0
03/17/2021	0
04/23/2021	0
05/04/2021	0
06/30/2021	0
07/26/2021	0
07/27/2021	0
06/06/2022	0
07/18/2022	0

date	previous_year_to_date
11/14/2022	0
12/12/2022	0

Значение `period_no` равное -1 указывает, что функция `inyeartodate ()` сравнивает сегмент входного года с предыдущим годом. При входной дате 26 июля 2021 г. сегмент с 1 января по 26 июля 2021 года изначально был определен как период текущего года до сегодняшнего дня. Затем `period_no` смещает этот сегмент на полный год назад, в результате чего границы диапазона дат переносятся на 1 января и 26 июля 2020 года.

Диаграмма функции inyeartodate, пример с аргументом period_no



Таким образом, любая транзакция, совершенная в период с 1 января по 26 июля, возвращает булев результат `TRUE`.

Пример 3. Аргумент first_month_of_year

Скрипт загрузки и результаты

Обзор

Откройте редактор загрузки данных и добавьте приведенный ниже скрипт загрузки на новую вкладку.

Скрипт загрузки содержит следующее:

- Используется тот же набор данных и сценарий, что в первом примере.
- Создание поля `in_year_to_date`, которое определяет, какие транзакции совершены в течение того же года до 26 июля 2021 года.

В этом примере мы устанавливаем март в качестве первого месяца финансового года.

Скрипт загрузки

```
SET DateFormat='MM/DD/YYYY';

Transactions:
    Load
        *,
        inyeartodate(date,'07/26/2021', 0,3) as in_year_to_date
        ;
```

```
Load
*
Inline
[
id,date,amount
8188,'01/13/2020',37.23
8189,'02/26/2020',17.17
8190,'03/27/2020',88.27
8191,'04/16/2020',57.42
8192,'05/21/2020',53.80
8193,'06/14/2020',82.06
8194,'08/07/2020',40.39
8195,'09/05/2020',87.21
8196,'01/22/2021',95.93
8197,'02/03/2021',45.89
8198,'03/17/2021',36.23
8199,'04/23/2021',25.66
8200,'05/04/2021',82.77
8201,'06/30/2021',69.98
8202,'07/26/2021',76.11
8203,'07/27/2021',25.12
8204,'06/06/2022',46.23
8205,'07/18/2022',84.21
8206,'11/14/2022',96.24
8207,'12/12/2022',67.67
];
```

Результаты

Загрузите данные и откройте лист. Создайте новую таблицу и добавьте эти поля как измерения:

- `date`
- `in_year_to_date`

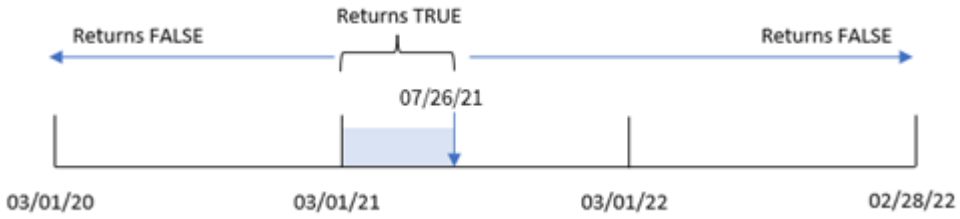
Результирующая таблица

date	in_year_to_date
01/13/2020	0
02/26/2020	0
03/27/2020	0
04/16/2020	0
05/21/2020	0
06/14/2020	0
08/07/2020	0
09/05/2020	0
01/22/2021	0

date	in_year_to_date
02/03/2021	0
03/17/2021	-1
04/23/2021	-1
05/04/2021	-1
06/30/2021	-1
07/26/2021	-1
07/27/2021	0
06/06/2022	0
07/18/2022	0
11/14/2022	0
12/12/2022	0

Используя 3 в качестве аргумента `first_month_of_year` в функции `inyeartodate()`, функция начинает год 1 марта. Затем `base_date` устанавливает дату окончания 26 июля 2021 года для этого сегмента года.

Диаграмма функции inyeartodate, пример с аргументом first_month_of_year



Таким образом, для любой транзакции, совершенной между 1 марта и 26 июля, возвращается булев результат `TRUE`, а для транзакций с датами вне этих границ – значение `FALSE`.

Пример 4. Пример объекта диаграммы

Скрипт загрузки и выражение диаграммы

Обзор

Откройте редактор загрузки данных и добавьте приведенный ниже скрипт загрузки на новую вкладку.

Скрипт загрузки содержит тот же набор данных и сценарий, что в первом примере. Однако в этом примере в приложение загружается неизмененный набор данных. Расчет, который определяет, какие транзакции совершены в том же году до 26 июля 2021 года, создается в качестве меры в объекте диаграммы в приложении.

Скрипт загрузки

```
SET DateFormat='MM/DD/YYYY';

Transactions:
Load
*
Inline
[
id,date,amount
8188,'01/13/2020',37.23
8189,'02/26/2020',17.17
8190,'03/27/2020',88.27
8191,'04/16/2020',57.42
8192,'05/21/2020',53.80
8193,'06/14/2020',82.06
8194,'08/07/2020',40.39
8195,'09/05/2020',87.21
8196,'01/22/2021',95.93
8197,'02/03/2021',45.89
8198,'03/17/2021',36.23
8199,'04/23/2021',25.66
8200,'05/04/2021',82.77
8201,'06/30/2021',69.98
8202,'07/26/2021',76.11
8203,'07/27/2021',25.12
8204,'06/06/2022',46.23
8205,'07/18/2022',84.21
8206,'11/14/2022',96.24
8207,'12/12/2022',67.67
];
```

Результаты

Загрузите данные и откройте лист. Создайте новую таблицу и добавьте это поле как измерение:`date`.

Создайте следующую меру:

```
=inyeartodate(date,'07/26/2021', 0)
```

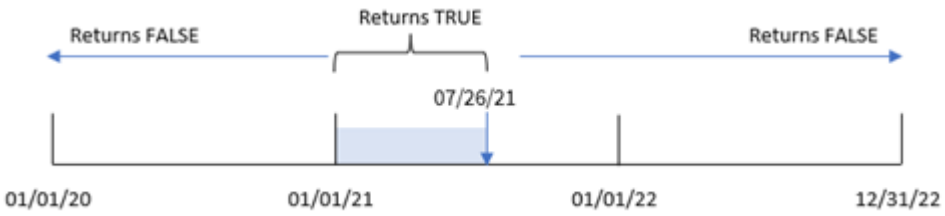
Результирующая таблица

date	=inyeartodate(date,'07/26/2021', 0)
01/13/2020	0
02/26/2020	0

date	=inyeartodate(date,'07/26/2021', 0)
03/27/2020	0
04/16/2020	0
05/21/2020	0
06/14/2020	0
08/07/2020	0
09/05/2020	0
01/22/2021	-1
02/03/2021	-1
03/17/2021	-1
04/23/2021	-1
05/04/2021	-1
06/30/2021	-1
07/26/2021	-1
07/27/2021	0
06/06/2022	0
07/18/2022	0
11/14/2022	0
12/12/2022	0

Мера `in_year_to_date` создается в объекте диаграммы с помощью функции `inyeartodate()`. Первый предоставленный аргумент определяет, какое поле оценивается. Второй аргумент — это жестко закодированная дата «26 июля 2021 года», которая представляет собой `base_date`, определяющий конечную границу сопоставленного сегмента года. `period_no` = 0 — это последний аргумент, означающий, что функция не сравнивает год, предшествующие или следующие за сегментированным годом.

Диаграмма функции inyeartodate, пример с объектом диаграммы



Любая транзакция, совершенная в период с 1 января по 26 июля 2021 года, возвращает булев результат `TRUE`. Даты транзакций до 2021 года и после 26 июля 2021 года возвращают `FALSE`.

Пример 5. Сценарий

Скрипт загрузки и выражение диаграммы

Обзор

Откройте редактор загрузки данных и добавьте приведенный ниже скрипт загрузки на новую вкладку.

Скрипт загрузки содержит следующее:

- Набор данных, загруженный в таблицу, называется `Products`
- Информация об идентификаторе продукта, типе продукта дате изготовления и себестоимости.

Конечный пользователь хотел бы получить объект диаграммы, отображающий по типу продукта стоимость продуктов по типу продукта, произведенных в 2021 году до 26 июля.

Скрипт загрузки

```
Products:
Load
*
Inline
[
product_id,product_type,manufacture_date,cost_price
8188,product A,'01/13/2020',37.23
8189,product B,'02/26/2020',17.17
8190,product B,'03/27/2020',88.27
8191,product C,'04/16/2020',57.42
8192,product D,'05/21/2020',53.80
8193,product D,'08/14/2020',82.06
8194,product C,'10/07/2020',40.39
8195,product B,'12/05/2020',87.21
8196,product A,'01/22/2021',95.93
8197,product B,'02/03/2021',45.89
8198,product C,'03/17/2021',36.23
8199,product C,'04/23/2021',25.66
8200,product B,'05/04/2021',82.77
8201,product D,'06/30/2021',69.98
8202,product D,'07/26/2021',76.11
8203,product D,'12/27/2021',25.12
8204,product C,'06/06/2022',46.23
8205,product C,'07/18/2022',84.21
8206,product A,'11/14/2022',96.24
8207,product B,'12/12/2022',67.67
];
```

Результаты

Загрузите данные и откройте лист. Создайте новую таблицу и добавьте это поле как измерение:`product_type`.

Создайте меру, которая вычисляет сумму для каждого продукта, произведенного в 2021 году до 27 июля:

```
=sum(if(inyeartodate(manufacture_date,makedate(2021,07,26),0),cost_price,0))
```

Задайте параметру меры **Формат чисел** значение **Денежный**.

Результирующая таблица

product_type	=sum(if(inyeartodate(manufacture_date,makedate (2021,07,26),0),cost_price,0))
product A	$95.93
product B	$128.66
product C	$61.89
product D	$146.09

Функция `inyeartodate()` возвращает логическое значение при проверки дат производства каждого продукта. Для любого продукта, изготовленного в 2021 году до 27 июля, функция `inyeartodate()` возвращает булево значение `TRUE` и суммирует `cost_price`.

Продукт D – единственный продукт, который также был произведен после 26 июля 2021 года. Запись с `product_ID` 8203 была изготовлена 27 декабря и стоила $25.12. Таким образом, эта стоимость не была включена в общую сумму продукта D в объекте диаграммы.

lastworkdate

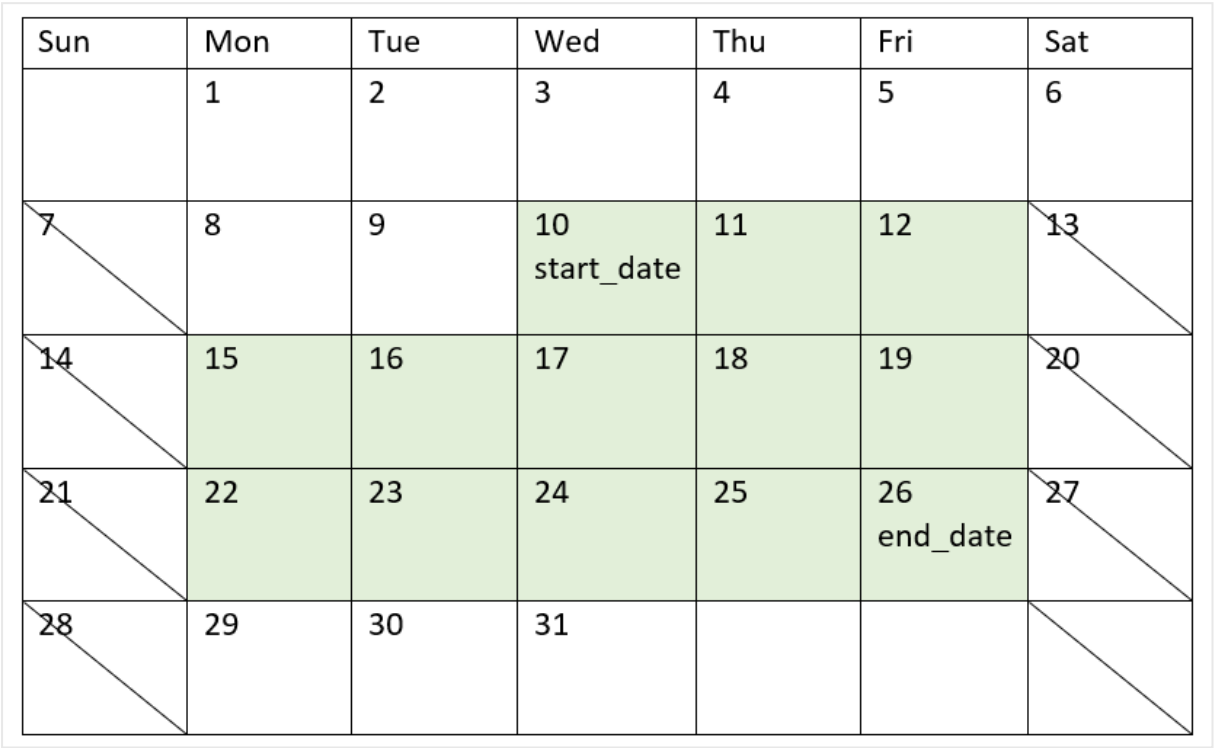
Функция **lastworkdate** возвращает самую раннюю дату окончания для достижения указанного числа рабочих дней **no_of_workdays** (понедельник-пятница) с начальной датой **start_date** и с учетом выходных, которые можно дополнительно задать в поле **holiday**. Поля **start_date** и **holiday** должны быть действительными датами или метками времени.

Синтаксис:

```
lastworkdate(start_date, no_of_workdays {, holiday})
```

Возвращаемые типы данных: целое

Календарь, демонстрирующий использование функции `lastworkdate()`



Sun	Mon	Tue	Wed	Thu	Fri	Sat
	1	2	3	4	5	6
7	8	9	10 start_date	11	12	13
14	15	16	17	18	19	20
21	22	23	24	25	26 end_date	27
28	29	30	31			

Ограничения

Не предусмотрено способа адаптации функции `lastworkdate()` для регионов или сценариев, где используются рабочие недели, которые начинаются не в понедельник и заканчиваются не в пятницу.

Параметр holiday должен быть постоянной строкой. Он не принимает выражений.

Когда это следует использовать

Функция `lastworkdate()` широко используется в составе выражения, кода пользователю требуется вычислить предполагаемую дату окончания проекта или назначения, на основе времени начала проекта и праздников, которые приходятся на этот период.

Региональные настройки

Если не указано иное, в примерах, приведенных в данном разделе, используется следующий формат даты: MM/ДД/ГГГГ. Формат даты указан в операторе `SET DateFormat` скрипта загрузки данных. В вашей системе может быть установлен другой формат даты по умолчанию в зависимости от региональных настроек и других факторов. Можно изменить формат в примерах в соответствии с потребностями. Или можно изменить форматы в скрипте загрузки в соответствии с этими примерами.

Региональные настройки по умолчанию в приложениях основаны на системных региональных настройках компьютера или сервера, где установлено ПО Qlik Sense. Если на сервере Qlik Sense, к которому обращается пользователь, выбран шведский язык, то редактор загрузки данных будет использовать шведские региональные настройки для даты, времени и валюты. Эти параметры регионального формата не связаны с языком, отображаемым в интерфейсе пользователя Qlik Sense. Qlik Sense будет отображаться на языке, который используется в браузере.

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. Праздник обозначен как постоянная строка даты. Можно указать несколько дат праздников, разделенных запятыми. **Пример:** `'12/25/2013', '12/26/2013', '12/31/2013', '01/01/2014'`

Пример 1. Базовый пример

Скрипт загрузки и результаты

Обзор

Откройте редактор загрузки данных и добавьте приведенный ниже скрипт загрузки на новую вкладку.

Скрипт загрузки содержит следующее:

- Набор данных, содержащий идентификаторы и даты начала проектов, а также предполагаемые трудозатраты в днях, необходимые для их выполнения. Набор данных загружается в таблицу под именем `Projects`.
- Предшествующая загрузка, которая содержит функцию `lastworkdate()`, настроенную как поле `end_date` и определяющую запланированное окончания каждого проекта.

Скрипт загрузки

```
SET DateFormat='MM/DD/YYYY';

Projects:
    Load
       *,
       LastWorkDate(start_date,effort) as end_date
       ;
Load
id,
start_date,
effort
```

```
Inline
[
id,start_date,effort
1,01/01/2022,14
2,02/10/2022,17
3,05/17/2022,5
4,06/01/2022,12
5,08/10/2022,26
];
```

Результаты

Загрузите данные и откройте лист. Создайте новую таблицу и добавьте эти поля как измерения:

- `id`
- `start_date`
- `effort`
- `end_date`

Результирующая таблица

id	start_date	effort	end_date
1	01/01/2022	14	01/20/2022
2	02/10/2022	17	03/04/2022
3	05/17/2022	5	05/23/2022
4	06/01/2022	12	06/16/2022
5	08/10/2022	26	09/14/2022

Так как запланированные праздничные дни отсутствуют, функция прибавляет определенное количество рабочих дней (с понедельника по пятницу) в дате начала, чтобы найти самую раннюю возможную дату окончания.

В следующем календаре показаны даты начала и окончания для проекта 3, рабочие дни выделены зеленым.

Календарь с датами начала и окончания проекта 3

Sun	Mon	Tue	Wed	Thu	Fri	Sat
1	2	3	4	5	6	7
8	9	10	11	12	13	14
15	16	17 Start Date	18	19	20	21
22	23 End Date	24	25	26	27	28
29	30	31				

Пример 2. Один праздник

Скрипт загрузки и результаты

Обзор

Откройте редактор загрузки данных и добавьте приведенный ниже скрипт загрузки на новую вкладку.

Скрипт загрузки содержит следующее:

- Набор данных, содержащий идентификаторы и даты начала проектов, а также предполагаемые трудозатраты в днях, необходимые для их выполнения. Набор данных загружается в таблицу под именем `Projects`.
- Предшествующая загрузка, которая содержит функцию `lastworkdate()`, настроенную как поле `end_date` и определяющую запланированное окончания каждого проекта.

Однако запланирован один государственный праздник 18 мая 2022 года. Функция `lastworkdate()` в предыдущей загрузке включает праздник в третьем аргументе, чтобы определить, когда планируется завершение каждого проекта.

Скрипт загрузки

```
SET DateFormat='MM/DD/YYYY';

Projects:
    Load
       *,
       LastWorkDate(start_date,effort, '05/18/2022') as end_date
       ;
Load
id,
start_date,
effort
Inline
[
id,start_date,effort
1,01/01/2022,14
2,02/10/2022,17
3,05/17/2022,5
4,06/01/2022,12
5,08/10/2022,26
];
```

Результаты

Загрузите данные и откройте лист. Создайте новую таблицу и добавьте эти поля как измерения:

- id
- start_date
- effort
- end_date

Результирующая таблица

id	start_date	effort	end_date
1	01/01/2022	14	01/20/2022
2	02/10/2022	17	03/04/2022
3	05/17/2022	5	05/24/2022
4	06/01/2022	12	06/16/2022
5	08/10/2022	26	09/14/2022

Один запланированный праздник передан в качестве третьего аргумента функции `lastworkdate()`. В результате, дата окончания проекта 3 сдвигается на один день вперед, так как праздник выпадает на один из рабочих дней до даты окончания.

В следующем календаре отображаются даты начала и окончания для проекта 3, а также показано, что из-за праздника дата окончания проекта переносится на один день.

Календарь с датами начала и окончания проекта 3 и государственным праздником 18 мая

Sun	Mon	Tue	Wed	Thu	Fri	Sat
1	2	3	4	5	6	7
8	9	10	11	12	13	14
15	16	17 Start Date	18 Holiday	19	20	21
22	23	24 End Date	25	26	27	28
29	30	31				

Пример 3. Несколько праздников

Скрипт загрузки и результаты

Обзор

Откройте редактор загрузки данных и добавьте приведенный ниже скрипт загрузки на новую вкладку.

Скрипт загрузки содержит следующее:

- Набор данных, содержащий идентификаторы и даты начала проектов, а также предполагаемые трудозатраты в днях, необходимые для их выполнения. Набор данных загружается в таблицу под именем `Projects`.
- Предшествующая загрузка, которая содержит функцию `lastworkdate()`, настроенную как поле `end_date` и определяющую запланированное окончание каждого проекта.

Однако запланировано четыре праздничных дня в мае – 19, 20, 21 и 22. Функция `lastworkdate()` в предыдущей загрузке включает все праздничные дни в третьем аргументе, чтобы определить, когда планируется завершение каждого проекта.

Скрипт загрузки

```
SET DateFormat='MM/DD/YYYY';

Projects:
    Load
       *,
       LastWorkDate(start_date,effort, '05/19/2022','05/20/2022','05/21/2022','05/22/2022') as
end_date
       ;
Load
id,
start_date,
effort
Inline
[
id,start_date,effort
1,01/01/2022,14
2,02/10/2022,17
3,05/17/2022,5
4,06/01/2022,12
5,08/10/2022,26
];
```

Результаты

Загрузите данные и откройте лист. Создайте новую таблицу и добавьте эти поля как измерения:

- id
- start_date
- effort
- end_date

Результирующая таблица

id	start_date	effort	end_date
1	01/01/2022	14	01/20/2022
2	02/10/2022	17	03/04/2022
3	05/17/2022	5	05/25/2022
4	06/01/2022	12	06/16/2022
5	08/10/2022	26	09/14/2022

Четыре праздничных дня переданы в виде списка аргументов в функции `lastworkdate()` после даты начала и количества рабочих дней.

В следующем календаре отображаются даты начала и окончания для проекта 3, а также показано, что из-за праздников дата окончания проекта переносится на три дня.

Календарь с датами начала и окончания проекта 3 и праздниками с 19 по 22 мая

Sun	Mon	Tue	Wed	Thu	Fri	Sat
1	2	3	4	5	6	7
8	9	10	11	12	13	14
15	16	17 Start Date	18	19 Holiday	20 Holiday	21 Holiday
22 Holiday	23	24	25 End Date	26	27	28
29	30	31				

Пример 4. Один праздник (диаграмма)

Скрипт загрузки и выражение диаграммы

Обзор

Используется тот же набор данных и сценарий, что в первом примере.

Однако в этом примере в приложение загружается неизмененный набор данных. Поле `end_date` вычисляется как мера в диаграмме.

Скрипт загрузки

```
SET DateFormat='MM/DD/YYYY';

Projects:
Load
id,
start_date,
effort
Inline
[
```

```
id,start_date,effort
1,01/01/2022,14
2,02/10/2022,17
3,05/17/2022,5
4,06/01/2022,12
5,08/10/2022,26
];
```

Результаты

Загрузите данные и откройте лист. Создайте новую таблицу и добавьте эти поля как измерения:

- `id`
- `start_date`
- `effort`

Чтобы вычислить end_date, создайте следующую меру:

- `=LastWorkDate(start_date,effort,'05/18/2022')`

Результирующая таблица

id	start_date	effort	=LastWorkDate(start_date,effort,'05/18/2022')
1	01/01/2022	14	01/20/2022
2	02/10/2022	17	03/04/2022
3	05/17/2022	5	05/23/2022
4	06/01/2022	12	06/16/2022
5	08/10/2022	26	09/14/2022

Один запланированный праздник вводится как мера в диаграмме. В результате, дата окончания проекта 3 сдвигается на один день вперед, так как праздник выпадает на один из рабочих дней до даты окончания.

В следующем календаре отображаются даты начала и окончания для проекта 3, а также показано, что из-за праздника дата окончания проекта переносится на один день.

Календарь с датами начала и окончания проекта 3 и государственным праздником 18 мая

Sun	Mon	Tue	Wed	Thu	Fri	Sat
1	2	3	4	5	6	7
8	9	10	11	12	13	14
15	16	17 Start Date	18 Holiday	19	20	21
22	23	24 End Date	25	26	27	28
29	30	31				

localtime

Эта функция возвращает метку текущего времени 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

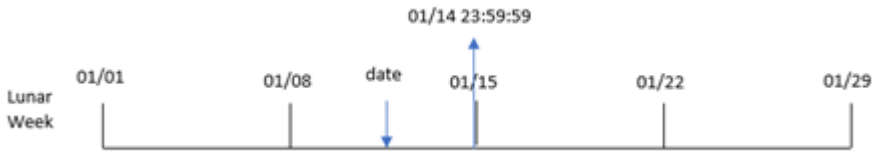
Эта функция возвращает значение, соответствующее метке времени последней миллисекунды последнего дня лунной недели, содержащей значение, указанное в поле **date**. При определении лунных недель в Qlik Sense первым днем первой недели считается 1 января. Все недели, кроме последней будут содержать ровно 7 дней.

Синтаксис:

```
LunarweekEnd(date[, period_no[, first_week_day]])
```

Возвращаемые типы данных: двойное значение

Диаграмма с примером функции `lunarweekend()`



Функция `lunarweekend()` определяет, на какую лунную неделю выпадает `date`. Затем она возвращает метку времени в формате даты для последней миллисекунды этой недели.

Аргументы

Аргумент	Описание
date	Дата или метка времени 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.

Когда это следует использовать

Функция `lunarweekend()` широко используется в составе выражения, когда пользователю требуется учитывать в расчетах часть недели, которая еще не прошла. В отличие от функции `weekend()`, заключительная лунная неделя каждого календарного года заканчивается 31 декабря. Например, функцию `lunarweekend()` можно использовать для расчета процента, еще не начисленного в течение недели.

Примеры функции

Пример	Результат
`lunarweekend('01/12/2013')`	Возвращает 01/14/2013 23:59:59.
`lunarweekend('01/12/2013', -1)`	Возвращает 01/07/2013 23:59:59.
`lunarweekend('01/12/2013', 0, 1)`	Возвращает 01/15/2013 23:59:59.

Региональные настройки

Если не указано иное, в примерах, приведенных в данном разделе, используется следующий формат даты: ММ/ДД/ГГГГ. Формат даты указан в операторе `SET DateFormat` скрипта загрузки данных. В вашей системе может быть установлен другой формат даты по умолчанию в

зависимости от региональных настроек и других факторов. Можно изменить формат в примерах в соответствии с потребностями. Или можно изменить форматы в скрипте загрузки в соответствии с этими примерами.

Региональные настройки по умолчанию в приложениях основаны на системных региональных настройках компьютера или сервера, где установлено ПО Qlik Sense. Если на сервере Qlik Sense, к которому обращается пользователь, выбран шведский язык, то редактор загрузки данных будет использовать шведские региональные настройки для даты, времени и валюты. Эти параметры регионального формата не связаны с языком, отображаемым в интерфейсе пользователя Qlik Sense. Qlik Sense будет отображаться на языке, который используется в браузере.

Пример 1. Без дополнительных аргументов

Скрипт загрузки и результаты

Обзор

Откройте редактор загрузки данных и добавьте приведенный ниже скрипт загрузки на новую вкладку.

Скрипт загрузки содержит следующее:

- Набор данных, содержащий набор транзакций за 2022 год, загружается в таблицу под именем `Transactions`.
- Поле даты было предоставлено в формате системной переменной `DateFormat` (MM/DD/YYYY).
- Создание поля `end_of_week`, возвращающего метку времени окончания лунной недели, в течение которой совершены транзакции.

Скрипт загрузки

```
SET DateFormat='MM/DD/YYYY';

Transactions:
    Load
        *,
        lunarweekend(date) as end_of_week,
        timestamp(lunarweekend(date)) as end_of_week_timestamp
        ;
Load
*
Inline
[
id,date,amount
8188,1/7/2022,17.17
8189,1/19/2022,37.23
8190,2/28/2022,88.27
8191,2/5/2022,57.42
8192,3/16/2022,53.80
8193,4/1/2022,82.06
8194,5/7/2022,40.39
```

```
8195,5/16/2022,87.21
8196,6/15/2022,95.93
8197,6/26/2022,45.89
8198,7/9/2022,36.23
8199,7/22/2022,25.66
8200,7/23/2022,82.77
8201,7/27/2022,69.98
8202,8/2/2022,76.11
8203,8/8/2022,25.12
8204,8/19/2022,46.23
8205,9/26/2022,84.21
8206,10/14/2022,96.24
8207,10/29/2022,67.67
];
```

Результаты

Загрузите данные и откройте лист. Создайте новую таблицу и добавьте эти поля как измерения:

- `date`
- `end_of_week`
- `end_of_week_timestamp`

Результирующая таблица

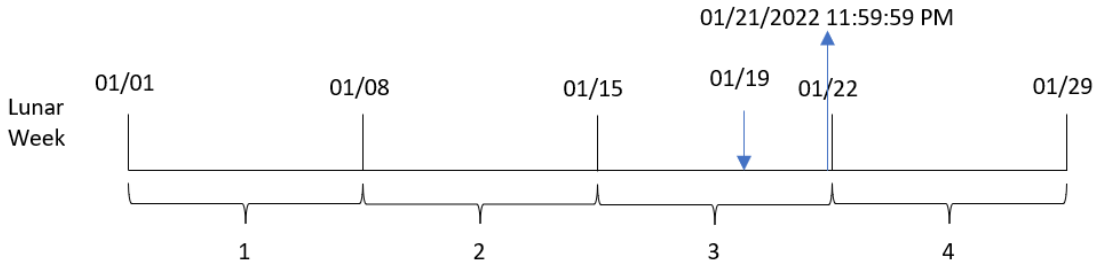
date	end_of_week	end_of_week_timestamp
1/7/2022	01/07/2022	1/7/2022 11:59:59 PM
1/19/2022	01/21/2022	1/21/2022 11:59:59 PM
2/5/2022	02/11/2022	2/11/2022 11:59:59 PM
2/28/2022	03/04/2022	3/4/2022 11:59:59 PM
3/16/2022	03/18/2022	3/18/2022 11:59:59 PM
4/1/2022	04/01/2022	4/1/2022 11:59:59 PM
5/7/2022	05/13/2022	5/13/2022 11:59:59 PM
5/16/2022	05/20/2022	5/20/2022 11:59:59 PM
6/15/2022	06/17/2022	6/17/2022 11:59:59 PM
6/26/2022	07/01/2022	7/1/2022 11:59:59 PM
7/9/2022	07/15/2022	7/15/2022 11:59:59 PM
7/22/2022	07/22/2022	7/22/2022 11:59:59 PM
7/23/2022	07/29/2022	7/29/2022 11:59:59 PM
7/27/2022	07/29/2022	7/29/2022 11:59:59 PM
8/2/2022	08/05/2022	8/5/2022 11:59:59 PM

date	end_of_week	end_of_week_timestamp
8/8/2022	08/12/2022	8/12/2022 11:59:59 PM
8/19/2022	08/19/2022	8/19/2022 11:59:59 PM
9/26/2022	09/30/2022	9/30/2022 11:59:59 PM
10/14/2022	10/14/2022	10/14/2022 11:59:59 PM
10/29/2022	11/04/2022	11/4/2022 11:59:59 PM

Поле `end_of_week` создано предшествующим оператором load с использованием функции `lunarweekend()`, где в качестве аргумента функции передано поле `date`.

Функция `lunarweekend()` определяет, к какой лунной неделе относится значение даты, и возвращает метку времени для последней миллисекунды этой недели.

Диаграмма функции `lunarweekend()`, пример без дополнительных аргументов



Транзакция 8189 совершена 19 января. Функция `lunarweekend()` определяет, что лунная неделя начинается 15 января. Поэтому значение `end_of_week` для этой транзакции возвращает последнюю миллисекунду лунной недели, то есть 21 января в 23:59:59 (11:59:59 PM).

Пример 2. Скрипт period_no

Скрипт загрузки и результаты

Обзор

Откройте редактор загрузки данных и добавьте приведенный ниже скрипт загрузки на новую вкладку.

Скрипт загрузки содержит следующее:

- Используется тот же набор данных и сценарий, что в первом примере.
- Создание поля `previous_lunar_week_end`, возвращающего метку времени окончания лунной недели, которая предшествует совершению транзакции.

Скрипт загрузки

```
SET DateFormat='MM/DD/YYYY';

Transactions:
    Load
        *,
        lunarweekend(date,-1) as previous_lunar_week_end,
        timestamp(lunarweekend(date,-1)) as previous_lunar_week_end_timestamp
        ;
Load
*
Inline
[
id,date,amount
8188,1/7/2022,17.17
8189,1/19/2022,37.23
8190,2/28/2022,88.27
8191,2/5/2022,57.42
8192,3/16/2022,53.80
8193,4/1/2022,82.06
8194,5/7/2022,40.39
8195,5/16/2022,87.21
8196,6/15/2022,95.93
8197,6/26/2022,45.89
8198,7/9/2022,36.23
8199,7/22/2022,25.66
8200,7/23/2022,82.77
8201,7/27/2022,69.98
8202,8/2/2022,76.11
8203,8/8/2022,25.12
8204,8/19/2022,46.23
8205,9/26/2022,84.21
8206,10/14/2022,96.24
8207,10/29/2022,67.67
];
```

Результаты

Загрузите данные и откройте лист. Создайте новую таблицу и добавьте эти поля как измерения:

- date
- previous_lunar_week_end
- previous_lunar_week_end_timestamp

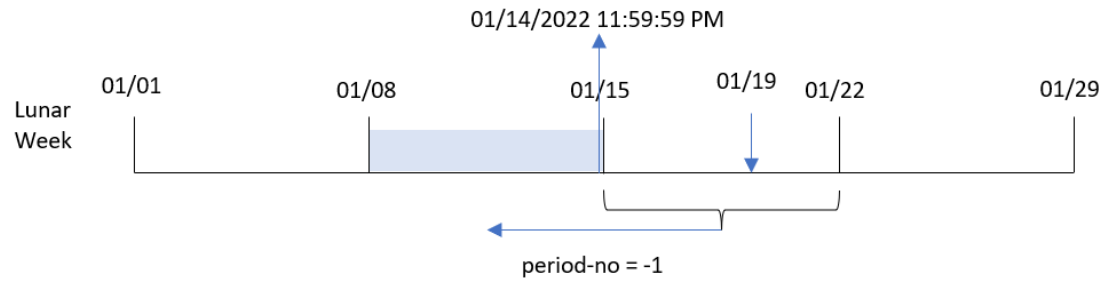
Результирующая таблица

date	previous_lunar_week_end	previous_lunar_week_end_timestamp
1/7/2022	12/31/2021	12/31/2021 11:59:59 PM
1/19/2022	01/14/2022	1/14/2022 11:59:59 PM

date	previous_lunar_week_end	previous_lunar_week_end_timestamp
2/5/2022	02/04/2022	2/4/2022 11:59:59 PM
2/28/2022	02/25/2022	2/25/2022 11:59:59 PM
3/16/2022	03/11/2022	3/18/2022 11:59:59 PM
4/1/2022	03/25/2022	3/25/2022 11:59:59 PM
5/7/2022	05/06/2022	5/6/2022 11:59:59 PM
5/16/2022	05/13/2022	5/13/2022 11:59:59 PM
6/15/2022	06/10/2022	6/10/2022 11:59:59 PM
6/26/2022	06/24/2022	6/24/2022 11:59:59 PM
7/9/2022	07/08/2022	7/8/2022 11:59:59 PM
7/22/2022	07/15/2022	7/15/2022 11:59:59 PM
7/23/2022	07/22/2022	7/22/2022 11:59:59 PM
7/27/2022	07/22/2022	7/22/2022 11:59:59 PM
8/2/2022	07/29/2022	7/29/2022 11:59:59 PM
8/8/2022	08/05/2022	8/5/2022 11:59:59 PM
8/19/2022	08/12/2022	8/12/2022 11:59:59 PM
9/26/2022	09/23/2022	9/23/2022 11:59:59 PM
10/14/2022	10/07/2022	10/7/2022 11:59:59 PM
10/29/2022	10/28/2022	10/28/2022 11:59:59 PM

В этом случае, так как в качестве аргумента смещения в функции `lunarweekend()` использовалось `period_no` = -1, функция сначала определяет лунную неделю, в течение которой совершены транзакции. Затем она возвращается на неделю назад и определяет последнюю миллисекунду предыдущей лунной недели.

Диаграмма функции `lunarweekend()`, пример с аргументом period_no



Транзакция 8189 совершена 19 января. Функция `lunarweekend()` определяет, что лунная неделя начинается 15 января. Таким образом, предыдущая лунная неделя началась 8 января и закончилась 14 января в 23:59:59 (11:59:59 PM). Именно это значение возвращается для поля `previous_lunar_week_end`.

Пример 3. Аргумент first_week_day

Скрипт загрузки и результаты

Обзор

Откройте редактор загрузки данных и добавьте приведенный ниже скрипт загрузки на новую вкладку.

Скрипт загрузки содержит тот же набор данных и сценарий, что в первом примере. В этом примере лунные недели начинаются с 5 января.

Скрипт загрузки

```
SET DateFormat='MM/DD/YYYY';

Transactions:
    Load
        *,
        lunarweekend(date,0,4) as end_of_week,
timestamp(lunarweekend(date,0,4)) as end_of_week_timestamp
;
Load
*
Inline
[
id,date,amount
8188,1/7/2022,17.17
8189,1/19/2022,37.23
8190,2/28/2022,88.27
8191,2/5/2022,57.42
8192,3/16/2022,53.80
8193,4/1/2022,82.06
8194,5/7/2022,40.39
8195,5/16/2022,87.21
8196,6/15/2022,95.93
8197,6/26/2022,45.89
8198,7/9/2022,36.23
8199,7/22/2022,25.66
8200,7/23/2022,82.77
8201,7/27/2022,69.98
8202,8/2/2022,76.11
8203,8/8/2022,25.12
8204,8/19/2022,46.23
8205,9/26/2022,84.21
8206,10/14/2022,96.24
```

```
8207,10/29/2022,67.67
];
```

Результаты

Загрузите данные и откройте лист. Создайте новую таблицу и добавьте эти поля как измерения:

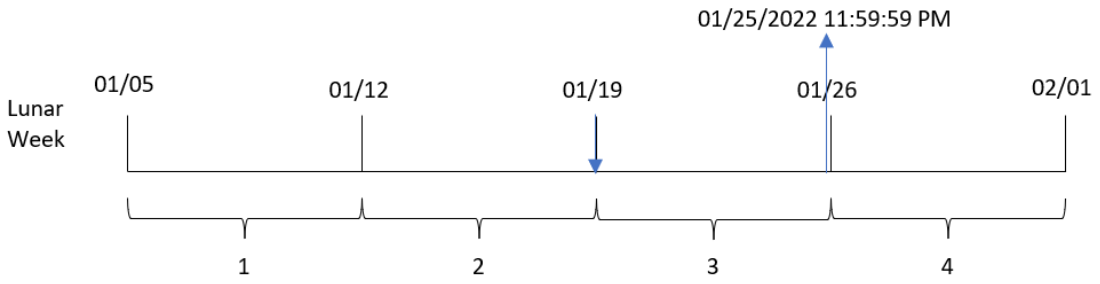
- `date`
- `end_of_week`
- `end_of_week_timestamp`

Результирующая таблица

date	end_of_week	end_of_week_timestamp
1/7/2022	01/11/2022	1/11/2022 11:59:59 PM
1/19/2022	01/25/2022	1/25/2022 11:59:59 PM
2/5/2022	02/08/2022	2/8/2022 11:59:59 PM
2/28/2022	03/01/2022	3/1/2022 11:59:59 PM
3/16/2022	03/22/2022	3/22/2022 11:59:59 PM
4/1/2022	04/05/2022	4/5/2022 11:59:59 PM
5/7/2022	05/10/2022	5/10/2022 11:59:59 PM
5/16/2022	05/17/2022	5/17/2022 11:59:59 PM
6/15/2022	06/21/2022	6/21/2022 11:59:59 PM
6/26/2022	06/28/2022	6/28/2022 11:59:59 PM
7/9/2022	07/12/2022	7/12/2022 11:59:59 PM
7/22/2022	07/26/2022	7/26/2022 11:59:59 PM
7/23/2022	07/26/2022	7/26/2022 11:59:59 PM
7/27/2022	08/02/2022	8/2/2022 11:59:59 PM
8/2/2022	08/02/2022	8/2/2022 11:59:59 PM
8/8/2022	08/09/2022	8/9/2022 11:59:59 PM
8/19/2022	08/23/2022	8/23/2022 11:59:59 PM
9/26/2022	09/27/2022	9/27/2022 11:59:59 PM
10/14/2022	10/18/2022	10/18/2022 11:59:59 PM
10/29/2022	11/01/2022	11/1/2022 11:59:59 PM

В этом примере используется аргумент `first_week_date` = 4 в функции `lunarweekend()`, потому начало года переносится с 1 на 5 января.

Диаграмма функции lunarweekend(), пример с аргументом first_week_day



Транзакция 8189 совершена 19 января. Так как лунные недели начинаются 5 января, функция `lunarweekend()` определяет, что лунная неделя, к которой относится 19 января, также начинается в этот день. Таким образом, эта лунная неделя заканчивается 25 января в 23:59:59 (11:59:59 PM). Именно это значение возвращается для поля `end_of_week`.

Пример 4. Пример объекта диаграммы

Скрипт загрузки и выражение диаграммы

Обзор

Откройте редактор загрузки данных и добавьте приведенный ниже скрипт загрузки на новую вкладку.

Скрипт загрузки содержит тот же набор данных и сценарий, что в первом примере.

Однако в этом примере в приложение загружается неизмененный набор данных. Расчет, который возвращает метку времени окончания лунной недели, в течение которой совершены транзакции, создается как мера в объекте диаграммы в приложении.

Скрипт загрузки

```
Transactions:
Load
*
Inline
[
id,date,amount
8188,1/7/2022,17.17
8189,1/19/2022,37.23
8190,2/28/2022,88.27
8191,2/5/2022,57.42
8192,3/16/2022,53.80
8193,4/1/2022,82.06
8194,5/7/2022,40.39
8195,5/16/2022,87.21
8196,6/15/2022,95.93
8197,6/26/2022,45.89
8198,7/9/2022,36.23
```

```
8199,7/22/2022,25.66
8200,7/23/2022,82.77
8201,7/27/2022,69.98
8202,8/2/2022,76.11
8203,8/8/2022,25.12
8204,8/19/2022,46.23
8205,9/26/2022,84.21
8206,10/14/2022,96.24
8207,10/29/2022,67.67
];
```

Результаты

Загрузите данные и откройте лист. Создайте новую таблицу и добавьте это поле как измерение: `date`.

Добавьте следующие меры:

```
=lunarweekend(date)
```

```
=timestamp(lunarweekend(date))
```

Результирующая таблица

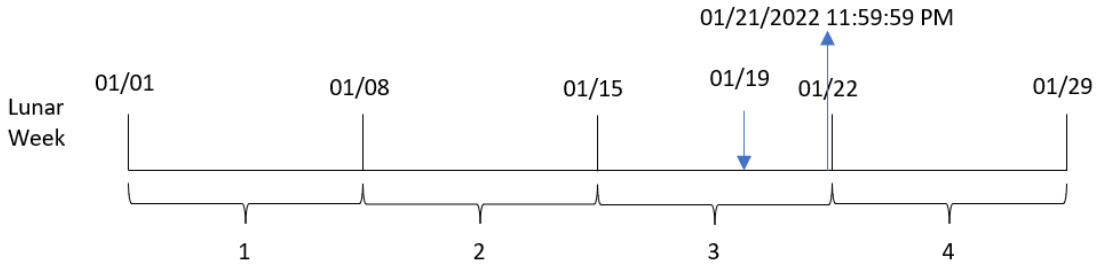
date	=lunarweekend(date)	=timestamp(lunarweekend(date))
1/7/2022	01/07/2022	1/7/2022 11:59:59 PM
1/19/2022	01/21/2022	1/21/2022 11:59:59 PM
2/5/2022	02/11/2022	2/11/2022 11:59:59 PM
2/28/2022	03/04/2022	3/4/2022 11:59:59 PM
3/16/2022	03/18/2022	3/18/2022 11:59:59 PM
4/1/2022	04/01/2022	4/1/2022 11:59:59 PM
5/7/2022	05/13/2022	5/13/2022 11:59:59 PM
5/16/2022	05/20/2022	5/20/2022 11:59:59 PM
6/15/2022	06/17/2022	6/17/2022 11:59:59 PM
6/26/2022	07/01/2022	7/1/2022 11:59:59 PM
7/9/2022	07/15/2022	7/15/2022 11:59:59 PM
7/22/2022	07/22/2022	7/22/2022 11:59:59 PM
7/23/2022	07/29/2022	7/29/2022 11:59:59 PM
7/27/2022	07/29/2022	7/29/2022 11:59:59 PM
8/2/2022	08/05/2022	8/5/2022 11:59:59 PM
8/8/2022	08/12/2022	8/12/2022 11:59:59 PM
8/19/2022	08/19/2022	8/19/2022 11:59:59 PM

date	=lunarweekend(date)	=timestamp(lunarweekend(date))
9/26/2022	09/30/2022	9/30/2022 11:59:59 PM
10/14/2022	10/14/2022	10/14/2022 11:59:59 PM
10/29/2022	11/04/2022	11/4/2022 11:59:59 PM

Мера `end_of_week` создана в объекте диаграммы с использованием функции `lunarweekend()`, где в качестве аргумента функции передано поле `date`.

Функция `lunarweekend()` определяет, к какой лунной неделе относится значение даты, и возвращает метку времени для последней миллисекунды этой недели.

Диаграмма функции `lunarweekend()`, пример с объектом диаграммы



Транзакция 8189 совершена 19 января. Функция `lunarweekend()` определяет, что лунная неделя начинается 15 января. Поэтому значение `end_of_week` для этой транзакции возвращает последнюю миллисекунду лунной недели, то есть 21 января в 23:59:59 (11:59:59 PM).

Пример 5. Сценарий

Скрипт загрузки и выражение диаграммы

Обзор

Откройте редактор загрузки данных и добавьте приведенный ниже скрипт загрузки на новую вкладку.

Скрипт загрузки содержит следующее:

- Набор данных, загруженный в таблицу, называется `Employee_Expenses`
- Идентификаторы сотрудников, имена сотрудников и средние ежедневные заявки на возмещение расходов каждого сотрудника.

Конечному пользователю требуется получить объект диаграммы, отображающий по идентификатору и имени сотрудника расчетные заявки на возмещение расходов, которые еще предстоят в течение оставшейся лунной недели.

Скрипт загрузки

```
Employee_Expenses:
Load
*
Inline
[
employee_id,employee_name,avg_daily_claim
182,Mark, $15
183,Deryck, $12.5
184,Dexter, $12.5
185,Sydney,$27
186,Agatha,$18
];
```

Результаты

Выполните следующие действия.

1. Загрузите данные и откройте лист. Создайте новую таблицу.
2. Добавьте следующие поля как измерения:
   - `employee_id`
   - `employee_name`
3. Затем, чтобы рассчитать накопленный процент, создайте следующую меру:
   `=(lunarweekend(today(1))-today(1))*avg_daily_claim`
4. Задайте параметру меры **Формат чисел** значение **Денежный**.

Результирующая таблица

employee_id	employee_name	=(lunarweekend(today(1))-today(1))*avg_daily_claim
182	Mark	$75.00
183	Deryck	$62.50
184	Dexter	$62.50
185	Sydney	$135.00
186	Agatha	$90.00

Используя сегодняшнюю дату в качестве единственного аргумента, функция `lunarkweekend()` возвращает дату окончания текущей лунной недели. Затем, вычитая сегодняшнюю дату из даты окончания лунной недели, выражение возвращает количество дней, оставшихся в этой неделе.

Затем это значение умножается на среднюю ежедневную заявку на возмещение расходов каждого сотрудника для расчета оценочной суммы заявок, которые каждый сотрудник должен подать до конца лунной недели.

lunarweekname

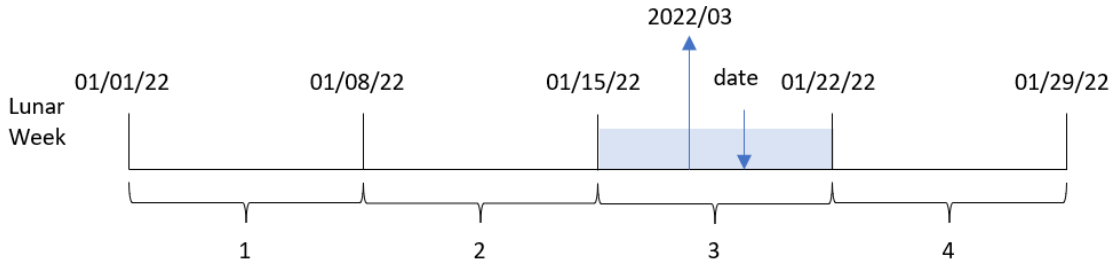
Эта функция возвращает значение года и номер лунной недели, соответствующие метке времени первой миллисекунды первого дня лунной недели, содержащего значение, указанное в поле **date**. При определении лунных недель в Qlik Sense первым днем первой недели считается 1 января. Все недели, кроме последней, будут содержать ровно 7 дней.

Синтаксис:

```
LunarWeekName(date [, period_no[, first_week_day]])
```

Возвращаемые типы данных: двойное значение

Диаграмма с примером функции `lunarweekname()`



Функция `lunarweekname()` определяет, к какой лунной неделе относится дата, начиная отсчет недель с 1 января. Затем она возвращает значение, которое включает `year/weekcount`.

Аргументы

Аргумент	Описание
date	Дата или метка времени 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.

Когда это следует использовать

Функция `lunarweekname()` полезна, когда требуется сравнить агрегации по лунным неделям. Например, с помощью функции можно определить общий объем продаж продуктов по лунной неделе. Лунные недели полезны, когда требуется обеспечить, чтобы все значения в первой неделе года содержали только значения начиная с 1 января.

Эти измерения можно создать в скрипте загрузки с помощью функции создания поля в таблице основного календаря. Эту функцию также можно использовать непосредственно в диаграмме в качестве вычисляемого измерения.

Примеры функции

Пример	Результат
`lunarweekname('01/12/2013')`	Возвращает 2006/02.
`lunarweekname('01/12/2013', -1)`	Возвращает 2006/01.
`lunarweekname('01/12/2013', 0, 1)`	Возвращает 2006/02.

Региональные настройки

Если не указано иное, в примерах, приведенных в данном разделе, используется следующий формат даты: ММ/ДД/ГГГГ. Формат даты указан в операторе `SET DateFormat` скрипта загрузки данных. В вашей системе может быть установлен другой формат даты по умолчанию в зависимости от региональных настроек и других факторов. Можно изменить формат в примерах в соответствии с потребностями. Или можно изменить форматы в скрипте загрузки в соответствии с этими примерами.

Региональные настройки по умолчанию в приложениях основаны на системных региональных настройках компьютера или сервера, где установлено ПО Qlik Sense. Если на сервере Qlik Sense, к которому обращается пользователь, выбран шведский язык, то редактор загрузки данных будет использовать шведские региональные настройки для даты, времени и валюты. Эти параметры регионального формата не связаны с языком, отображаемым в интерфейсе пользователя Qlik Sense. Qlik Sense будет отображаться на языке, который используется в браузере.

Пример 1. Дата без дополнительных аргументов

Скрипт загрузки и результаты

Обзор

Откройте редактор загрузки данных и добавьте приведенный ниже скрипт загрузки на новую вкладку.

Скрипт загрузки содержит следующее:

- Набор данных, содержащий набор транзакций за 2022 год, загружается в таблицу под именем `Transactions`.
- Поле даты было предоставлено в формате системной переменной `DateFormat` (MM/DD/YYYY).
- Создание поля `lunar_week_name`, возвращающего год и номер лунной недели, в течение которой совершены транзакции.

Скрипт загрузки

```
SET DateFormat='MM/DD/YYYY';

Transactions:
    Load
        *,
        lunarweekname(date) as lunar_week_name
        ;
Load
*
Inline
[
id,date,amount
8188,1/7/2022,17.17
8189,1/19/2022,37.23
8190,2/28/2022,88.27
8191,2/5/2022,57.42
8192,3/16/2022,53.80
8193,4/1/2022,82.06
8194,5/7/2022,40.39
8195,5/16/2022,87.21
8196,6/15/2022,95.93
8197,6/26/2022,45.89
8198,7/9/2022,36.23
8199,7/22/2022,25.66
8200,7/23/2022,82.77
8201,7/27/2022,69.98
8202,8/2/2022,76.11
8203,8/8/2022,25.12
8204,8/19/2022,46.23
8205,9/26/2022,84.21
8206,10/14/2022,96.24
8207,10/29/2022,67.67
];
```

Результаты

Загрузите данные и откройте лист. Создайте новую таблицу и добавьте эти поля как измерения:

- `date`
- `lunar_week_name`

Результирующая таблица

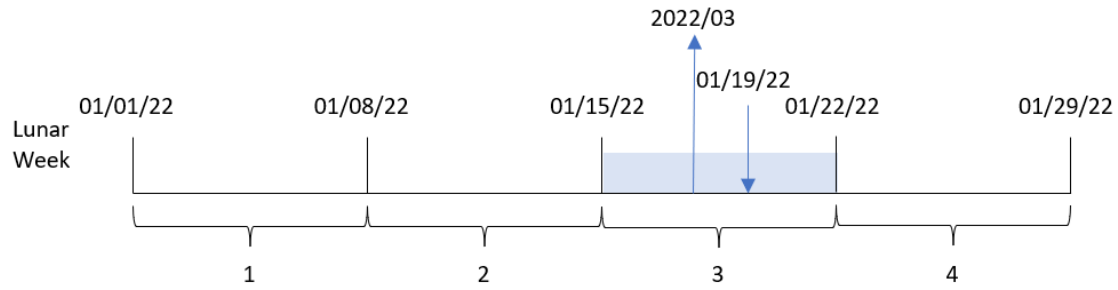
date	lunar_week_name
1/7/2022	2022/01
1/19/2022	2022/03
2/5/2022	2022/06
2/28/2022	2022/09

date	lunar_week_name
3/16/2022	2022/11
4/1/2022	2022/13
5/7/2022	2022/19
5/16/2022	2022/20
6/15/2022	2022/24
6/26/2022	2022/26
7/9/2022	2022/28
7/22/2022	2022/29
7/23/2022	2022/30
7/27/2022	2022/30
8/2/2022	2022/31
8/8/2022	2022/32
8/19/2022	2022/33
9/26/2022	2022/39
10/14/2022	2022/41
10/29/2022	2022/44

Поле `lunar_week_name` создано предшествующим оператором load с использованием функции `lunarweekname()`, где в качестве аргумента передано поле `date`.

Функция `lunarweekname()` определяет, к какой лунной неделе относится значение даты, и возвращает год и номер луной недели для этой даты.

Диаграмма функции `lunarweekname()`, пример без дополнительных аргументов



Транзакция 8189 совершена 19 января. Функция `lunarweekname()` определяет, что эта дата попадает в лунную неделю, которая начинается 15 января; это третья лунная неделя года. Поэтому для этой транзакции возвращается значение `lunar_week_name` = 2022/03.

Пример 2. Дата с аргументом period_no

Скрипт загрузки и результаты

Обзор

Откройте редактор загрузки данных и добавьте приведенный ниже скрипт загрузки на новую вкладку.

Скрипт загрузки содержит следующее:

- Используется тот же набор данных и сценарий, что в первом примере.
- Создание поля `previous_lunar_week_name`, возвращающего год и номер лунной недели, которая предшествует совершению транзакций.

Скрипт загрузки

```
SET DateFormat='MM/DD/YYYY';

Transactions:
    Load
        *,
        lunarweekname(date,-1) as previous_lunar_week_name
        ;
Load
*
Inline
[
id,date,amount
8188,1/7/2022,17.17
8189,1/19/2022,37.23
8190,2/28/2022,88.27
8191,2/5/2022,57.42
8192,3/16/2022,53.80
8193,4/1/2022,82.06
8194,5/7/2022,40.39
8195,5/16/2022,87.21
8196,6/15/2022,95.93
8197,6/26/2022,45.89
8198,7/9/2022,36.23
8199,7/22/2022,25.66
8200,7/23/2022,82.77
8201,7/27/2022,69.98
8202,8/2/2022,76.11
8203,8/8/2022,25.12
8204,8/19/2022,46.23
8205,9/26/2022,84.21
8206,10/14/2022,96.24
8207,10/29/2022,67.67
];
```

Результаты

Загрузите данные и откройте лист. Создайте новую таблицу и добавьте эти поля как измерения:

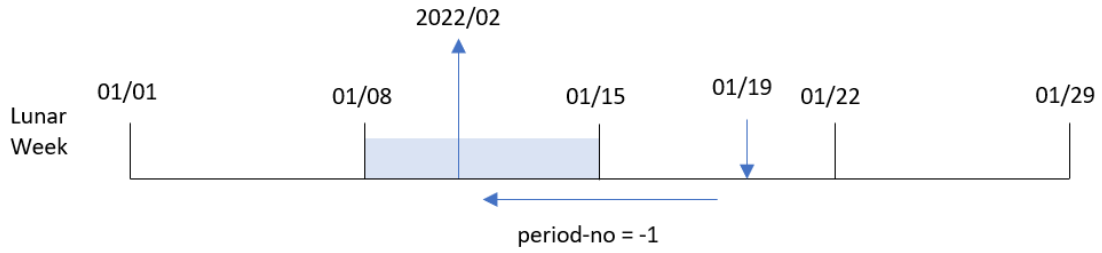
- `date`
- `previous_lunar_week_name`

Результирующая таблица

date	previous_lunar_week_name
1/7/2022	2021/52
1/19/2022	2022/02
2/5/2022	2022/05
2/28/2022	2022/08
3/16/2022	2022/10
4/1/2022	2022/12
5/7/2022	2022/18
5/16/2022	2022/19
6/15/2022	2022/23
6/26/2022	2022/25
7/9/2022	2022/27
7/22/2022	2022/28
7/23/2022	2022/29
7/27/2022	2022/29
8/2/2022	2022/30
8/8/2022	2022/31
8/19/2022	2022/32
9/26/2022	2022/38
10/14/2022	2022/40
10/29/2022	2022/43

В этом случае, так как в качестве аргумента смещения в функции `lunarweekname()` использовалось `period_no` = -1, функция сначала определяет лунную неделю, в течение которой совершены транзакции. Затем она возвращает год и номер предыдущей недели.

Диаграмма функции `lunarweekname()`, пример с аргументом period_no



Транзакция 8189 совершена 19 января. Функция `lunarweekname()` определяет, что эта транзакция совершена на третьей неделе года, поэтому возвращается год и номер предыдущей недели, 2022/02, для поля `previous_lunar_week_name`.

Пример 3. Дата с аргументом first_week_day

Скрипт загрузки и результаты

Обзор

Откройте редактор загрузки данных и добавьте приведенный ниже скрипт загрузки на новую вкладку.

Скрипт загрузки содержит тот же набор данных и сценарий, что в первом примере. В этом примере лунные недели начинаются с 5 января.

Скрипт загрузки

```
SET DateFormat='MM/DD/YYYY';

Transactions:
    Load
        *,
        lunarweekname(date,0,4) as lunar_week_name
        ;
Load
*
Inline
[
id,date,amount
8188,1/7/2022,17.17
8189,1/19/2022,37.23
8190,2/28/2022,88.27
8191,2/5/2022,57.42
8192,3/16/2022,53.80
8193,4/1/2022,82.06
8194,5/7/2022,40.39
8195,5/16/2022,87.21
8196,6/15/2022,95.93
8197,6/26/2022,45.89
```

```
8198,7/9/2022,36.23
8199,7/22/2022,25.66
8200,7/23/2022,82.77
8201,7/27/2022,69.98
8202,8/2/2022,76.11
8203,8/8/2022,25.12
8204,8/19/2022,46.23
8205,9/26/2022,84.21
8206,10/14/2022,96.24
8207,10/29/2022,67.67
];
```

Результаты

Загрузите данные и откройте лист. Создайте новую таблицу и добавьте эти поля как измерения:

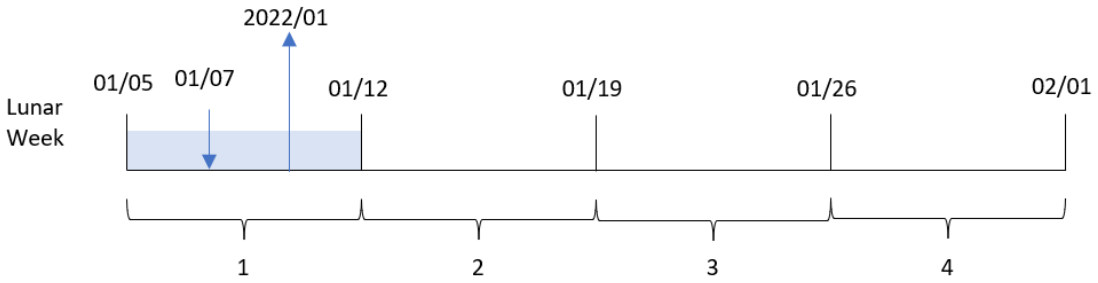
- `date`
- `lunar_week_name`

Результирующая таблица

date	lunar_week_name
1/7/2022	2022/01
1/19/2022	2022/03
2/5/2022	2022/05
2/28/2022	2022/08
3/16/2022	2022/11
4/1/2022	2022/13
5/7/2022	2022/18
5/16/2022	2022/19
6/15/2022	2022/24
6/26/2022	2022/25
7/9/2022	2022/27
7/22/2022	2022/29
7/23/2022	2022/29
7/27/2022	2022/30
8/2/2022	2022/30
8/8/2022	2022/31
8/19/2022	2022/33
9/26/2022	2022/38

date	lunar_week_name
10/14/2022	2022/41
10/29/2022	2022/43

Диаграмма функции `lunarweekname()`, пример с аргументом first_week_day



В этом примере используется аргумент `first_week_date` = 4 в функции `lunarweekname()`, потому начало лунных недель переносится с 1 на 5 января.

Транзакция 8188 совершена 7 января. Так как лунные недели начинаются 5 января, функция `lunarweekname()` определяет, что 7 января относится к первой лунной неделе года. Поэтому для этой транзакции возвращается значение `lunar_week_name` = 2022/01.

Пример 4. Пример объекта диаграммы

Скрипт загрузки и выражение диаграммы

Обзор

Откройте редактор загрузки данных и добавьте приведенный ниже скрипт загрузки на новую вкладку.

Скрипт загрузки содержит тот же набор данных и сценарий, что в первом примере.

Однако в этом примере в приложение загружается неизмененный набор данных. Расчет, который возвращает номер лунной недели и год, когда были совершены транзакции, создается как мера в объекте диаграммы в приложении.

Скрипт загрузки

```
Transactions:
Load
*
Inline
[
id,date,amount
8188,1/7/2022,17.17
8189,1/19/2022,37.23
8190,2/28/2022,88.27
```

```
8191,2/5/2022,57.42
8192,3/16/2022,53.80
8193,4/1/2022,82.06
8194,5/7/2022,40.39
8195,5/16/2022,87.21
8196,6/15/2022,95.93
8197,6/26/2022,45.89
8198,7/9/2022,36.23
8199,7/22/2022,25.66
8200,7/23/2022,82.77
8201,7/27/2022,69.98
8202,8/2/2022,76.11
8203,8/8/2022,25.12
8204,8/19/2022,46.23
8205,9/26/2022,84.21
8206,10/14/2022,96.24
8207,10/29/2022,67.67
];
```

Результаты

Загрузите данные и откройте лист. Создайте новую таблицу и добавьте это поле как измерение: `date`.

Чтобы рассчитать дату начала лунной недели, когда была совершена транзакция, создайте следующую меру:

```
=lunarweekname(date)
```

Результирующая таблица

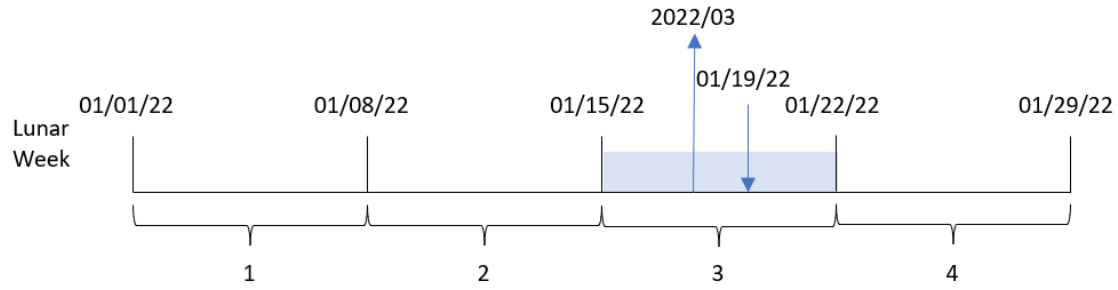
date	=lunarweekname(date)
1/7/2022	2022/01
1/19/2022	2022/03
2/5/2022	2022/06
2/28/2022	2022/09
3/16/2022	2022/11
4/1/2022	2022/13
5/7/2022	2022/19
5/16/2022	2022/20
6/15/2022	2022/24
6/26/2022	2022/26
7/9/2022	2022/28
7/22/2022	2022/29
7/23/2022	2022/30

date	=lunarweekname(date)
7/27/2022	2022/30
8/2/2022	2022/31
8/8/2022	2022/32
8/19/2022	2022/33
9/26/2022	2022/39
10/14/2022	2022/41
10/29/2022	2022/44

Мера `lunar_week_name` создана в объекте диаграммы с использованием функции `lunarweekname()`, где в качестве аргумента функции передано поле `date`.

Функция `lunarweekname()` определяет, к какой лунной неделе относится значение даты, и возвращает год и номер луной недели для этой даты.

Диаграмма функции `lunarweekname()`, пример с объектом диаграммы



Транзакция 8189 совершена 19 января. Функция `lunarweekname()` определяет, что эта дата попадает в лунную неделю, которая начинается 15 января; это третья лунная неделя года. Таким образом, для транзакции возвращается значение `lunar_week_name` = 2022/03.

Пример 5. Сценарий

Скрипт загрузки и выражение диаграммы

Обзор

Откройте редактор загрузки данных и добавьте приведенный ниже скрипт загрузки на новую вкладку.

Скрипт загрузки содержит следующее:

- Набор данных, содержащий набор транзакций за 2022 год, загружается в таблицу под именем `Transactions`.

- Поле даты было предоставлено в формате системной переменной `DateFormat` (MM/DD/YYYY).

Конечному пользователю нужен объект диаграммы, на котором представлены общие продажи по неделям для текущего года. Неделя 1 с продолжительностью семь дней должна начаться 1 января. Этого можно добиться, даже когда это измерение недоступно в модели данных, используя функцию `lunarweekname()` в качестве вычисляемого измерения в диаграмме.

Скрипт загрузки

```
SET DateFormat='MM/DD/YYYY';

Transactions:
Load
*
Inline
[
id,date,amount
8188,1/7/2022,17.17
8189,1/19/2022,37.23
8190,2/28/2022,88.27
8191,2/5/2022,57.42
8192,3/16/2022,53.80
8193,4/1/2022,82.06
8194,5/7/2022,40.39
8195,5/16/2022,87.21
8196,6/15/2022,95.93
8197,6/26/2022,45.89
8198,7/9/2022,36.23
8199,7/22/2022,25.66
8200,7/23/2022,82.77
8201,7/27/2022,69.98
8202,8/2/2022,76.11
8203,8/8/2022,25.12
8204,8/19/2022,46.23
8205,9/26/2022,84.21
8206,10/14/2022,96.24
8207,10/29/2022,67.67
];
```

Результаты

Выполните следующие действия.

1. Загрузите данные и откройте лист. Создайте новую таблицу.
2. Создайте вычисляемое измерение, используя следующее выражение:
   `=lunarweekname(date)`
3. Чтобы рассчитать общий объем продаж, создайте следующую меру агрегирования:
   `=sum(amount)`
4. Задайте параметру меры **Формат чисел** значение **Денежный**.

Результирующая таблица

=lunarweekname(date)	=sum(amount)
2022/01	$17.17
2022/03	$37.23
2022/06	$57.42
2022/09	$88.27
2022/11	$53.80
2022/13	$82.06
2022/19	$40.39
2022/20	$87.21
2022/24	$95.93
2022/26	$45.89
2022/28	$36.23
2022/29	$25.66
2022/30	$152.75
2022/31	$76.11
2022/32	$25.12
2022/33	$46.23
2022/39	$84.21
2022/41	$96.24
2022/44	$67.67

lunarweekstart

Эта функция возвращает значение, соответствующее метке времени первой миллисекунды первого дня лунной недели, содержащей значение, указанное в поле **date**. При определении лунных недель в Qlik Sense первым днем первой недели считается 1 января. Все недели, кроме последней будут содержать ровно 7 дней.

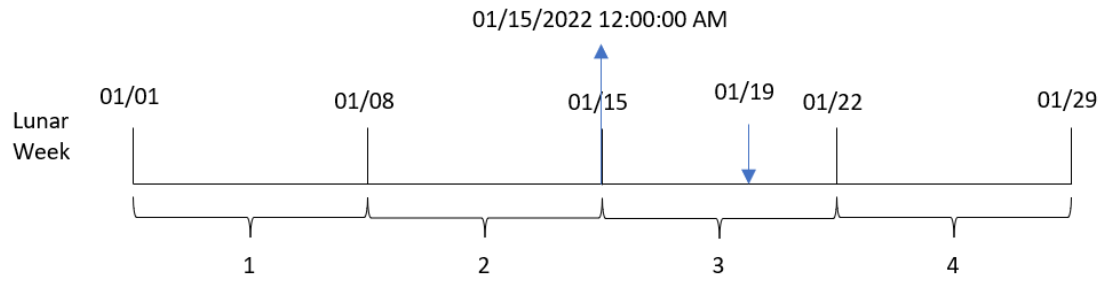
Синтаксис:

```
LunarweekStart(date[, period_no[, first_week_day]])
```

Возвращаемые типы данных: двойное значение

Функция `lunarweekstart()` определяет, на какую лунную неделю выпадает `date`. Затем она возвращает метку времени в формате даты для первой миллисекунды этой недели.

Диаграмма с примером функции `lunarweekstart()`



Аргументы

Аргумент	Описание
date	Дата или метка времени 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.

Когда это следует использовать

Функция `lunarweekstart()` широко используется в составе выражения, когда пользователю требуется учитывать в расчетах часть недели, которая уже прошла. В отличие от функции `weekstart()`, в начале каждого нового календарного года недели начинаются 1 января, и каждая последующая неделя начинается через семь дней. Функцию `lunarweekstart()` не затрагивает системная переменная `FirstWeekDay`.

Например, можно использовать `lunarweekstart()`, если требуется рассчитать проценты, накопленные в течение недели до текущей даты.

Примеры функции

Пример	Результат
`lunarweekstart ('01/12/2013')`	Возвращает 01/08/2013.
`lunarweekstart ('01/12/2013', -1)`	Возвращает 01/01/2013.
`lunarweekstart ('01/12/2013', 0, 1 )`	Возвращает 01/09/2013, поскольку first_week_day = 1, означает, что начало года изменяется переносится на 01/02/2013.

Региональные настройки

Если не указано иное, в примерах, приведенных в данном разделе, используется следующий формат даты: MM/ДД/ГГГГ. Формат даты указан в операторе `SET DateFormat` скрипта загрузки данных. В вашей системе может быть установлен другой формат даты по умолчанию в зависимости от региональных настроек и других факторов. Можно изменить формат в примерах в соответствии с потребностями. Или можно изменить форматы в скрипте загрузки в соответствии с этими примерами.

Региональные настройки по умолчанию в приложениях основаны на системных региональных настройках компьютера или сервера, где установлено ПО Qlik Sense. Если на сервере Qlik Sense, к которому обращается пользователь, выбран шведский язык, то редактор загрузки данных будет использовать шведские региональные настройки для даты, времени и валюты. Эти параметры регионального формата не связаны с языком, отображаемым в интерфейсе пользователя Qlik Sense. Qlik Sense будет отображаться на языке, который используется в браузере.

Пример 1. Без дополнительных аргументов

Скрипт загрузки и результаты

Обзор

Откройте редактор загрузки данных и добавьте приведенный ниже скрипт загрузки на новую вкладку.

Скрипт загрузки содержит следующее:

- Набор данных, содержащий набор транзакций за 2022 год, загружается в таблицу под именем `Transactions`.
- Поле даты было предоставлено в формате системной переменной `DateFormat` (MM/DD/YYYY).
- Создание поля `start_of_week`, возвращающего метку времени начала лунной недели, в течение которой совершены транзакции.

Скрипт загрузки

```
SET DateFormat='MM/DD/YYYY';

Transactions:
    Load
        *,
        lunarweekstart(date) as start_of_week,
        timestamp(lunarweekstart(date)) as start_of_week_timestamp
        ;
Load
*
Inline
[
id,date,amount
8188,1/7/2022,17.17
```

```
8189,1/19/2022,37.23
8190,2/28/2022,88.27
8191,2/5/2022,57.42
8192,3/16/2022,53.80
8193,4/1/2022,82.06
8194,5/7/2022,40.39
8195,5/16/2022,87.21
8196,6/15/2022,95.93
8197,6/26/2022,45.89
8198,7/9/2022,36.23
8199,7/22/2022,25.66
8200,7/23/2022,82.77
8201,7/27/2022,69.98
8202,8/2/2022,76.11
8203,8/8/2022,25.12
8204,8/19/2022,46.23
8205,9/26/2022,84.21
8206,10/14/2022,96.24
8207,10/29/2022,67.67
];
```

Результаты

Загрузите данные и откройте лист. Создайте новую таблицу и добавьте эти поля как измерения:

- `date`
- `start_of_week`
- `start_of_week_timestamp`

Результирующая таблица

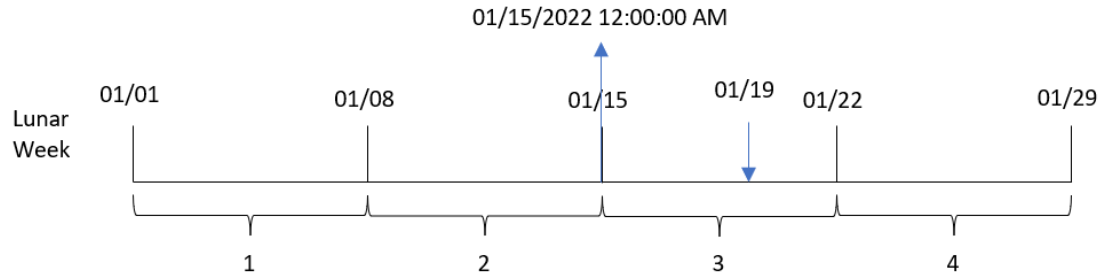
date	start_of_week	start_of_week_timestamp
1/7/2022	01/01/2022	1/1/2022 12:00:00 AM
1/19/2022	01/15/2022	1/15/2022 12:00:00 AM
2/5/2022	02/05/2022	2/5/2022 12:00:00 AM
2/28/2022	02/26/2022	2/26/2022 12:00:00 AM
3/16/2022	03/12/2022	3/12/2022 12:00:00 AM
4/1/2022	03/26/2022	3/26/2022 12:00:00 AM
5/7/2022	05/07/2022	5/7/2022 12:00:00 AM
5/16/2022	05/14/2022	5/14/2022 12:00:00 AM
6/15/2022	06/11/2022	6/11/2022 12:00:00 AM
6/26/2022	06/25/2022	6/25/2022 12:00:00 AM
7/9/2022	07/09/2022	7/9/2022 12:00:00 AM
7/22/2022	07/16/2022	7/16/2022 12:00:00 AM

date	start_of_week	start_of_week_timestamp
7/23/2022	07/23/2022	7/23/2022 12:00:00 AM
7/27/2022	07/23/2022	7/23/2022 12:00:00 AM
8/2/2022	07/30/2022	7/30/2022 12:00:00 AM
8/8/2022	08/06/2022	8/6/2022 12:00:00 AM
8/19/2022	08/13/2022	8/13/2022 12:00:00 AM
9/26/2022	09/24/2022	9/24/2022 12:00:00 AM
10/14/2022	10/08/2022	10/8/2022 12:00:00 AM
10/29/2022	10/29/2022	10/29/2022 12:00:00 AM

Поле «`start_of_week`» создано предшествующим оператором load с использованием функции `lunarweekstart()`, где в качестве аргумента функции передано поле `date`.

Функция `lunarweekstart()` определяет, к какой лунной неделе относится дата, и возвращает метку времени для первой миллисекунды этой недели.

Диаграмма функции `lunarweekstart()`, пример без дополнительных аргументов



Транзакция 8189 совершена 19 января. Функция `lunarweekstart()` определяет, что лунная неделя начинается 15 января. Поэтому значение `start_of_week` для этой транзакции возвращает первую миллисекунду лунной недели, то есть 15 января в 00:00:00 (12:00:00 AM).

Пример 2. Скрипт period_no

Скрипт загрузки и результаты

Обзор

Откройте редактор загрузки данных и добавьте приведенный ниже скрипт загрузки на новую вкладку.

Скрипт загрузки содержит следующее:

- Используется тот же набор данных и сценарий, что в первом примере.
- Создание поля `previous_lunar_week_start`, возвращающего метку времени начала лунной недели, которая предшествует совершению транзакции.

Скрипт загрузки

```
SET DateFormat='MM/DD/YYYY';

Transactions:
    Load
        *,
        lunarweekstart(date,-1) as previous_lunar_week_start,
        timestamp(lunarweekstart(date,-1)) as previous_lunar_week_start_timestamp
        ;
Load
*
Inline
[
id,date,amount
8188,1/7/2022,17.17
8189,1/19/2022,37.23
8190,2/28/2022,88.27
8191,2/5/2022,57.42
8192,3/16/2022,53.80
8193,4/1/2022,82.06
8194,5/7/2022,40.39
8195,5/16/2022,87.21
8196,6/15/2022,95.93
8197,6/26/2022,45.89
8198,7/9/2022,36.23
8199,7/22/2022,25.66
8200,7/23/2022,82.77
8201,7/27/2022,69.98
8202,8/2/2022,76.11
8203,8/8/2022,25.12
8204,8/19/2022,46.23
8205,9/26/2022,84.21
8206,10/14/2022,96.24
8207,10/29/2022,67.67
];
```

Результаты

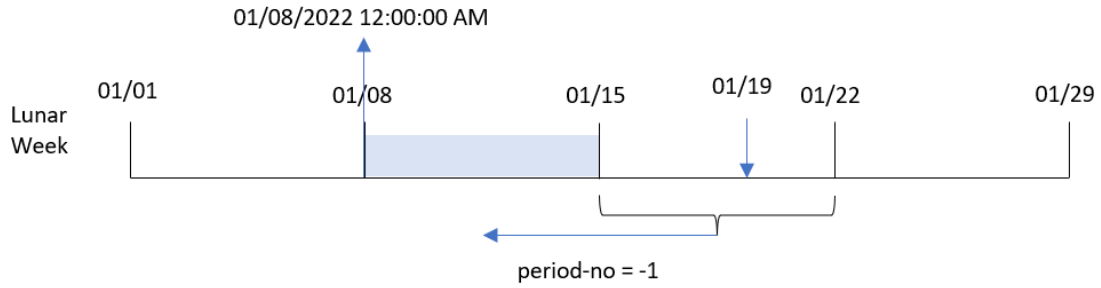
Результирующая таблица

date	previous_lunar_week_start	previous_lunar_week_start_timestamp
1/7/2022	12/24/2021	12/24/2021 12:00:00 AM
1/19/2022	01/08/2022	1/8/2022 12:00:00 AM
2/5/2022	01/29/2022	1/29/2022 12:00:00 AM

date	previous_lunar_week_start	previous_lunar_week_start_timestamp
2/28/2022	02/19/2022	2/19/2022 12:00:00 AM
3/16/2022	03/05/2022	3/5/2022 12:00:00 AM
4/1/2022	03/19/2022	3/19/2022 12:00:00 AM
5/7/2022	04/30/2022	4/30/2022 12:00:00 AM
5/16/2022	05/07/2022	5/7/2022 12:00:00 AM
6/15/2022	06/04/2022	6/4/2022 12:00:00 AM
6/26/2022	06/18/2022	6/18/2022 12:00:00 AM
7/9/2022	07/02/2022	7/2/2022 12:00:00 AM
7/22/2022	07/09/2022	7/9/2022 12:00:00 AM
7/23/2022	07/16/2022	7/16/2022 12:00:00 AM
7/27/2022	07/16/2022	7/16/2022 12:00:00 AM
8/2/2022	07/23/2022	7/23/2022 12:00:00 AM
8/8/2022	07/30/2022	7/30/2022 12:00:00 AM
8/19/2022	08/06/2022	8/6/2022 12:00:00 AM
9/26/2022	09/17/2022	9/17/2022 12:00:00 AM
10/14/2022	10/01/2022	10/1/2022 12:00:00 AM
10/29/2022	10/22/2022	10/22/2022 12:00:00 AM

В этом случае, так как в качестве аргумента смещения в функции `lunarweekstart()` использовалось `period_no` = -1, функция сначала определяет лунную неделю, в течение которой совершаются транзакции. Затем она возвращается на неделю назад и определяет первую миллисекунду предыдущей лунной недели.

Диаграмма функции `lunarweekstart()`, пример с аргументом period_no



Транзакция 8189 совершена 19 января. Функция `lunarweekstart()` определяет, что лунная неделя начинается 15 января. Таким образом, предыдущая лунная неделя началась 8 января в 00:00:00 (12:00:00 AM). Именно это значение возвращается для поля `previous_lunar_week_start`.

Пример 3. Аргумент first_week_day

Скрипт загрузки и результаты

Обзор

Откройте редактор загрузки данных и добавьте приведенный ниже скрипт загрузки на новую вкладку.

Скрипт загрузки содержит тот же набор данных и сценарий, что в первом примере. В этом примере лунные недели начинаются с 5 января.

Скрипт загрузки

```
SET DateFormat='MM/DD/YYYY';

Transactions:
    Load
        *,
        lunarweekstart(date,0,4) as start_of_week,
        timestamp(lunarweekstart(date,0,4)) as start_of_week_timestamp
        ;
Load
*
Inline
[
id,date,amount
8188,1/7/2022,17.17
8189,1/19/2022,37.23
8190,2/28/2022,88.27
8191,2/5/2022,57.42
8192,3/16/2022,53.80
8193,4/1/2022,82.06
8194,5/7/2022,40.39
8195,5/16/2022,87.21
8196,6/15/2022,95.93
8197,6/26/2022,45.89
8198,7/9/2022,36.23
8199,7/22/2022,25.66
8200,7/23/2022,82.77
8201,7/27/2022,69.98
8202,8/2/2022,76.11
8203,8/8/2022,25.12
8204,8/19/2022,46.23
8205,9/26/2022,84.21
8206,10/14/2022,96.24
8207,10/29/2022,67.67
];
```

Результаты

Загрузите данные и откройте лист. Создайте новую таблицу и добавьте эти поля как измерения:

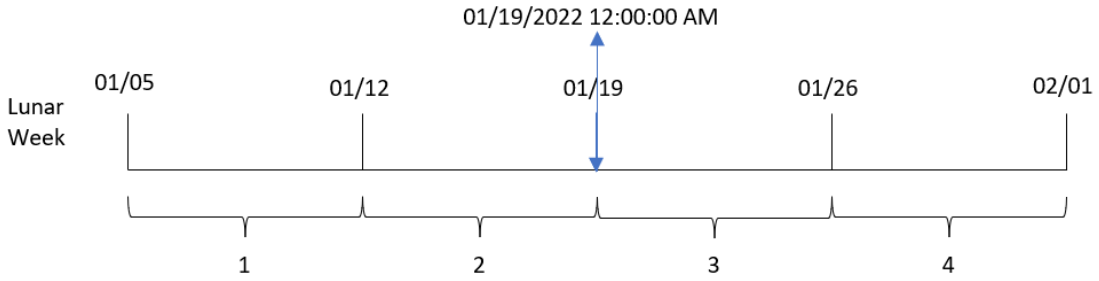
- date
- start_of_week
- start_of_week_timestamp

Результирующая таблица

date	start_of_week	start_of_week_timestamp
1/7/2022	01/05/2022	1/5/2022 12:00:00 AM
1/19/2022	01/19/2022	1/19/2022 12:00:00 AM
2/5/2022	02/02/2022	2/2/2022 12:00:00 AM
2/28/2022	02/23/2022	2/23/2022 12:00:00 AM
3/16/2022	03/16/2022	3/16/2022 12:00:00 AM
4/1/2022	03/30/2022	3/30/2022 12:00:00 AM
5/7/2022	05/04/2022	5/4/2022 12:00:00 AM
5/16/2022	05/11/2022	5/11/2022 12:00:00 AM
6/15/2022	06/15/2022	6/15/2022 12:00:00 AM
6/26/2022	06/22/2022	6/22/2022 12:00:00 AM
7/9/2022	07/06/2022	7/6/2022 12:00:00 AM
7/22/2022	07/20/2022	7/20/2022 12:00:00 AM
7/23/2022	07/20/2022	7/20/2022 12:00:00 AM
7/27/2022	07/27/2022	7/27/2022 12:00:00 AM
8/2/2022	07/27/2022	7/27/2022 12:00:00 AM
8/8/2022	08/03/2022	8/3/2022 12:00:00 AM
8/19/2022	08/17/2022	8/17/2022 12:00:00 AM
9/26/2022	09/21/2022	9/21/2022 12:00:00 AM
10/14/2022	10/12/2022	10/12/2022 12:00:00 AM
10/29/2022	10/26/2022	10/26/2022 12:00:00 AM

В этом примере используется аргумент `first_week_date` = 4 в функции `lunarweekstart()`, потому начало года переносится с 1 на 5 января.

Диаграмма функции `lunarweekstart()`, пример с аргументом first_week_day



Транзакция 8189 совершена 19 января. Так как лунные недели начинаются 5 января, функция `lunarweekstart()` определяет, что лунная неделя, к которой относится 19 января, также начинается 19 января в 00:00:00 (12:00:00 AM). Поэтому это значение возвращается для поля `start_of_week`.

Пример 4. Пример объекта диаграммы

Скрипт загрузки и выражение диаграммы

Обзор

Откройте редактор загрузки данных и добавьте приведенный ниже скрипт загрузки на новую вкладку.

Скрипт загрузки содержит тот же набор данных и сценарий, что в первом примере.

Однако в этом примере в приложение загружается неизмененный набор данных. Расчет, возвращающий метку времени начала лунной недели, в течение которой совершены транзакции, создается как мера в объекте диаграммы в приложении.

Скрипт загрузки

```
Transactions:
Load
*
Inline
[
id,date,amount
8188,1/7/2022,17.17
8189,1/19/2022,37.23
8190,2/28/2022,88.27
8191,2/5/2022,57.42
8192,3/16/2022,53.80
8193,4/1/2022,82.06
8194,5/7/2022,40.39
8195,5/16/2022,87.21
8196,6/15/2022,95.93
8197,6/26/2022,45.89
8198,7/9/2022,36.23
8199,7/22/2022,25.66
```

```
8200,7/23/2022,82.77
8201,7/27/2022,69.98
8202,8/2/2022,76.11
8203,8/8/2022,25.12
8204,8/19/2022,46.23
8205,9/26/2022,84.21
8206,10/14/2022,96.24
8207,10/29/2022,67.67
];
```

Результаты

Загрузите данные и откройте лист. Создайте новую таблицу и добавьте это поле как измерение: `date`.

Добавьте следующие меры:

```
=lunarweekstart(date)
```

```
=timestamp(lunarweekstart(date))
```

Результирующая таблица

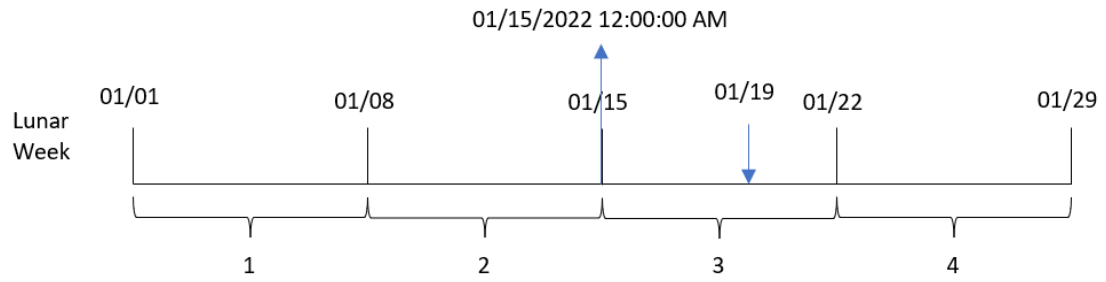
date	=lunarweekstart(date)	=timestamp(lunarweekstart(date))
1/7/2022	01/01/2022	1/1/2022 12:00:00 AM
1/19/2022	01/15/2022	1/15/2022 12:00:00 AM
2/5/2022	02/05/2022	2/5/2022 12:00:00 AM
2/28/2022	02/26/2022	2/26/2022 12:00:00 AM
3/16/2022	03/12/2022	3/12/2022 12:00:00 AM
4/1/2022	03/26/2022	3/26/2022 12:00:00 AM
5/7/2022	05/07/2022	5/7/2022 12:00:00 AM
5/16/2022	05/14/2022	5/14/2022 12:00:00 AM
6/15/2022	06/11/2022	6/11/2022 12:00:00 AM
6/26/2022	06/25/2022	6/25/2022 12:00:00 AM
7/9/2022	07/09/2022	7/9/2022 12:00:00 AM
7/22/2022	07/16/2022	7/16/2022 12:00:00 AM
7/23/2022	07/23/2022	7/23/2022 12:00:00 AM
7/27/2022	07/23/2022	7/23/2022 12:00:00 AM
8/2/2022	07/30/2022	7/30/2022 12:00:00 AM
8/8/2022	08/06/2022	8/6/2022 12:00:00 AM
8/19/2022	08/13/2022	8/13/2022 12:00:00 AM
9/26/2022	09/24/2022	9/24/2022 12:00:00 AM

date	=lunarweekstart(date)	=timestamp(lunarweekstart(date))
10/14/2022	10/08/2022	10/8/2022 12:00:00 AM
10/29/2022	10/29/2022	10/29/2022 12:00:00 AM

Мера `start_of_week` создана в объекте диаграммы с использованием функции `lunarweekstart()`, где в качестве аргумента функции передано поле даты.

Функция `lunarweekstart()` определяет, к какой лунной неделе относится значение даты, и возвращает метку времени для последней миллисекунды этой недели.

Диаграмма функции `lunarweekstart()`, пример с объектом диаграммы



Транзакция 8189 совершена 19 января. Функция `lunarweekstart()` определяет, что лунная неделя начинается 15 января. Поэтому значение `start_of_week` для этой транзакции возвращает первую миллисекунду этого дня, то есть 15 января в 00:00:00 (12:00:00 AM).

Пример 5. Сценарий

Скрипт загрузки и выражение диаграммы

Обзор

Откройте редактор загрузки данных и добавьте приведенный ниже скрипт загрузки на новую вкладку.

Скрипт загрузки содержит следующее:

- Набор данных, содержащий набор остатков по кредитам, загруженный в таблицу под именем `Loans`.
- Данные, включая идентификаторы кредитов, остаток на начало недели и простую процентную ставку, взимаемую по каждому кредиту за год.

Конечному пользователю требуется объект диаграммы, который будет отображать по идентификатору кредита текущий процент, начисленный по каждому кредиту в течение недели до текущей даты.

Скрипт загрузки

```
Loans:
Load
*
Inline
[
loan_id,start_balance,rate
8188,$10000.00,0.024
8189,$15000.00,0.057
8190,$17500.00,0.024
8191,$21000.00,0.034
8192,$90000.00,0.084
];
```

Результаты

Выполните следующие действия.

1. Загрузите данные и откройте лист. Создайте новую таблицу.
2. Добавьте следующие поля как измерения:
   - `loan_id`
   - `start_balance`
3. Затем, чтобы рассчитать накопленный процент, создайте следующую меру:
   `=start_balance*(rate*(today(1)-lunarweekstart(today(1)))/365)`
4. Задайте параметру меры **Формат чисел** значение **Денежный**.

Результирующая таблица

loan_id	start_balance	=start_balance*(rate*(today(1)- lunarweekstart (today(1)))/365)
8188	$10000.00	$15.07
8189	$15000.00	$128.84
8190	$17500.00	$63.29
8191	$21000.00	$107.59
8192	$90000.00	$1139.18

Используя сегодняшнюю дату в качестве единственного аргумента, функция `lunarweekstart()` возвращает дату начала текущего года. Вычитая этот результат из текущей даты, выражение возвращает количество дней, прошедших до сих пор в течение этой недели.

Затем это значение умножается на процентную ставку и делится на 365, чтобы получить эффективную процентную ставку начисленную за этот период. После этого результат умножается на начальный остаток кредита, чтобы вернуть проценты, начисленные до сих пор в течение этой недели.

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

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).

Когда это следует использовать

Функция `makedate()` обычно используется в скриптах для создания данных с целью формирования календаря. Это также полезно, когда поле даты недоступно напрямую в виде даты и требуется выполнить преобразования для извлечения компонентов года, месяца и даты.

В этих примерах используется формат даты MM/DD/YYYY. Формат даты указан в операторе `SET DateFormat` в верхней части скрипта загрузки данных. Измените формат в примерах согласно своим пожеланиям.

Примеры функции

Пример	Результат
`makedate(2012)`	Возвращает 01/01/2012.
`makedate(12)`	Возвращает 01/01/2012.
`makedate(2012,12)`	Возвращает 12/01/2012.
`makedate(2012,2,14)`	Возвращает 02/14/2012.

Региональные настройки

Если не указано иное, в примерах, приведенных в данном разделе, используется следующий формат даты: ММ/ДД/ГГГГ. Формат даты указан в операторе `SET DateFormat` скрипта загрузки данных. В вашей системе может быть установлен другой формат даты по умолчанию в зависимости от региональных настроек и других факторов. Можно изменить формат в примерах в соответствии с потребностями. Или можно изменить форматы в скрипте загрузки в соответствии с этими примерами.

Региональные настройки по умолчанию в приложениях основаны на системных региональных настройках компьютера или сервера, где установлено ПО Qlik Sense. Если на сервере Qlik Sense, к которому обращается пользователь, выбран шведский язык, то редактор загрузки данных будет использовать шведские региональные настройки для даты, времени и валюты. Эти параметры регионального формата не связаны с языком, отображаемым в интерфейсе пользователя Qlik Sense. Qlik Sense будет отображаться на языке, который используется в браузере.

Пример 1. Базовый пример

Скрипт загрузки и результаты

Обзор

Откройте редактор загрузки данных и добавьте приведенный ниже скрипт загрузки на новую вкладку.

Скрипт загрузки содержит следующее:

- Набор данных, содержащий набор транзакций за 2018 год, загружается в таблицу под именем `Transactions`.
- Поле даты было предоставлено в формате системной переменной `DateFormat` (MM/DD/YYYY).
- Создание поля `transaction_date`, которое возвращает дату в формате MM/DD/YYYY.

Скрипт загрузки

```
SET DateFormat='MM/DD/YYYY';

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, customer_id
3750, 2018, 08, 30, 12423.56, 23, 2038593
3751, 2018, 09, 07, 5356.31, 6, 203521
3752, 2018, 09, 16, 15.75, 1, 5646471
3753, 2018, 09, 22, 1251, 7, 3036491
3754, 2018, 09, 22, 21484.21, 1356, 049681
3756, 2018, 09, 22, -59.18, 2, 2038593
3757, 2018, 09, 23, 3177.4, 21, 203521
];
```

Результаты

Загрузите данные и откройте лист. Создайте новую таблицу и добавьте эти поля как измерения:

- transaction_year
- transaction_month
- transaction_day
- transaction_date

Результирующая таблица

transaction_year	transaction_month	transaction_day	transaction_date
2018	08	30	08/30/2018
2018	09	07	09/07/2018
2018	09	16	09/16/2018
2018	09	22	09/22/2018
2018	09	23	09/23/2018

Поле transaction_date создано предшествующим оператором load с использованием функции makedate(), где в качестве аргумента функции переданы поля года, месяца и даты.

Затем функция объединяет и преобразует эти значения в поле даты, возвращая результаты в формате системной переменной DateFormat.

Пример 2. Измененная переменная DateFormat

Скрипт загрузки и результаты

Обзор

Откройте редактор загрузки данных и добавьте приведенный ниже скрипт загрузки на новую вкладку.

Скрипт загрузки содержит следующее:

- Используется тот же набор данных и сценарий, что в первом примере.
- Создание поля transaction_date в формате DD/MM/YYYY без изменения системной переменной DateFormat.

Скрипт загрузки

```
SET DateFormat='MM/DD/YYYY';

Transactions:
    Load
        *,
        date(makedate(transaction_year, transaction_month, transaction_day), 'DD/MM/YYYY') as
transaction_date
        ;
Load * Inline [
transaction_id, transaction_year, transaction_month, transaction_day, transaction_amount,
transaction_quantity, customer_id
```

```
3750, 2018, 08, 30, 12423.56, 23, 2038593
3751, 2018, 09, 07, 5356.31, 6, 203521
3752, 2018, 09, 16, 15.75, 1, 5646471
3753, 2018, 09, 22, 1251, 7, 3036491
3754, 2018, 09, 22, 21484.21, 1356, 049681
3756, 2018, 09, 22, -59.18, 2, 2038593
3757, 2018, 09, 23, 3177.4, 21, 203521
];
```

Результаты

Загрузите данные и откройте лист. Создайте новую таблицу и добавьте эти поля как измерения:

- `transaction_year`
- `transaction_month`
- `transaction_day`
- `transaction_date`

Результирующая таблица

transaction_year	transaction_month	transaction_day	transaction_date
2018	08	30	30/08/2018
2018	09	07	07/09/2018
2018	09	16	16/09/2018
2018	09	22	22/09/2018
2018	09	23	23/09/2018

В этом примере функция `makedate()` вложена в функцию `date()`. Второй аргумент функции `date()` задает формат для результатов функции `makedate()` — DD/MM/YYYY.

Пример 3. Пример объекта диаграммы

Скрипт загрузки и выражение диаграммы

Обзор

Откройте редактор загрузки данных и добавьте приведенный ниже скрипт загрузки на новую вкладку.

Скрипт загрузки содержит следующее:

- Набор данных, содержащий набор транзакций за 2018 год, загружается в таблицу под именем `Transactions`.
- Даты транзакций, предоставленные в виде двух полей: `year` и `month`.

Создайте меру объекта `transaction_date`, которая возвращает дату в формате MM/DD/YYYY.

Скрипт загрузки

```
SET DateFormat='MM/DD/YYYY';

Transactions:
Load * Inline [
transaction_id, transaction_year, transaction_month, transaction_amount, transaction_quantity,
customer_id
3750, 2018, 08, 12423.56, 23, 2038593
3751, 2018, 09, 5356.31, 6, 203521
3752, 2018, 09, 15.75, 1, 5646471
3753, 2018, 09, 1251, 7, 3036491
3754, 2018, 09, 21484.21, 1356, 049681
3756, 2018, 09, -59.18, 2, 2038593
3757, 2018, 09, 3177.4, 21, 203521
];
```

Результаты

Загрузите данные и откройте лист. Создайте новую таблицу и добавьте эти поля как измерения:

- `year`
- `month`

Чтобы определить `transaction_date`, создайте эту меру:

```
=makedate(transaction_year,transaction_month)
```

Результирующая таблица

transaction_year	transaction_month	transaction_date
2018	08	08/01/2018
2018	09	09/01/2018

Мера `transaction_date` создана в объекте диаграммы с использованием функции `makedate()`, где в качестве аргументов функции переданы поля года и месяца.

Затем функция объединяет эти значения, а также добавляет предполагаемое значение дня 01. Затем эти значения преобразуются в поле даты, а результаты возвращаются в формате системной переменной `DateFormat`.

Пример 4. Сценарий

Скрипт загрузки и выражение диаграммы

Обзор

Создайте набор данных календаря для календарного года 2022.

Скрипт загрузки

```
SET DateFormat='MM/DD/YYYY';

Calendar:
       load
                *
    where year(date)=2022;
load
       date(recno()+makedate(2021,12,31)) as date
AutoGenerate 400;
```

Результаты

Результирующая таблица

date
01/01/2022
01/02/2022
01/03/2022
01/04/2022
01/05/2022
01/06/2022
01/07/2022
01/08/2022
01/09/2022
01/10/2022
01/11/2022
01/12/2022
01/13/2022
01/14/2022
01/15/2022
01/16/2022
01/17/2022
01/18/2022
01/19/2022
01/20/2022
01/21/2022

date
01/22/2022
01/23/2022
01/24/2022
01/25/2022
+ 340 строк

Функция `makedate()` создает значение даты для 31 декабря 2021 года. Функция `recno()` предоставляет номер для текущей записи, загружаемой в таблицу, начиная от 1. Таким образом первая запись имеет дату 1 января 2022 года. Каждый последующий `recno()` будет увеличивать дату на 1. Это выражение помещается в функцию `date()` для преобразования значения в дату. Функция `autogenerate` повторяет этот процесс 400 раз. В завершение, с помощью предыдущей загрузки можно использовать условие `where`, чтобы загрузить даты только из 2022 года. Этот скрипт создает календарь, содержащий все даты 2022 года.

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

Аргумент	Описание
hh	Час – целое число.
mm	Минута – целое число. Если минута не задана, используется 00.
ss	Секунда – целое число. Если секунда не задана, используется 00.

Когда это следует использовать

Функция `maketime()` обычно используется в скриптах для создания данных с целью формирования поля времени. В некоторых случаях, когда поле времени создается на основе входного текста, эту функцию можно использовать для получения времени на основе его компонентов.

В этих примерах используется формат времени `h:mm:ss`. Формат времени указан в операторе `SET TimeFormat` в верхней части скрипта загрузки данных. Измените формат в примерах согласно своим пожеланиям.

Примеры функции

Пример	Результат
`maketime(22)`	Возвращает 22:00:00.
`maketime(22, 17)`	Возвращает 22:17:00.
`maketime(22,17,52 )`	Возвращает 22:17:52.

Региональные настройки

Если не указано иное, в примерах, приведенных в данном разделе, используется следующий формат даты: ММ/ДД/ГГГГ. Формат даты указан в операторе `SET DateFormat` скрипта загрузки данных. В вашей системе может быть установлен другой формат даты по умолчанию в зависимости от региональных настроек и других факторов. Можно изменить формат в примерах в соответствии с потребностями. Или можно изменить форматы в скрипте загрузки в соответствии с этими примерами.

Региональные настройки по умолчанию в приложениях основаны на системных региональных настройках компьютера или сервера, где установлено ПО Qlik Sense. Если на сервере Qlik Sense, к которому обращается пользователь, выбран шведский язык, то редактор загрузки данных будет использовать шведские региональные настройки для даты, времени и валюты. Эти параметры регионального формата не связаны с языком, отображаемым в интерфейсе пользователя Qlik Sense. Qlik Sense будет отображаться на языке, который используется в браузере.

Пример 1. Функция maketime()

Скрипт загрузки и результаты

Обзор

Откройте редактор загрузки данных и добавьте приведенный ниже скрипт загрузки на новую вкладку.

Скрипт загрузки содержит следующее:

- Набор данных, содержащий набор транзакций, загруженный в таблицу под именем `Transactions`.
- Время транзакций, предоставленное в виде трех полей: `hours`, `minutes` и `seconds`.
- Создание поля `transaction_time`, которое возвращает время в формате системной переменной `TimeFormat`.

Скрипт загрузки

```
SET TimeFormat='h:mm:ss TT';

Transactions:
    Load
        *,
        maketime(transaction_hour, transaction_minute, transaction_second) as transaction_time
```

```
        ;
Load * Inline [
transaction_id, transaction_hour, transaction_minute, transaction_second, transaction_amount,
transaction_quantity, customer_id
3750, 18, 43, 30, 12423.56, 23, 2038593
3751, 6, 32, 07, 5356.31, 6, 203521
3752, 12, 09, 16, 15.75, 1, 5646471
3753, 21, 43, 41, 7, 3036491
3754, 17, 55, 22, 21484.21, 1356, 049681
3756, 2, 52, 22, -59.18, 2, 2038593
3757, 9, 25, 23, 3177.4, 21, 203521
];
```

Результаты

Загрузите данные и откройте лист. Создайте новую таблицу и добавьте эти поля как измерения:

- `transaction_hour`
- `transaction_minute`
- `transaction_second`
- `transaction_time`

Результирующая таблица

transaction_hour	transaction_minute	transaction_second	transaction_time
2	52	22	2:52:22 AM
6	32	07	6:32:07 AM
9	25	23	9:25:23 AM
12	09	16	12:09:16 PM
17	55	22	5:55:22 PM
18	43	30	6:43:30 PM
21	43	41	9:43:41 PM

Поле `transaction_time` создано предшествующим оператором load с использованием функции `maketime()`, где в качестве аргумента функции переданы поля часов, минут и секунд.

Затем функция объединяет и преобразует эти значения в поле времени, возвращая результаты в формате времени системной переменной `TimeFormat`.

Пример 2. Функция time()

Скрипт загрузки и результаты

Обзор

Откройте редактор загрузки данных и добавьте приведенный ниже скрипт загрузки на новую вкладку.

Скрипт загрузки содержит следующее:

- Используется тот же набор данных и сценарий, что в первом примере.
- Создание поля `transaction_time`, которое позволяет отображать результаты в 24-часовом формате времени, не изменяя системную переменную `TimeFormat`.

Скрипт загрузки

```
SET TimeFormat='h:mm:ss TT';

Transactions:
    Load
        *,
        time(maketime(transaction_hour, transaction_minute, transaction_second),'h:mm:ss') as
transaction_time
        ;
Load * Inline [
transaction_id, transaction_hour, transaction_minute, transaction_second, transaction_amount,
transaction_quantity, customer_id
3750, 18, 43, 30, 12423.56, 23, 2038593
3751, 6, 32, 07, 5356.31, 6, 203521
3752, 12, 09, 16, 15.75, 1, 5646471
3753, 21, 43, 41, 7, 3036491
3754, 17, 55, 22, 21484.21, 1356, 049681
3756, 2, 52, 22, -59.18, 2, 2038593
3757, 9, 25, 23, 3177.4, 21, 203521
];
```

Результаты

Загрузите данные и откройте лист. Создайте новую таблицу и добавьте эти поля как измерения:

- `transaction_hour`
- `transaction_minute`
- `transaction_second`
- `transaction_time`

Результирующая таблица

transaction_hour	transaction_minute	transaction_second	transaction_time
2	52	22	2:52:22
6	32	07	6:32:07
9	25	23	9:25:23
12	09	16	12:09:16
17	55	22	17:55:22
18	43	30	18:43:30
21	43	41	21:43:41

В этом примере функция `maketime()` вложена в функцию `time()`. Второй аргумент функции `time()` задает формат для результатов функции `maketime()` — `h:mm:ss`.

Пример 3. Пример объекта диаграммы

Скрипт загрузки и выражение диаграммы

Обзор

Откройте редактор загрузки данных и добавьте приведенный ниже скрипт загрузки на новую вкладку.

Скрипт загрузки содержит следующее:

- Набор данных, содержащий набор транзакций, загруженный в таблицу под именем `Transactions`.
- Время транзакций, предоставленное в виде двух полей: `hours` и `minutes`.
- Создание поля `transaction_time`, которое возвращает время в формате системной переменной `TimeFormat`.

Создайте меру объекта диаграммы `transaction_time`, которая возвращает дату в формате `h:mm:ss TT`.

Скрипт загрузки

```
SET TimeFormat='h:mm:ss TT';

Transactions:
Load * Inline [
transaction_id, transaction_hour, transaction_minute, transaction_amount, transaction_
quantity, customer_id
3750, 18, 43, 12423.56, 23, 2038593
3751, 6, 32, 5356.31, 6, 203521
3752, 12, 09, 15.75, 1, 5646471
3753, 21, 43, 7, 3036491
3754, 17, 55, 21484.21, 1356, 049681
```

```
3756, 2, 52, -59.18, 2, 2038593
3757, 9, 25, 3177.4, 21, 203521
];
```

Результаты

Загрузите данные и откройте лист. Создайте новую таблицу и добавьте эти поля как измерения:

- transaction_hour
- transaction_minute

Чтобы вычислить transaction_time, создайте эту меру:

```
=maketime(transaction_hour,transaction_minute)
```

Результирующая таблица

transaction_hour	transaction_minute	=maketime(transaction_hour, transaction_minute)
2	52	2:52:00 AM
6	32	6:32:00 AM
9	25	9:25:00 AM
12	09	12:09:00 PM
17	55	5:55:00 PM
18	43	6:43:30 PM
21	43	9:43:00 PM

Мера transaction_time создана в объекте диаграммы с использованием функции maketime(), где в качестве аргументов функции переданы поля часов и минут.

Затем функция объединяет эти значения, для секунд задается значение 00. Затем эти значения преобразуются в поле времени, а результаты возвращаются в формате системной переменной TimeFormat.

Пример 4. Сценарий

Скрипт загрузки и выражение диаграммы

Обзор

Создайте набор данных календаря для января 2022 года с разбивкой на восьмичасовые приращения.

Скрипт загрузки

```
SET TimestampFormat='M/D/YYYY h:mm:ss[.fff] TT';

tmpCalendar:
       load
```

```
              *
    where year(date)=2022;
load
       date(recno()+makedate(2021,12,31)) as date
AutoGenerate 31;

Left join(tmpCalendar)
load
       maketime((recno()-1)*8,00,00) as time
autogenerate 3;

Calendar:
load
       timestamp(date + time) as timestamp
resident tmpCalendar;

drop table tmpCalendar;
```

Результаты

Результирующая таблица

timestamp
1/1/2022 12:00:00 AM
1/1/2022 8:00:00 AM
1/1/2022 4:00:00 PM
1/2/2022 12:00:00 AM
1/2/2022 8:00:00 AM
1/2/2022 4:00:00 PM
1/3/2022 12:00:00 AM
1/3/2022 8:00:00 AM
1/3/2022 4:00:00 PM
1/4/2022 12:00:00 AM
1/4/2022 8:00:00 AM
1/4/2022 4:00:00 PM
1/5/2022 12:00:00 AM
1/5/2022 8:00:00 AM
1/5/2022 4:00:00 PM
1/6/2022 12:00:00 AM
1/6/2022 8:00:00 AM

timestamp
1/6/2022 4:00:00 PM
1/7/2022 12:00:00 AM
1/7/2022 8:00:00 AM
1/7/2022 4:00:00 PM
1/8/2022 12:00:00 AM
1/8/2022 8:00:00 AM
1/8/2022 4:00:00 PM
1/9/2022 12:00:00 AM
+ 68 строк

Начальная функция `autogenerate` создает календарь, содержащий все даты в январе, в таблице под именем `tmpCalendar`.

Создается вторая таблица, содержащая три записи. Для каждой записи берется `recno() - 1` (значения 0, 1, 2), результат умножается на 8, в результате создаются значения 0, 8, 16. Эти значения используются в качестве параметра часа в функции `maketime()`, минутам и секундам задается значение 0. В результате, таблица содержит три поля времени: 12:00:00 AM, 8:00:00 AM и 4:00:00 PM.

Эта таблица присоединяется к таблице `tmpCalendar`. Так как между соединяемыми таблицами отсутствуют совпадающие поля, строки времени добавляются к каждой строке даты. В результате каждая строка даты теперь повторяется три раза для каждого значения времени.

В заключение создается таблица Calendar с использованием резидентной загрузки таблицы `tmpCalendar`. Поля даты и времени объединены и помещены в функцию `timestamp()` для создания поля метки времени.

После этого таблица `tmpCalendar` отбрасывается.

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

Функция `makeweekdate()` доступна как функция скрипта и как функция диаграммы. Функция вычислит дату на основе переданных ей параметров. Если параметр дня недели опускается, функция возвращает дату понедельника этой недели.

Функция `makeweekdate()` не учитывает системные переменные `BrokenWeek`, `ReferenceDay` или `FirstWeekDay`. Неделя 1 начинается в первый понедельник января. Например, в 2022 году неделя 1 начинается 3 января.

Аргументы

Аргумент	Описание
YYYY	Год – целое число.
WW	Неделя – целое число. Неделя может иметь положительное или отрицательное значение, можно задать значение больше 52, чтобы возвращать даты другого года.
D	День недели – целое число. Если день недели не задан, используется 0 (понедельник). Остальные дни недели назначаются следующим образом: 1 – вторник, 2 – среда, 3 – четверг, 4 – пятница, 5 – суббота и 6 – воскресенье.

Когда это следует использовать

Функция `makeweekdate()` обычно используется в скрипте для создания данных с целью формирования списка дат или вычисления дат, когда год, неделя и день недели предоставляются во входных данных.

Примеры функции

Пример	Результат
`makeweekdate(2014,6,6)`	возвращает 02/09/2014
`makeweekdate(2014,6,1)`	возвращает02/04/2014
`makeweekdate(2014,6)`	возвращает 02/03/2014 (для недели допускается значение 0)

Региональные настройки

Если не указано иное, в примерах, приведенных в данном разделе, используется следующий формат даты: ММ/ДД/ГГГГ. Формат даты указан в операторе `SET DateFormat` скрипта загрузки данных. В вашей системе может быть установлен другой формат даты по умолчанию в зависимости от региональных настроек и других факторов. Можно изменить формат в примерах в соответствии с потребностями. Или можно изменить форматы в скрипте загрузки в соответствии с этими примерами.

Региональные настройки по умолчанию в приложениях основаны на системных региональных настройках компьютера или сервера, где установлено ПО Qlik Sense. Если на сервере Qlik Sense, к которому обращается пользователь, выбран шведский язык, то редактор загрузки данных будет использовать шведские региональные настройки для даты, времени и валюты. Эти параметры регионального формата не связаны с языком, отображаемым в интерфейсе пользователя Qlik Sense. Qlik Sense будет отображаться на языке, который используется в браузере.

Пример 1. День указан

Скрипт загрузки и результаты

Обзор

Откройте редактор загрузки данных и добавьте приведенный ниже скрипт загрузки на новую вкладку.

Скрипт загрузки содержит следующее:

- Набор данных, содержащий еженедельные общие продажи за 2022 год, в таблице под именем `Sales`.
- Даты транзакций, предоставленные в виде трех полей: `year`, `week` и `sales`.
- Предыдущая загрузка, используемая для создания меры `end_of_week` с помощью функции `makeweekdate()`, которая возвращает дату для пятницы этой недели в формате MM/DD/YYYY.

Чтобы подтвердить, что возвращаемая дата выпадает на пятницу, в функцию `weekday()` также помещается выражение `end_of_week`, которое отображает день недели.

Скрипт загрузки

```
SET DateFormat='MM/DD/YYYY';

Transactions:
    Load
        *,
        makeweekdate(transaction_year, transaction_week,4) as end_of_week,
        weekday(makeweekdate(transaction_year, transaction_week,4)) as week_day
        ;
Load * Inline [
transaction_year, transaction_week, sales
2022, 01, 10000
2022, 02, 11250
2022, 03, 9830
2022, 04, 14010
2022, 05, 28402
2022, 06, 9992
2022, 07, 7292
];
```

Результаты

Загрузите данные и откройте лист. Создайте новую таблицу и добавьте эти поля как измерения:

- `transaction_year`
- `transaction_week`
- `end_of_week`
- `week_day`

Результирующая таблица

transaction_year	transaction_week	end_of_week	week_day
2022	01	01/07/2022	Fri
2022	02	01/14/2022	Fri
2022	03	01/21/2022	Fri
2022	04	01/28/2022	Fri
2022	05	02/04/2022	Fri
2022	06	02/11/2022	Fri
2022	07	02/18/2022	Fri

Поле `end_of_week` создано в предыдущем операторе load с помощью функции `makeweekdate()`. Поля `transaction_year`, `transaction_week` передаются функции в качестве аргументов года и недели. В качестве аргумента дня используется значение 4.

Затем функция объединяет и преобразует эти значения в поле даты, возвращая результаты в формате системной переменной `DateFormat`.

Функция `makeweekdate()` и ее аргументы также помещаются в функцию `weekday()` с целью возвращения поля `week_day`. Как показано в таблице выше, поле `week_day` показывает, что эти даты выпадают на пятницу.

Пример 2. День не указан

Скрипт загрузки и результаты

Обзор

Откройте редактор загрузки данных и добавьте приведенный ниже скрипт загрузки на новую вкладку.

Скрипт загрузки содержит следующее:

- Набор данных, содержащий еженедельные продажи за 2022 год, в таблице под именем `Sales`.
- Даты транзакций, предоставленные в виде трех полей: `year`, `week` и `sales`.
- Предыдущая загрузка, которая используется для создания меры `first_day_of_week` с помощью функции `makeweekdate()`. Будет возвращена дата для понедельника этой недели в формате MM/DD/YYYY.

Чтобы подтвердить, что возвращаемая дата выпадает на понедельник, в функцию `weekday()` также помещается выражение `first_day_of_week`, которое отображает день недели.

Скрипт загрузки

```
SET DateFormat='MM/DD/YYYY';
SET FirstWeekDay=6;

Transactions:
    Load
        *,
        makeweekdate(transaction_year, transaction_week) as first_day_of_week,
        day(makeweekdate(transaction_year, transaction_week)) as week_day
        ;
Load * Inline [
transaction_year, transaction_week, sales
2022, 01, 10000
2022, 02, 11250
2022, 03, 9830
2022, 04, 14010
2022, 05, 28402
2022, 06, 9992
2022, 07, 7292
];
```

Результаты

Загрузите данные и откройте лист. Создайте новую таблицу и добавьте эти поля как измерения:

- `transaction_year`
- `transaction_week`
- `first_day_of_week`
- `week_day`

Результирующая таблица

transaction_year	transaction_week	first_day_of_week	week_day
2022	01	01/03/2022	Mon
2022	02	01/10/2022	Mon
2022	03	01/17/2022	Mon
2022	04	01/24/2022	Mon
2022	05	01/31/2022	Mon
2022	06	02/07/2022	Mon
2022	07	02/14/2022	Mon

Поле `first_day_of_week` создано в предыдущем операторе load с помощью функции `makeweekdate()`. Параметры `transaction_year` и `transaction_week` передаются в качестве аргументов функции, а параметр day остается пустым.

Затем функция объединяет и преобразует эти значения в поле даты, возвращая результаты в формате системной переменной `DateFormat`.

Функция `makeweekdate()` и ее аргументы также помещаются в функцию `weekday()`, в результате чего возвращается поле `week_day`. Как видно по таблице выше, поле `week_day` показывает, что эти данные выпадают на понедельник (хота переменная `FirstWeekDay` задает воскресенье в качестве первого дня недели), так как параметр day в функции `makeweekdate()` оставлен пустым.

Пример 3. Пример объекта диаграммы

Скрипт загрузки и выражение диаграммы

Обзор

Откройте редактор загрузки данных и добавьте приведенный ниже скрипт загрузки на новую вкладку.

Скрипт загрузки содержит следующее:

- Набор данных, содержащий еженедельные продажи за 2022 год, в таблице под именем `Sales`.
- Даты транзакций, предоставленные в виде трех полей: `year`, `week` и `sales`.

В этом примере объект диаграммы используется для создания меры, эквивалентной вычислению `end_of_week` из первого примера. Эта мера будет использовать функцию `makeweekdate()`, чтобы вернуть дату для пятницы этой недели в формате MM/DD/YYYY.

Для подтверждения того, что возвращаемая дата выпадает на пятницу, создается вторая мера, которая возвращает день недели.

Скрипт загрузки

```
SET DateFormat='MM/DD/YYYY';

Master_Calendar:
Load * Inline [
transaction_year, transaction_week, sales
2022, 01, 10000
2022, 02, 11250
2022, 03, 9830
2022, 04, 14010
2022, 05, 28402
2022, 06, 9992
2022, 07, 7292
];
```

Результаты

Выполните следующие действия.

1. Загрузите данные и откройте лист. Создайте новую таблицу и добавьте эти поля как измерения:
   - `transaction_year`
   - `transaction_week`
2. Чтобы выполнить вычисление, эквивалентное вычислению поля `end_of_week` из первого примера, создайте следующую меру:
   `=makeweekdate(transaction_year,transaction_week,4)`
3. Чтобы рассчитать день недели для каждой транзакции, создайте следующую меру:
   `=weekday(makeweekdate(transaction_year,transaction_week,4))`

Результирующая таблица

transaction_year	transaction_week	=makeweekdate (transaction_year,transaction_week,4)	=weekday(makeweekdate (transaction_year,transaction_week,4))
2022	01	01/07/2022	Fri
2022	02	01/14/2022	Fri
2022	03	01/21/2022	Fri
2022	04	01/28/2022	Fri
2022	05	02/04/2022	Fri
2022	06	02/11/2022	Fri
2022	07	02/18/2022	Fri

Поле, эквивалентное полю `end_of_week`, создается как мера в объекте диаграммы с помощью функции `makeweekdate()`. Поля `transaction_year` и `transaction_week` передаются функции в качестве аргументов года и недели. В качестве аргумента дня используется значение 4.

Затем функция объединяет и преобразует эти значения в поле даты, возвращая результаты в формате системной переменной `DateFormat`.

Функция `makeweekdate()` и ее аргументы также помещаются в функцию `weekday()`, в результате чего возвращается вычисление, эквивалентное вычислению поля `week_day` из первого примера. Как видно по таблице выше, в последнем столбце справа показано, что эти даты выпадают на пятницу.

Пример 4. Сценарий

Скрипт загрузки и выражение диаграммы

Обзор

В этом примере создайте список дат, содержащий все пятницы в 2022 году.

Откройте редактор загрузки данных и добавьте приведенный ниже скрипт загрузки на новую вкладку.

Скрипт загрузки

```
SET DateFormat='MM/DD/YYYY';

Calendar:
    load
           *,
        weekday(date) as weekday
    where year(date)=2022;
load
    makeweekdate(2022,recno()-2,4) as date
AutoGenerate 60;
```

Результаты

Результирующая таблица

date	weekday
01/07/2022	Fri
01/14/2022	Fri
01/21/2022	Fri
01/28/2022	Fri
02/04/2022	Fri
02/11/2022	Fri
02/18/2022	Fri
02/25/2022	Fri
03/04/2022	Fri
03/11/2022	Fri
03/18/2022	Fri
03/25/2022	Fri
04/01/2022	Fri

date	weekday
04/08/2022	Fri
04/15/2022	Fri
04/22/2022	Fri
04/29/2022	Fri
05/06/2022	Fri
05/13/2022	Fri
05/20/2022	Fri
05/27/2022	Fri
06/03/2022	Fri
06/10/2022	Fri
06/17/2022	Fri
+ 27 строк	

Функция `makeweekdate()` находит все пятницы в 2022 году. Использование параметра week = -2 позволяет предотвратить пропуски дат. В заключение предыдущая загрузка создает дополнительное поле `weekday` для уточнения, что каждое значение `date` попадает на пятницу.

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

Когда это следует использовать

Функция `minute()` полезна, когда требуется сравнить агрегирования по минутам. Например, эту функцию можно использовать, если требуется просмотреть распределения количества действий по минутам.

Эти измерения можно создать в скрипте загрузки с помощью функции создания поля в таблице основного календаря. Их также можно использовать непосредственно в диаграмме в качестве вычисляемых измерений.

Примеры функции

Пример	Результат
`minute ( '09:14:36' )`	Возвращает 14.
`minute ( '0.5555' )`	Возвращает 19 (так как 0,5555 = 13:19:55)

Региональные настройки

Если не указано иное, в примерах, приведенных в данном разделе, используется следующий формат даты: ММ/ДД/ГГГГ. Формат даты указан в операторе `SET DateFormat` скрипта загрузки данных. В вашей системе может быть установлен другой формат даты по умолчанию в зависимости от региональных настроек и других факторов. Можно изменить формат в примерах в соответствии с потребностями. Или можно изменить форматы в скрипте загрузки в соответствии с этими примерами.

Региональные настройки по умолчанию в приложениях основаны на системных региональных настройках компьютера или сервера, где установлено ПО Qlik Sense. Если на сервере Qlik Sense, к которому обращается пользователь, выбран шведский язык, то редактор загрузки данных будет использовать шведские региональные настройки для даты, времени и валюты. Эти параметры регионального формата не связаны с языком, отображаемым в интерфейсе пользователя Qlik Sense. Qlik Sense будет отображаться на языке, который используется в браузере.

Пример 1. Переменная (скрипт)

Скрипт загрузки и результаты

Обзор

Откройте редактор загрузки данных и добавьте приведенный ниже скрипт загрузки на новую вкладку.

Скрипт загрузки содержит следующее:

- Набор данных, содержащий транзакции по метке времени, загруженный в таблицу под именем `Transactions`.
- Используется системная переменная `TimeStamp` со значением по умолчанию `M/D/YYYY h:mm:ss[.fff] TT`.
- Создание поля `minute` для вычисления, когда совершены транзакции.

Скрипт загрузки

```
SET TimestampFormat='M/D/YYYY h:mm:ss[.fff] TT';

Transactions:
    Load
        *,
        minute(timestamp) as minute
        ;
Load
*
Inline
[
id,timestamp,amount
9497,'2022-01-05 19:04:57',47.25,
9498,'2022-01-03 14:21:53',51.75,
9499,'2022-01-03 05:40:49',73.53,
```

```
9500,'2022-01-04 18:49:38',15.35,
9501,'2022-01-01 22:10:22',31.43,
9502,'2022-01-05 19:34:46',13.24,
9503,'2022-01-04 22:58:34',74.34,
9504,'2022-01-06 11:29:38',50.00,
9505,'2022-01-02 08:35:54',36.34,
9506,'2022-01-06 08:49:09',74.23
];
```

Результаты

Загрузите данные и откройте лист. Создайте новую таблицу и добавьте эти поля как измерения:

- `timestamp`
- `minute`

Результирующая таблица

timestamp	minute
2022-01-01 22:10:22	10
2022-01-02 08:35:54	35
2022-01-03 05:40:49	40
2022-01-03 14:21:53	21
2022-01-04 18:49:38	49
2022-01-04 22:58:34	58
2022-01-05 19:04:57	4
2022-01-05 19:34:46	34
2022-01-06 08:49:09	49
2022-01-06 11:29:38	29

Значение в поле `minute` создаются с использованием функции `minute()` и путем передачи `timestamp` в качестве выражения в предшествующем операторе load.

Пример 2. Объект диаграммы (диаграмма)

Скрипт загрузки и выражение диаграммы

Обзор

Откройте редактор загрузки данных и добавьте приведенный ниже скрипт загрузки на новую вкладку.

Скрипт загрузки содержит следующее:

- Используется тот же набор данных и сценарий, что в первом примере.
- Используется системная переменная `TimeStamp` со значением по умолчанию `M/D/YYYY h:mm:ss[.fff] TT`.

Однако в этом примере в приложение загружается неизмененный набор данных. Значения `minute` рассчитывается с использованием меры в объекте диаграммы.

Скрипт загрузки

```
SET TimestampFormat='M/D/YYYY h:mm:ss[.fff] TT';

Transactions:
Load
*
Inline
[
id,timestamp,amount
9497,'2022-01-05 19:04:57',47.25,
9498,'2022-01-03 14:21:53',51.75,
9499,'2022-01-03 05:40:49',73.53,
9500,'2022-01-04 18:49:38',15.35,
9501,'2022-01-01 22:10:22',31.43,
9502,'2022-01-05 19:34:46',13.24,
9503,'2022-01-04 22:58:34',74.34,
9504,'2022-01-06 11:29:38',50.00,
9505,'2022-01-02 08:35:54',36.34,
9506,'2022-01-06 08:49:09',74.23
];
```

Результаты

Загрузите данные и откройте лист. Создайте новую таблицу и добавьте это поле как измерение: `timestamp`.

Создайте следующую меру:

```
=minute(timestamp)
```

Результирующая таблица

timestamp	minute
2022-01-01 22:10:22	10
2022-01-02 08:35:54	35
2022-01-03 05:40:49	40
2022-01-03 14:21:53	21
2022-01-04 18:49:38	49
2022-01-04 22:58:34	58

timestamp	minute
2022-01-05 19:04:57	4
2022-01-05 19:34:46	34
2022-01-06 08:49:09	49
2022-01-06 11:29:38	29

Значения для `minute` создаются с использованием функции `minute()`, где `timestamp` передается в виде выражения в мере для объекта диаграммы.

Пример 3. Сценарий

Скрипт загрузки и выражение диаграммы

Обзор

Откройте редактор загрузки данных и добавьте приведенный ниже скрипт загрузки на новую вкладку.

Скрипт загрузки содержит следующее:

- Набор данных с метками времени, который создается для представления записей проходов через турникет.
- Информация, содержащая каждый `timestamp` с соответствующим `id`, загруженная в таблицу под названием `Ticket_Barrier_Tracker`.
- Используется системная переменная `TimeStamp` со значением по умолчанию `M/D/YYYY h:mm:ss[.fff] TT`.

Пользователю нужен объект диаграммы, который по минутам отображает количество проходов через турникет.

Скрипт загрузки

```
SET TimestampFormat='M/D/YYYY h:mm:ss[.fff] TT';

tmpTimeStampCreator:
       load
               *
    where year(date)=2022;
load
       date(recno()+makedate(2021,12,31)) as date
AutoGenerate 1;

join load
       maketime(floor(rand()*24),floor(rand()*59),floor(rand()*59)) as time
autogenerate 10000;

Ticket_Barrier_Tracker:
load
```

```
       recno() as id,
       timestamp(date + time) as timestamp
resident tmpTimeStampCreator;

drop table tmpTimeStampCreator;
```

Результаты

Выполните следующие действия.

1. Загрузите данные и откройте лист. Создайте новую таблицу.
2. Создайте вычисляемое измерение, используя следующее выражение:
   `=minute(timestamp)`
3. Добавьте следующую меру агрегации для вычисления общего количества проходов:
   `=count(id)`
4. Задайте параметру меры **Формат чисел** значение **Денежный**.

Результирующая таблица

minute(timestamp)	=count(id)
0	174
1	171
2	175
3	165
4	188
5	176
6	158
7	187
8	178
9	178
10	197
11	161
12	166
13	184
14	159
15	161
16	152
17	160
18	176

minute(timestamp)	=count(id)
19	164
20	170
21	170
22	142
23	145
24	155
+ 35 строк	

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

Пример 1. Набор данных DateFormat (скрипт)

Скрипт загрузки и результаты

Обзор

Откройте Редактор загрузки данных и добавьте приведенный ниже скрипт загрузки на новую вкладку.

Скрипт загрузки содержит следующее:

- Набор данных с датами под именем `Master_Calendar`. Системной переменной `DateFormat` задано значение DD/MM/YYYY.
- Предшествующая загрузка, создающая дополнительное поле под именем `month_name` с использованием функции `month()`.
- Дополнительное поле под именем `long_date` с использованием функции `date()` для выражения даты в полном формате.

Скрипт загрузки

```
SET DateFormat='DD/MM/YYYY';

Master_Calendar:
Load
        date,
        date(date,'dd-MMMM-YYYY') as long_date,
        month(date) as month_name
Inline
[
date
03/01/2022
03/02/2022
03/03/2022
03/04/2022
03/05/2022
03/06/2022
03/07/2022
03/08/2022
03/09/2022
03/10/2022
03/11/2022
];
```

Результаты

Загрузите данные и откройте лист. Создайте новую таблицу и добавьте эти поля как измерения:

- `date`
- `long_date`
- `month_name`

Результирующая таблица

date	long_date	monthname
03/01/2022	03-January- 2022	Jan
03/02/2022	03-February- 2022	Feb
03/03/2022	03-March- 2022	Mar
03/04/2022	03-April- 2022	Apr

date	long_date	monthname
03/05/2022	03-May- 2022	May
03/06/2022	03-June- 2022	Jun
03/07/2022	03-July- 2022	Jul
03/08/2022	03-August- 2022	Aug
03/09/2022	03-September- 2022	Sep
03/10/2022	03-October- 2022	Oct
03/11/2022	03-November- 2022	Nov

Имя месяца правильно вычисляется функцией `month()` в скрипте.

Пример 2. Даты ANSI (скрипт)

Скрипт загрузки и результаты

Обзор

Откройте Редактор загрузки данных и добавьте приведенный ниже скрипт загрузки на новую вкладку.

Скрипт загрузки содержит следующее:

- Набор данных с датами под именем `Master_Calendar`. Используется системная переменная DateFormat DD/MM/YYYY. Однако даты, включенные в набор данных, имеют формат стандарта ANSI.
- Предшествующая загрузка, создающая дополнительное поле под именем `month_name` с использованием функции month().
- Дополнительное поле под именем `long_date` с использованием функции date() для выражения даты в полном формате.

Скрипт загрузки

```
SET DateFormat='DD/MM/YYYY';
Master_Calendar:
Load
       date,
       date(date,'dd-MMMM-YYYY') as long_date,
       month(date) as month_name

Inline
[
date
2022-01-11
2022-02-12
2022-03-13
2022-04-14
```

```
2022-05-15
2022-06-16
2022-07-17
2022-08-18
2022-09-19
2022-10-20
2022-11-21
];
```

Результаты

Загрузите данные и откройте лист. Создайте новую таблицу и добавьте эти поля как измерения:

- `date`
- `long_date`
- `month_name`

Результирующая таблица

date	long_date	month_name
03/11/2022	11-March- 2022	11
03/12/2022	12-March- 2022	12
03/13/2022	13-March- 2022	13
03/14/2022	14-March- 2022	14
03/15/2022	15-March- 2022	15
03/16/2022	16-March- 2022	16
03/17/2022	17-March- 2022	17
03/18/2022	18-March- 2022	18
03/19/2022	19-March- 2022	19
03/20/2022	20-March- 2022	20
03/21/2022	21-March- 2022	21

Имя месяца правильно вычисляется функцией `month()` в скрипте.

Пример 3. Неформатированные даты (скрипт)

Скрипт загрузки и результаты

Обзор

Откройте Редактор загрузки данных и добавьте приведенный ниже скрипт загрузки на новую вкладку.

Скрипт загрузки содержит следующее:

- Набор данных с датами под именем `Master_Calendar`. Используется системная переменная `DateFormat` DD/MM/YYYY.
- Предшествующая загрузка, создающая дополнительное поле под именем `month_name` с использованием функции `month()` .
- Первоначальная дата без форматирования под именем `unformatted_date`.
- Дополнительное поле под именем `long_date` с использованием функции `date()` для выражения даты в полном формате.

Скрипт загрузки

```
SET DateFormat='DD/MM/YYYY';

Master_Calendar:
Load
      unformatted_date,
      date(unformatted_date,'dd-MMMM-YYYY') as long_date,
      month(unformatted_date) as month_name

Inline
[
unformatted_date
44868
44898
44928
44958
44988
45018
45048
45078
45008
45038
45068
];
```

Результаты

Загрузите данные и откройте лист. Создайте новую таблицу и добавьте эти поля как измерения:

- `unformatted_date`
- `long_date`
- `month_name`

Результирующая таблица

unformatted_date	long_date	month_name
44868	03-January- 2022	Jan
44898	03-February- 2022	Feb

unformatted_date	long_date	month_name
44928	03-March- 2022	Mar
44958	03-April- 2022	Apr
44988	03-May- 2022	May
45018	03-June- 2022	Jun
45048	03-July- 2022	Jul
45078	03-August- 2022	Aug
45008	03-September- 2022	Sep
45038	03-October- 2022	Oct
45068	03-November- 2022	Nov

Имя месяца правильно вычисляется функцией `month()` в скрипте.

Пример 4. Расчет месяца окончания срока действия

Скрипт загрузки и выражение диаграммы

Обзор

Откройте Редактор загрузки данных и добавьте приведенный ниже скрипт загрузки на новую вкладку.

Скрипт загрузки содержит следующее:

- Набор данных с заказами, размещенными в марте, под именем `Subscriptions`. Данная таблица содержит слишком много полей.
    - id
    - order_date
    - amount

Скрипт загрузки

```
Subscriptions:
Load
        id,
        order_date,
        amount
Inline
[
id,order_date,amount
1,03/01/2022,231.24
2,03/02/2022,567.28
3,03/03/2022,364.28
```

```
4,03/04/2022,575.76
5,03/05/2022,638.68
6,03/06/2022,785.38
7,03/07/2022,967.46
8,03/08/2022,287.67
9,03/09/2022,764.45
10,03/10/2022,875.43
11,03/11/2022,957.35
];
```

Результаты

Загрузите данные и откройте лист. Создайте новую таблицу и добавьте это поле как измерение: `order_date`.

Чтобы рассчитать месяц, когда истекает срок действия заказа, создайте эту меру: `=month(order_date+180)`.

Результирующая таблица

order_date	=month(order_date+180)
03/01/2022	Jul
03/02/2022	Aug
03/03/2022	Aug
03/04/2022	Sep
03/05/2022	Oct
03/06/2022	Nov
03/07/2022	Dec
03/08/2022	Jan
03/09/2022	Mar
03/10/2022	Apr
03/11/2022	May

Функция `month()` правильно определяет, что срок действия заказа, размещенного 11 марта, истекает в июле.

monthend

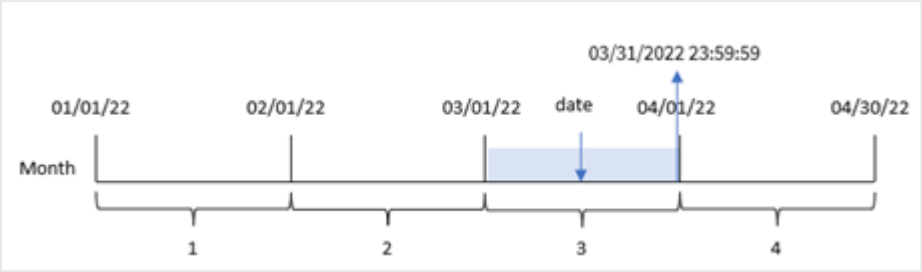
Эта функция возвращает значение, соответствующее метке времени, включающей последнюю миллисекунду последнего дня месяца, содержащего значение, указанное в поле `date`. По умолчанию для вывода используется формат `DateFormat`, установленный в скрипте.

Синтаксис:

```
MonthEnd(date[, period_no])
```

Другими словами, функция `monthend()` определяет, на какой месяц приходится дата. Затем она возвращает метку времени в формате даты для последней миллисекунды этого месяца.

Диаграмма функции `monthend`.



Когда это следует использовать

Функция `monthend()` используется в составе выражения, когда в расчетах требуется учитывать часть месяца, которая еще не прошла, например, если требуется рассчитать общую сумму процентов, еще не начисленных в течение месяца.

Возвращаемые типы данных: двойное значение

Аргументы

Аргумент	Описание
date	Дата или метка времени 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.

Региональные настройки

Если не указано иное, в примерах, приведенных в данном разделе, используется следующий формат даты: ММ/ДД/ГГГГ. Формат даты указан в операторе `SET DateFormat` скрипта загрузки данных. В вашей системе может быть установлен другой формат даты по умолчанию в зависимости от региональных настроек и других факторов. Можно изменить формат в примерах в соответствии с потребностями. Или можно изменить форматы в скрипте загрузки в соответствии с этими примерами.

Региональные настройки по умолчанию в приложениях основаны на системных региональных настройках компьютера или сервера, где установлено ПО Qlik Sense. Если на сервере Qlik Sense, к которому обращается пользователь, выбран шведский язык, то редактор загрузки данных будет использовать шведские региональные настройки для даты, времени и валюты. Эти параметры регионального формата не связаны с языком, отображаемым в интерфейсе пользователя Qlik Sense. Qlik Sense будет отображаться на языке, который используется в браузере.

Примеры функции

Пример	Результат
`monthend('02/19/2012')`	Возвращает 02/29/2012 23:59:59.
`monthend('02/19/2001', -1)`	Возвращает 01/31/2001 23:59:59.

Пример 1. Базовый пример

Скрипт загрузки и результаты

Обзор

Откройте редактор загрузки данных и добавьте приведенный ниже скрипт загрузки на новую вкладку.

Скрипт загрузки содержит следующее:

- Набор данных, содержащий набор транзакций за 2022 год, загруженный в таблицу под именем `Transactions`.
- Поле даты, предоставленное в формате `MM/DD/YYYY` системной переменной `DateFormat`.
- Предыдущий оператор load, который содержит следующее:
  - Функция `monthend()`, заданная как поле `end_of_month`.
  - Функция `timestamp`, заданная как поле `end_of_month_timestamp`.

Скрипт загрузки

```
SET DateFormat='MM/DD/YYYY';

Transactions:
    Load
    *,
    monthend(date) as end_of_month,
    timestamp(monthend(date)) as end_of_month_timestamp
    ;
Load
*
Inline
[
id,date,amount
8188,1/7/2022,17.17
8189,1/19/2022,37.23
8190,2/28/2022,88.27
8191,2/5/2022,57.42
8192,3/16/2022,53.80
8193,4/1/2022,82.06
8194,5/7/2022,40.39
8195,5/16/2022,87.21
8196,6/15/2022,95.93
8197,6/26/2022,45.89
8198,7/9/2022,36.23
```

```
8199,7/22/2022,25.66
8200,7/23/2022,82.77
8201,7/27/2022,69.98
8202,8/2/2022,76.11
8203,8/8/2022,25.12
8204,8/19/2022,46.23
8205,9/26/2022,84.21
8206,10/14/2022,96.24
8207,10/29/2022,67.67
];
```

Результаты

Загрузите данные и откройте лист. Создайте новую таблицу и добавьте эти поля как измерения:

- id
- date
- end_of_month
- end_of_month_timestamp

Результирующая таблица

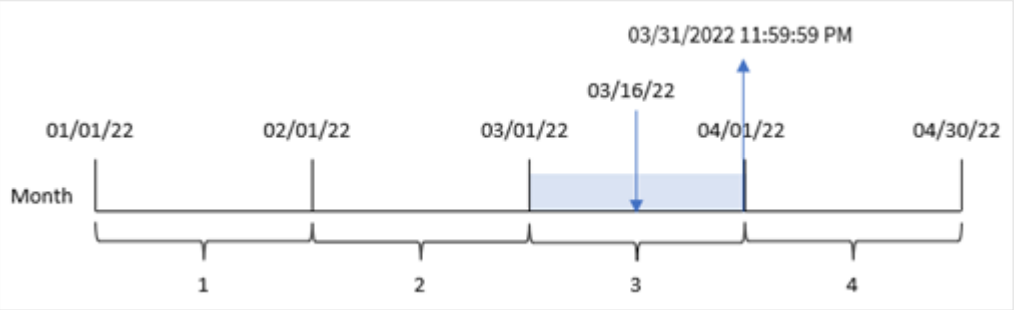
id	date	end_of_month	end_of_month_timestamp
8188	1/7/2022	01/31/2022	1/31/2022 11:59:59 PM
8189	1/19/2022	01/31/2022	1/31/2022 11:59:59 PM
8190	2/5/2022	02/28/2022	2/28/2022 11:59:59 PM
8191	2/28/2022	02/28/2022	2/28/2022 11:59:59 PM
8192	3/16/2022	03/31/2022	3/31/2022 11:59:59 PM
8193	4/1/2022	04/30/2022	4/30/2022 11:59:59 PM
8194	5/7/2022	05/31/2022	5/31/2022 11:59:59 PM
8195	5/16/2022	05/31/2022	5/31/2022 11:59:59 PM
8196	6/15/2022	06/30/2022	6/30/2022 11:59:59 PM
8197	6/26/2022	06/30/2022	6/30/2022 11:59:59 PM
8198	7/9/2022	07/31/2022	7/31/2022 11:59:59 PM
8199	7/22/2022	07/31/2022	7/31/2022 11:59:59 PM
8200	7/23/2022	07/31/2022	7/31/2022 11:59:59 PM
8201	7/27/2022	07/31/2022	7/31/2022 11:59:59 PM
8202	8/2/2022	08/31/2022	8/31/2022 11:59:59 PM
8203	8/8/2022	08/31/2022	8/31/2022 11:59:59 PM
8204	8/19/2022	08/31/2022	8/31/2022 11:59:59 PM

id	date	end_of_month	end_of_month_timestamp
8205	9/26/2022	09/30/2022	9/30/2022 11:59:59 PM
8206	10/14/2022	10/31/2022	10/31/2022 11:59:59 PM
8207	10/29/2022	10/31/2022	10/31/2022 11:59:59 PM

Поле `end_of_month` создано предшествующим оператором load с использованием функции `monthend()`, где в качестве аргумента функции передано поле даты.

Функция `monthend()` определяет, к какому месяцу относится значение даты, и возвращает метку времени для последней миллисекунды этого месяца.

Диаграмма функции `monthend`, выбранный месяц — март.



Транзакция 8192 совершена 16 марта. Функция `monthend()` возвращает последнюю миллисекунду этого месяца, то есть 23:59:59 (11:59:59 PM) 31 марта.

Пример 2. Скрипт period_no

Скрипт загрузки и результаты

Обзор

Используется тот же набор данных и сценарий, что в первом примере.

В этом примере стоит задача создать поле `previous_month_end`, которое возвращает метку времени окончания месяца, предшествующего совершению транзакции.

Скрипт загрузки

```
SET DateFormat='MM/DD/YYYY';

Transactions:
    Load
    *,
    monthend(date,-1) as previous_month_end,
    timestamp(monthend(date,-1)) as previous_month_end_timestamp
    ;
Load
*
Inline
```

```
[
id,date,amount
8188,1/7/2022,17.17
8189,1/19/2022,37.23
8190,2/28/2022,88.27
8191,2/5/2022,57.42
8192,3/16/2022,53.80
8193,4/1/2022,82.06
8194,5/7/2022,40.39
8195,5/16/2022,87.21
8196,6/15/2022,95.93
8197,6/26/2022,45.89
8198,7/9/2022,36.23
8199,7/22/2022,25.66
8200,7/23/2022,82.77
8201,7/27/2022,69.98
8202,8/2/2022,76.11
8203,8/8/2022,25.12
8204,8/19/2022,46.23
8205,9/26/2022,84.21
8206,10/14/2022,96.24
8207,10/29/2022,67.67
];
```

Результаты

Загрузите данные и откройте лист. Создайте новую таблицу и добавьте эти поля как измерения:

- `id`
- `date`
- `previous_month_end`
- `previous_month_end_timestamp`

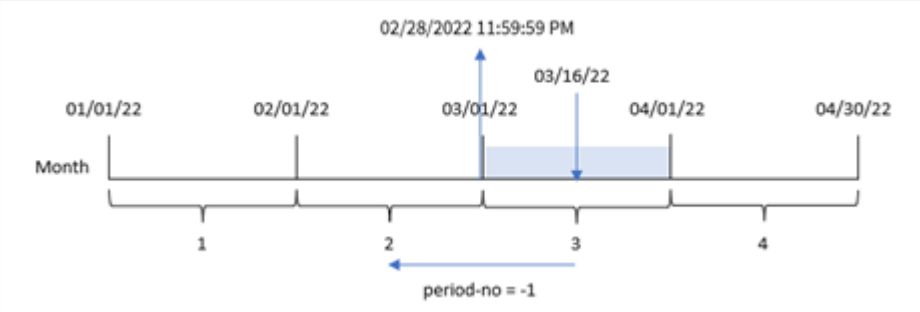
Результирующая таблица

id	date	previous_month_end	previous_month_end_timestamp
8188	1/7/2022	12/31/2021	12/31/2021 11:59:59 PM
8189	1/19/2022	12/31/2021	12/31/2021 11:59:59 PM
8190	2/5/2022	01/31/2022	1/31/2022 11:59:59 PM
8191	2/28/2022	01/31/2022	1/31/2022 11:59:59 PM
8192	3/16/2022	02/28/2022	2/28/2022 11:59:59 PM
8193	4/1/2022	03/31/2022	3/31/2022 11:59:59 PM
8194	5/7/2022	04/30/2022	4/30/2022 11:59:59 PM
8195	5/16/2022	04/30/2022	4/30/2022 11:59:59 PM
8196	6/15/2022	05/31/2022	5/31/2022 11:59:59 PM

id	date	previous_month_end	previous_month_end_timestamp
8197	6/26/2022	05/31/2022	5/31/2022 11:59:59 PM
8198	7/9/2022	06/30/2022	6/30/2022 11:59:59 PM
8199	7/22/2022	06/30/2022	6/30/2022 11:59:59 PM
8200	7/23/2022	06/30/2022	6/30/2022 11:59:59 PM
8201	7/27/2022	06/30/2022	6/30/2022 11:59:59 PM
8202	8/2/2022	07/31/2022	7/31/2022 11:59:59 PM
8203	8/8/2022	07/31/2022	7/31/2022 11:59:59 PM
8204	8/19/2022	07/31/2022	7/31/2022 11:59:59 PM
8205	9/26/2022	08/31/2022	8/31/2022 11:59:59 PM
8206	10/14/2022	09/30/2022	9/30/2022 11:59:59 PM
8207	10/29/2022	09/30/2022	9/30/2022 11:59:59 PM

Функция `monthend()` сначала определяет месяц, в течение которого совершены транзакции, так как используется аргумент сдвига `period_no` = `-1`. Затем она возвращается на месяц назад и определяет последнюю миллисекунду предыдущего месяца.

Диаграмма функции `monthend` с переменной period_no.



Транзакция 8192 совершена 16 марта. Функция `monthend()` определяет, что месяцем, предшествующим транзакции, является февраль. Затем она возвращает последнюю миллисекунду этого месяца, 23:59:59 (11:59:59 PM) 28 февраля.

Пример 3. Пример диаграммы

Скрипт загрузки и выражение диаграммы

Обзор

Используется тот же набор данных и сценарий, что в первом примере.

В этом примере в приложение загружается неизмененный набор данных. Поставлена задача создать расчет, который возвращает метку времени для конца месяца, в течение которого совершена транзакция, в качестве меры в диаграмме приложения.

Скрипт загрузки

```
Transactions:
Load
*
Inline
[
id,date,amount
8188,1/7/2022,17.17
8189,1/19/2022,37.23
8190,2/28/2022,88.27
8191,2/5/2022,57.42
8192,3/16/2022,53.80
8193,4/1/2022,82.06
8194,5/7/2022,40.39
8195,5/16/2022,87.21
8196,6/15/2022,95.93
8197,6/26/2022,45.89
8198,7/9/2022,36.23
8199,7/22/2022,25.66
8200,7/23/2022,82.77
8201,7/27/2022,69.98
8202,8/2/2022,76.11
8203,8/8/2022,25.12
8204,8/19/2022,46.23
8205,9/26/2022,84.21
8206,10/14/2022,96.24
8207,10/29/2022,67.67
];
```

Результаты

Загрузите данные и откройте лист. Создайте новую таблицу и добавьте эти поля как измерения:

- `date`
- `id`

Чтобы вычислить дату окончания месяца, в течение которого была совершена транзакция, создайте следующие меры:

- `=monthend(date)`
- `=timestamp(monthend(date))`

Результирующая таблица

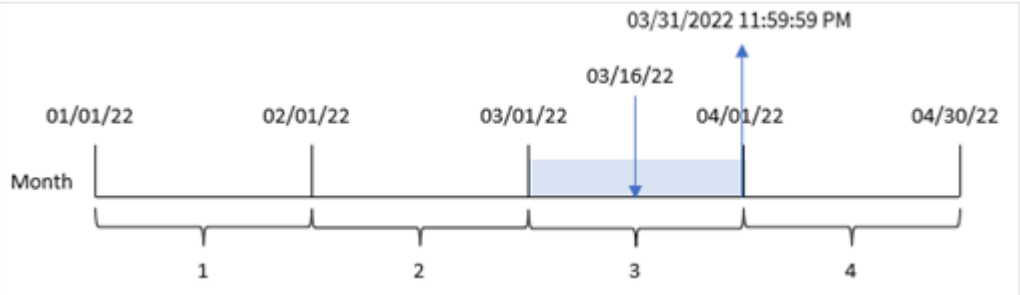
id	date	=monthend(date)	=timestamp(monthend(date))
8188	10/14/2022	10/31/2022	10/31/2022 11:59:59 PM

id	date	=monthend(date)	=timestamp(monthend(date))
8189	10/29/2022	10/31/2022	10/31/2022 11:59:59 PM
8190	9/26/2022	09/30/2022	9/30/2022 11:59:59 PM
8191	8/2/2022	08/31/2022	8/31/2022 11:59:59 PM
8192	8/8/2022	08/31/2022	8/31/2022 11:59:59 PM
8193	8/19/2022	08/31/2022	8/31/2022 11:59:59 PM
8194	7/9/2022	07/31/2022	7/31/2022 11:59:59 PM
8195	7/22/2022	07/31/2022	7/31/2022 11:59:59 PM
8196	7/23/2022	07/31/2022	7/31/2022 11:59:59 PM
8197	7/27/2022	07/31/2022	7/31/2022 11:59:59 PM
8198	6/15/2022	06/30/2022	6/30/2022 11:59:59 PM
8199	6/26/2022	06/30/2022	6/30/2022 11:59:59 PM
8200	5/7/2022	05/31/2022	5/31/2022 11:59:59 PM
8201	5/16/2022	05/31/2022	5/31/2022 11:59:59 PM
8202	4/1/2022	04/30/2022	4/30/2022 11:59:59 PM
8203	3/16/2022	03/31/2022	3/31/2022 11:59:59 PM
8204	2/5/2022	02/28/2022	2/28/2022 11:59:59 PM
8205	2/28/2022	02/28/2022	2/28/2022 11:59:59 PM
8206	1/7/2022	01/31/2022	1/31/2022 11:59:59 PM
8207	1/19/2022	01/31/2022	1/31/2022 11:59:59 PM

Мера `end_of_month` создана в диаграмме с использованием функции `monthend()`, где в качестве аргумента функции передано поле даты.

Функция `monthend()` определяет, к какому месяцу относится значение даты, и возвращает метку времени для последней миллисекунды этого месяца.

Диаграмма функции `monthend` с переменной period_no.



Транзакция 8192 совершена 16 марта. Функция `monthend()` возвращает последнюю миллисекунду этого месяца, то есть 23:59:59 (11:59:59 PM) 31 марта.

Пример 4. Сценарий

Скрипт загрузки и результаты

Обзор

В этом примере набор данных загружается в таблицу под именем `Employee_Expenses`. Данная таблица содержит следующие поля:

- Employee IDs (Идентификаторы сотрудников)
- Employee names (Имена сотрудников)
- Средние ежедневные заявки на возмещение расходов каждого сотрудника.

Конечному пользователю требуется получить диаграмму, которая по идентификатору и имени сотрудника отображает ожидаемые расходы в течение оставшейся части месяца.

Скрипт загрузки

```
Employee_Expenses:
Load
*
Inline
[
employee_id,employee_name,avg_daily_claim
182,Mark, $15
183,Deryck, $12.5
184,Dexter, $12.5
185,Sydney,$27
186,Agatha,$18
];
```

Результаты

Загрузите данные и откройте лист. Создайте новую таблицу и добавьте эти поля как измерения:

- `employee_id`
- `employee_name`

Чтобы рассчитать накопленный процент, создайте эту меру:

```
=floor(monthend(today(1),0)-today(1))*avg_daily_claim
```

Эта мера является динамической и дает разные результаты в таблице в зависимости от даты загрузки данных.

Задайте параметру **Формат чисел** меры значение **Денежный**.

Результирующая таблица

employee_id	employee_name	=floor(monthend(today(1),0)-today(1))*avg_daily_claim
182	Mark	$30.00
183	Deryck	$25.00
184	Dexter	$25.00
185	Sydney	$54.00
186	Agatha	$36.00

Функция `monthend()` возвращает дату окончания текущего месяца, используя сегодняшнюю дату в качестве единственного аргумента. Выражение возвращает количество дней, оставшихся в этом месяце, вычитая сегодняшнюю дату из даты окончания месяца.

Затем это значение умножается на среднюю ежедневную заявку на возмещение расходов каждого сотрудника для расчета оценочной суммы заявок, которые каждый сотрудник должен подать до конца месяца.

monthname

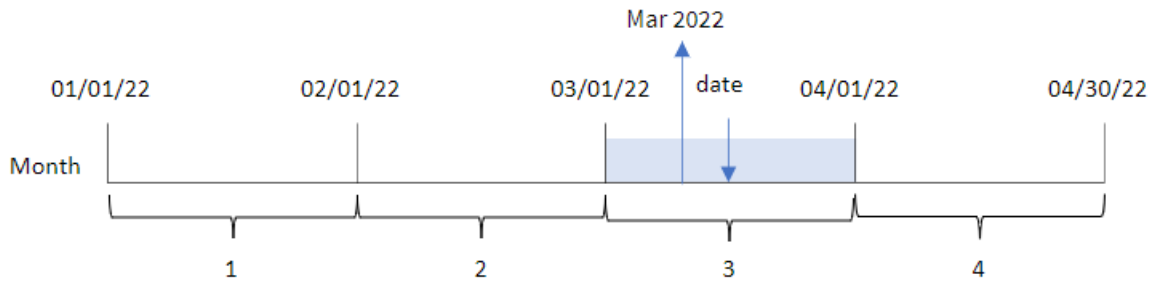
Эта функция возвращает значение, отображающее месяц (в формате переменной **MonthNames** скрипта) и год с базовым числовым значением, соответствующим метке времени, включающей первую миллисекунду первого дня указанного месяца.

Синтаксис:

```
MonthName(date[, period_no])
```

Возвращаемые типы данных: двойное значение

Диаграмма функции monthname



Аргументы

Аргумент	Описание
date	Дата или метка времени для вычисления.

Аргумент	Описание
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 функции

Пример	Результат
`monthname('10/19/2013')`	Возвращает Oct 2013
`monthname('10/19/2013', -1)`	Возвращает Sep 2013

Региональные настройки

Если не указано иное, в примерах, приведенных в данном разделе, используется следующий формат даты: ММ/ДД/ГГГГ. Формат даты указан в операторе `SET DateFormat` скрипта загрузки данных. В вашей системе может быть установлен другой формат даты по умолчанию в зависимости от региональных настроек и других факторов. Можно изменить формат в примерах в соответствии с потребностями. Или можно изменить форматы в скрипте загрузки в соответствии с этими примерами.

Региональные настройки по умолчанию в приложениях основаны на системных региональных настройках компьютера или сервера, где установлено ПО Qlik Sense. Если на сервере Qlik Sense, к которому обращается пользователь, выбран шведский язык, то редактор загрузки данных будет использовать шведские региональные настройки для даты, времени и валюты. Эти параметры регионального формата не связаны с языком, отображаемым в интерфейсе пользователя Qlik Sense. Qlik Sense будет отображаться на языке, который используется в браузере.

Пример 1. Базовый пример

Скрипт загрузки и результаты

Обзор

Откройте редактор загрузки данных и добавьте приведенный ниже скрипт загрузки на новую вкладку.

Скрипт загрузки содержит следующее:

- Набор данных, содержащий набор транзакций за 2022 год, загружается в таблицу под именем `Transactions`.
- Поле даты было предоставлено в формате системной переменной `DateFormat` (MM/DD/YYYY).
- Создание поля `transaction_month`, которое возвращает месяц, в котором осуществлялись транзакции.

Скрипт загрузки

```
SET DateFormat='MM/DD/YYYY';
SET MonthNames='Jan;Feb;Mar;Apr;May;Jun;Jul;Aug;Sep;Oct;Nov;Dec';

Transactions:
    Load
        *,
        monthname(date) as transaction_month
        ;
Load
*
Inline
[
id,date,amount
8188,1/7/2022,17.17
8189,1/19/2022,37.23
8190,2/28/2022,88.27
8191,2/5/2022,57.42
8192,3/16/2022,53.80
8193,4/1/2022,82.06
8194,5/7/2022,40.39
8195,5/16/2022,87.21
8196,6/15/2022,95.93
8197,6/26/2022,45.89
8198,7/9/2022,36.23
8199,7/22/2022,25.66
8200,7/23/2022,82.77
8201,7/27/2022,69.98
8202,8/2/2022,76.11
8203,8/8/2022,25.12
8204,8/19/2022,46.23
8205,9/26/2022,84.21
8206,10/14/2022,96.24
8207,10/29/2022,67.67
];
```

Результаты

Загрузите данные и откройте лист. Создайте новую таблицу и добавьте эти поля как измерения:

- `date`
- `transaction_month`

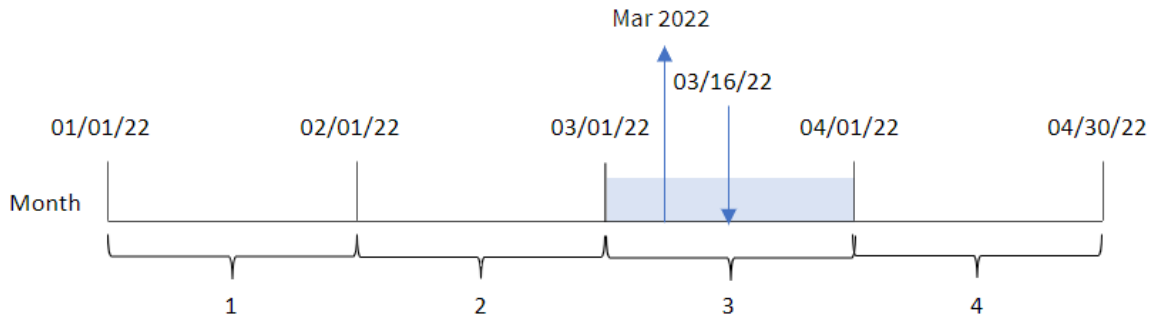
Результирующая таблица

date	transaction_month
1/7/2022	Jan 2022
1/19/2022	Jan 2022
2/5/2022	Feb 2022

date	transaction_month
2/28/2022	Feb 2022
3/16/2022	Mar 2022
4/1/2022	Apr 2022
5/7/2022	May 2022
5/16/2022	May 2022
6/15/2022	Jun 2022
6/26/2022	Jun 2022
7/9/2022	Jul 2022
7/22/2022	Jul 2022
7/23/2022	Jul 2022
7/27/2022	Jul 2022
8/2/2022	Aug 2022
8/8/2022	Aug 2022
8/19/2022	Aug 2022
9/26/2022	Sep 2022
10/14/2022	Oct 2022
10/29/2022	Oct 2022

Поле «`transaction_month`» создано предшествующим оператором load с использованием функции `monthname()`, где в качестве аргумента функции передано поле `date`.

Диаграмма функции monthname, базовый пример



Функция `monthname()` определяет, что транзакция 8192 произошла в марте 2022 года, и возвращает это значение с помощью системной переменной `MonthNames`.

Пример 2. Скрипт period_no

Скрипт загрузки и результаты

Обзор

Откройте редактор загрузки данных и добавьте приведенный ниже скрипт загрузки на новую вкладку.

Скрипт загрузки содержит следующее:

- Используется тот же встроенный набор данных и сценарий, что в первом примере.
- Создание поля `transaction_previous_month`, которое возвращает метку времени окончания месяца, предшествующего совершению транзакции.

Скрипт загрузки

```
SET DateFormat='MM/DD/YYYY';
SET MonthNames='Jan;Feb;Mar;Apr;May;Jun;Jul;Aug;Sep;Oct;Nov;Dec';

Transactions:
    Load
        *,
        monthname(date,-1) as transaction_previous_month
        ;
Load
*
Inline
[
id,date,amount
8188,1/7/2022,17.17
8189,1/19/2022,37.23
8190,2/28/2022,88.27
8191,2/5/2022,57.42
8192,3/16/2022,53.80
8193,4/1/2022,82.06
8194,5/7/2022,40.39
8195,5/16/2022,87.21
8196,6/15/2022,95.93
8197,6/26/2022,45.89
8198,7/9/2022,36.23
8199,7/22/2022,25.66
8200,7/23/2022,82.77
8201,7/27/2022,69.98
8202,8/2/2022,76.11
8203,8/8/2022,25.12
8204,8/19/2022,46.23
8205,9/26/2022,84.21
8206,10/14/2022,96.24
8207,10/29/2022,67.67
];
```

Результаты

Загрузите данные и откройте лист. Создайте новую таблицу и добавьте эти поля как измерения:

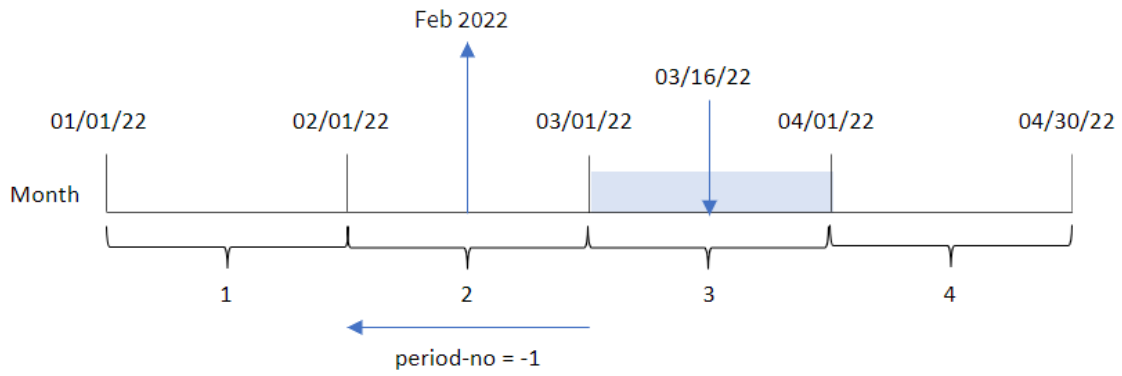
- `date`
- `transaction_previous_month`

Результирующая таблица

date	transaction_previous_month
1/7/2022	Dec 2021
1/19/2022	Dec 2021
2/5/2022	Jan 2022
2/28/2022	Jan 2022
3/16/2022	Feb 2022
4/1/2022	Mar 2022
5/7/2022	Apr 2022
5/16/2022	Apr 2022
6/15/2022	May 2022
6/26/2022	May 2022
7/9/2022	Jun 2022
7/22/2022	Jun 2022
7/23/2022	Jun 2022
7/27/2022	Jun 2022
8/2/2022	Jul 2022
8/8/2022	Jul 2022
8/19/2022	Jul 2022
9/26/2022	Aug 2022
10/14/2022	Sep 2022
10/29/2022	Sep 2022

В этом случае, так как в качестве аргумента смещения в функции `monthname()` использовалось `period_no` = -1, функция сначала определяет месяц, в котором совершаются транзакции. Затем она переходит на один месяц назад и возвращает название месяца и год.

Диаграмма функции monthname, пример с аргументом period_no



Транзакция 8192 совершена 16 марта. Функция `monthname()` определяет, что месяцем, предшествующим транзакции, был февраль, и возвращает месяц в формате системной переменной `MonthNames`, а также год 2022.

Пример 3. Пример объекта диаграммы

Скрипт загрузки и выражение диаграммы

Обзор

Откройте редактор загрузки данных и добавьте приведенный ниже скрипт загрузки на новую вкладку.

Скрипт загрузки содержит тот же встроенный набор данных и сценарий, что в первом примере. Однако в этом примере в приложение загружается неизмененный набор данных. Расчет, который возвращает метку времени окончания месяца, в котором совершены транзакции, создается как мера в объекте диаграммы в приложении.

Скрипт загрузки

```
SET DateFormat='MM/DD/YYYY';
SET MonthNames='Jan;Feb;Mar;Apr;May;Jun;Jul;Aug;Sep;Oct;Nov;Dec';

Transactions:
Load
*
Inline
[
id,date,amount
8188,1/7/2022,17.17
8189,1/19/2022,37.23
8190,2/28/2022,88.27
8191,2/5/2022,57.42
8192,3/16/2022,53.80
8193,4/1/2022,82.06
8194,5/7/2022,40.39
```

```
8195,5/16/2022,87.21
8196,6/15/2022,95.93
8197,6/26/2022,45.89
8198,7/9/2022,36.23
8199,7/22/2022,25.66
8200,7/23/2022,82.77
8201,7/27/2022,69.98
8202,8/2/2022,76.11
8203,8/8/2022,25.12
8204,8/19/2022,46.23
8205,9/26/2022,84.21
8206,10/14/2022,96.24
8207,10/29/2022,67.67
];
```

Результаты

Загрузите данные и откройте лист. Создайте новую таблицу и добавьте это поле как измерение:`date`.

Создайте следующую меру:

```
=monthname(date)
```

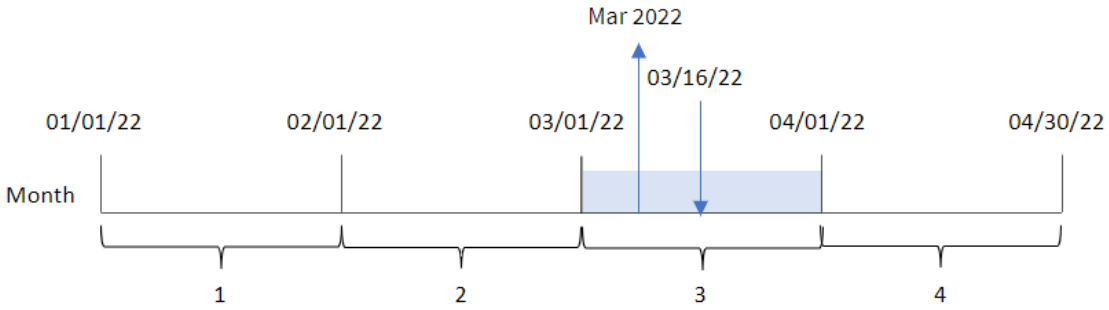
Результирующая таблица

date	=monthname(date)
1/7/2022	Jan 2022
1/19/2022	Jan 2022
2/5/2022	Feb 2022
2/28/2022	Feb 2022
3/16/2022	Mar 2022
4/1/2022	Apr 2022
5/7/2022	May 2022
5/16/2022	May 2022
6/15/2022	Jun 2022
6/26/2022	Jun 2022
7/9/2022	Jul 2022
7/22/2022	Jul 2022
7/23/2022	Jul 2022
7/27/2022	Jul 2022
8/2/2022	Aug 2022

date	=monthname(date)
8/8/2022	Aug 2022
8/19/2022	Aug 2022
9/26/2022	Sep 2022
10/14/2022	Oct 2022
10/29/2022	Oct 2022

Мера `month_name` создана в объекте диаграммы с использованием функции `monthname()`, где в качестве аргумента функции передано поле `date`.

Диаграмма функции monthname, пример с объектом диаграммы



Функция `monthname()` определяет, что транзакция 8192 произошла в марте 2022 года, и возвращает это значение с помощью системной переменной `MonthNames`.

monthsend

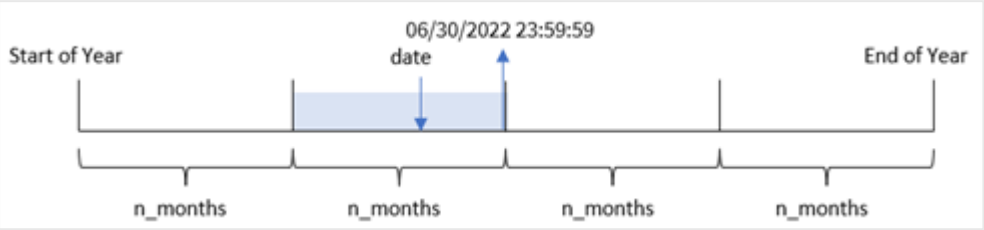
Эта функция возвращает значение, соответствующее метке времени последней миллисекунды месяца, двухмесячного периода, квартала, трети года (четыре месяца) или полугодия, содержащих базовую дату. Также можно найти метку времени окончания для предыдущего или последующего временного периода. По умолчанию для вывода используется формат `DateFormat`, установленный в скрипте.

Синтаксис:

```
MonthsEnd(n_months, date[, period_no [, first_month_of_year]])
```

Возвращаемые типы данных: двойное значение

Диаграмма функции `monthsend`.



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 (четыре месяца) или 6 (полугодие).
date	Дата или метка времени 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**.

Функция `monthsend()` делит год на сегменты на основе предоставленного аргумента `n_months`. Затем она проверяет, к какому сегменту относится каждая переданная дата, и возвращает последнюю миллисекунду этого сегмента в формате даты. Функция может возвращать метку времени окончания предыдущих или последующих сегментов, а также переопределять первый месяц года.

В качестве аргументов `n_month` в функции доступны следующие сегменты года.

Аргументы `n_month`

Период	Количество месяцев
месяц	1
два месяца	2
квартал	3
четыре месяца	4
полгода	6

Когда это следует использовать

Функция `monthsend()` используется в составе выражения, когда пользователю необходимо учитывать в расчетах часть месяца, которая уже прошла. Пользователю предоставляется возможность с помощью переменной выбрать необходимый период. Например, `monthsend()` может передавать входную переменную, чтобы пользователь мог вычислить общий процент, еще не начисленный за месяц, квартал или полугодие.

Региональные настройки

Если не указано иное, в примерах, приведенных в данном разделе, используется следующий формат даты: ММ/ДД/ГГГГ. Формат даты указан в операторе `SET DateFormat` скрипта загрузки данных. В вашей системе может быть установлен другой формат даты по умолчанию в зависимости от региональных настроек и других факторов. Можно изменить формат в примерах в соответствии с потребностями. Или можно изменить форматы в скрипте загрузки в соответствии с этими примерами.

Региональные настройки по умолчанию в приложениях основаны на системных региональных настройках компьютера или сервера, где установлено ПО Qlik Sense. Если на сервере Qlik Sense, к которому обращается пользователь, выбран шведский язык, то редактор загрузки данных будет использовать шведские региональные настройки для даты, времени и валюты. Эти параметры регионального формата не связаны с языком, отображаемым в интерфейсе пользователя Qlik Sense. Qlik Sense будет отображаться на языке, который используется в браузере.

Примеры функции

Пример	Результат
`monthsend(4, '07/19/2013')`	Возвращает 08/31/2013.
`monthsend(4, '10/19/2013', -1)`	Возвращает 08/31/2013.
`monthsend(4, '10/19/2013', 0, 2)`	Возвращает 01/31/2014. Поскольку началом года становится месяц 2.

Пример 1. Базовый пример

Скрипт загрузки и результаты

Обзор

Откройте редактор загрузки данных и добавьте приведенный ниже скрипт загрузки на новую вкладку.

Скрипт загрузки содержит следующее:

- Набор данных, содержащий набор транзакций за 2022 год, загруженный в таблицу под именем `Transactions`.
- Поле даты, предоставленное в формате переменной системы `DateFormat` (`(MM/DD/YYYY)`).

- Предыдущий оператор load, который содержит следующее:
    - Функция `monthsend`, заданная как поле `bi_monthly_end`. Она группирует транзакции по двухмесячным сегментам.
    - Функция `timestamp`, которая возвращает начальную метку времени сегмента для каждой транзакции.

Скрипт загрузки

```
SET DateFormat='MM/DD/YYYY';

Transactions:
    Load
    *,
    monthsend(2,date) as bi_monthly_end,
    timestamp(monthsend(2,date)) as bi_monthly_end_timestamp
    ;
Load
*
Inline
[
id,date,amount
8188,1/7/2022,17.17
8189,1/19/2022,37.23
8190,2/28/2022,88.27
8191,2/5/2022,57.42
8192,3/16/2022,53.80
8193,4/1/2022,82.06
8194,5/7/2022,40.39
8195,5/22/2022,87.21
8196,6/15/2022,95.93
8197,6/26/2022,45.89
8198,7/9/2022,36.23
8199,7/22/2022,25.66
8200,7/23/2022,82.77
8201,7/27/2022,69.98
8202,8/2/2022,76.11
8203,8/8/2022,25.12
8204,8/19/2022,46.23
8205,9/26/2022,84.21
8206,10/14/2022,96.24
8207,10/29/2022,67.67
];
```

Результаты

Загрузите данные и откройте лист. Создайте новую таблицу и добавьте эти поля как измерения:

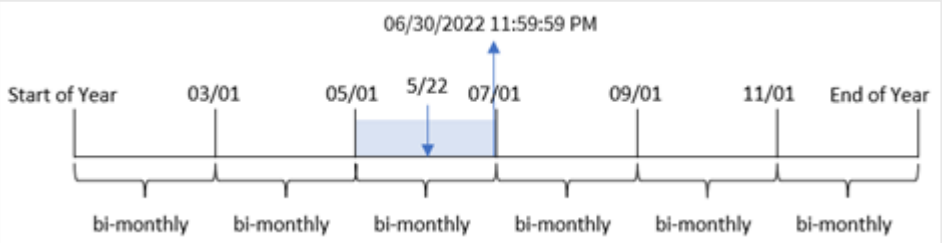
- `id`
- `date`
- `bi_monthly_end`
- `bi_monthly_end_timestamp`

Результирующая таблица

id	date	bi_monthly_end	bi_monthly_end_timestamp
8188	1/7/2022	02/28/2022	2/28/2022 11:59:59 PM
8189	1/19/2022	02/28/2022	2/28/2022 11:59:59 PM
8190	2/5/2022	02/28/2022	2/28/2022 11:59:59 PM
8191	2/28/2022	02/28/2022	2/28/2022 11:59:59 PM
8192	3/16/2022	04/30/2022	4/30/2022 11:59:59 PM
8193	4/1/2022	04/30/2022	4/30/2022 11:59:59 PM
8194	5/7/2022	06/30/2022	6/30/2022 11:59:59 PM
8195	5/22/2022	06/30/2022	6/30/2022 11:59:59 PM
8196	6/15/2022	06/30/2022	6/30/2022 11:59:59 PM
8197	6/26/2022	06/30/2022	6/30/2022 11:59:59 PM
8198	7/9/2022	08/31/2022	8/31/2022 11:59:59 PM
8199	7/22/2022	08/31/2022	8/31/2022 11:59:59 PM
8200	7/23/2022	08/31/2022	8/31/2022 11:59:59 PM
8201	7/27/2022	08/31/2022	8/31/2022 11:59:59 PM
8202	8/2/2022	08/31/2022	8/31/2022 11:59:59 PM
8203	8/8/2022	08/31/2022	8/31/2022 11:59:59 PM
8204	8/19/2022	08/31/2022	8/31/2022 11:59:59 PM
8205	9/26/2022	10/31/2022	10/31/2022 11:59:59 PM
8206	10/14/2022	10/31/2022	10/31/2022 11:59:59 PM
8207	10/29/2022	10/31/2022	10/31/2022 11:59:59 PM

Поле `bi_monthly_end` создано в предыдущем операторе load с помощью функции `monthsend()`. Первый предоставленный аргумент имеет значение `2`, что делит год на двухмесячные сегменты. Второй аргумент определяет, какое поле оценивается.

Диаграмма функции `monthsend` с двухмесячными сегментами.



Транзакция 8195 совершена 22 мая. Функция `monthsend()` изначально делит год на двухмесячные сегменты. Транзакция 8195 попадает в сегмент с 1 мая по 30 июня. В результате функция возвращает последнюю миллисекунду этого сегмента, 06/30/2022 11:59:59 PM.

Пример 2. Аргумент period_no

Скрипт загрузки и результаты

Обзор

Используется тот же набор данных и сценарий, что в первом примере.

В этом примере стоит задача создать поле `prev_bi_monthly_end`, которое возвращает первую миллисекунду двухмесячного сегмента, предшествующего совершению транзакции.

Скрипт загрузки

```
SET DateFormat='MM/DD/YYYY';

Transactions:
    Load
    *,
    monthsend(2,date,-1) as prev_bi_monthly_end,
    timestamp(monthsend(2,date,-1)) as prev_bi_monthly_end_timestamp
    ;
Load
*
Inline
[
id,date,amount
8188,1/7/2022,17.17
8189,1/19/2022,37.23
8190,2/28/2022,88.27
8191,2/5/2022,57.42
8192,3/16/2022,53.80
8193,4/1/2022,82.06
8194,5/7/2022,40.39
8195,5/22/2022,87.21
8196,6/15/2022,95.93
8197,6/26/2022,45.89
8198,7/9/2022,36.23
8199,7/22/2022,25.66
8200,7/23/2022,82.77
8201,7/27/2022,69.98
8202,8/2/2022,76.11
8203,8/8/2022,25.12
8204,8/19/2022,46.23
8205,9/26/2022,84.21
8206,10/14/2022,96.24
8207,10/29/2022,67.67
];
```

Результаты

Загрузите данные и откройте лист. Создайте новую таблицу и добавьте эти поля как измерения:

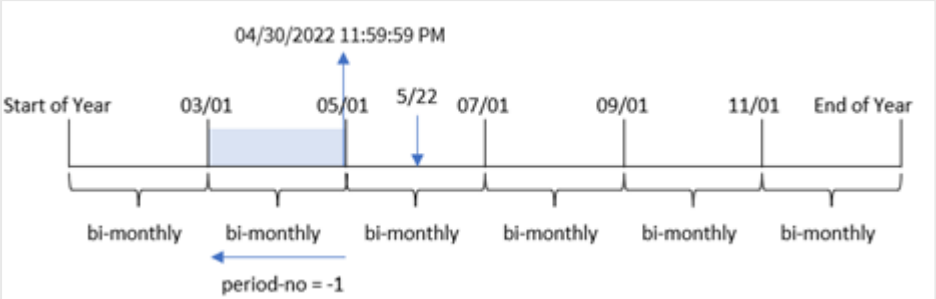
- id
- date
- prev_bi_monthly_end
- prev_bi_monthly_end_timestamp

Результирующая таблица

id	date	prev_bi_monthly_end	prev_bi_monthly_end_timestamp
8188	1/7/2022	12/31/2021	12/31/2021 11:59:59 PM
8189	1/19/2022	12/31/2021	12/31/2021 11:59:59 PM
8190	2/5/2022	12/31/2021	12/31/2021 11:59:59 PM
8191	2/28/2022	12/31/2021	12/31/2021 11:59:59 PM
8192	3/16/2022	02/28/2022	2/28/2022 11:59:59 PM
8193	4/1/2022	02/28/2022	2/28/2022 11:59:59 PM
8194	5/7/2022	04/30/2022	4/30/2022 11:59:59 PM
8195	5/22/2022	04/30/2022	4/30/2022 11:59:59 PM
8196	6/15/2022	04/30/2022	4/30/2022 11:59:59 PM
8197	6/26/2022	04/30/2022	4/30/2022 11:59:59 PM
8198	7/9/2022	06/30/2022	6/30/2022 11:59:59 PM
8199	7/22/2022	06/30/2022	6/30/2022 11:59:59 PM
8200	7/23/2022	06/30/2022	6/30/2022 11:59:59 PM
8201	7/27/2022	06/30/2022	6/30/2022 11:59:59 PM
8202	8/2/2022	06/30/2022	6/30/2022 11:59:59 PM
8203	8/8/2022	06/30/2022	6/30/2022 11:59:59 PM
8204	8/19/2022	06/30/2022	6/30/2022 11:59:59 PM
8205	9/26/2022	08/31/2022	8/31/2022 11:59:59 PM
8206	10/14/2022	08/31/2022	8/31/2022 11:59:59 PM
8207	10/29/2022	08/31/2022	8/31/2022 11:59:59 PM

Используя значение `-1` для аргумента `period_no`, функция `monthsend()` первоначально делит год на двухмесячные сегменты, а затем возвращает последнюю миллисекунду предыдущего двухмесячного сегмента для момента, когда происходит транзакция.

Диаграмма функции monthsend, которая возвращает предыдущий двухмесячный сегмент.



Транзакция 8195 совершена в сегменте с мая по июнь. В результате идентифицирован предыдущий двухмесячный сегмент с 1 марта по 30 апреля, поэтому функция вернула последнюю миллисекунду этого сегмента, 04/30/2022 11:59:59 PM.

Пример 3. Аргумент first_month_of_year

Скрипт загрузки и результаты

Обзор

Используется тот же набор данных и сценарий, что в первом примере.

В этом примере политика организации устанавливает апрель в качестве первого месяца финансового года.

Создание поля `bi_monthly_end`, которое группирует транзакции по двухмесячным сегментам, и возвращает метку времени последней миллисекунды сегмента для каждой транзакции.

Скрипт загрузки

```
SET DateFormat='MM/DD/YYYY';

Transactions:
    Load
    *,
    monthsend(2,date,0,4) as bi_monthly_end,
    timestamp(monthsend(2,date,0,4)) as bi_monthly_end_timestamp
    ;
Load
*
Inline
[
id,date,amount
8188,1/7/2022,17.17
8189,1/19/2022,37.23
8190,2/28/2022,88.27
8191,2/5/2022,57.42
8192,3/16/2022,53.80
8193,4/1/2022,82.06
8194,5/7/2022,40.39
8195,5/22/2022,87.21
```

```
8196,6/15/2022,95.93
8197,6/26/2022,45.89
8198,7/9/2022,36.23
8199,7/22/2022,25.66
8200,7/23/2022,82.77
8201,7/27/2022,69.98
8202,8/2/2022,76.11
8203,8/8/2022,25.12
8204,8/19/2022,46.23
8205,9/26/2022,84.21
8206,10/14/2022,96.24
8207,10/29/2022,67.67
];
```

Результаты

Загрузите данные и откройте лист. Создайте новую таблицу и добавьте эти поля как измерения:

- id
- date
- bi_monthly_end
- bi_monthly_end_timestamp

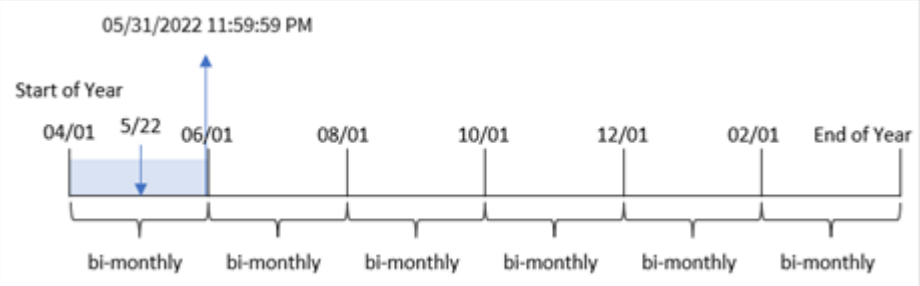
Результирующая таблица

id	date	bi_monthly_end	bi_monthly_end_timestamp
8188	1/7/2022	01/31/2022	1/31/2022 11:59:59 PM
8189	1/19/2022	01/31/2022	1/31/2022 11:59:59 PM
8190	2/5/2022	03/31/2022	3/31/2022 11:59:59 PM
8191	2/28/2022	03/31/2022	3/31/2022 11:59:59 PM
8192	3/16/2022	03/31/2022	3/31/2022 11:59:59 PM
8193	4/1/2022	05/31/2022	5/31/2022 11:59:59 PM
8194	5/7/2022	05/31/2022	5/31/2022 11:59:59 PM
8195	5/22/2022	05/31/2022	5/31/2022 11:59:59 PM
8196	6/15/2022	07/31/2022	7/31/2022 11:59:59 PM
8197	6/26/2022	07/31/2022	7/31/2022 11:59:59 PM
8198	7/9/2022	07/31/2022	7/31/2022 11:59:59 PM
8199	7/22/2022	07/31/2022	7/31/2022 11:59:59 PM
8200	7/23/2022	07/31/2022	7/31/2022 11:59:59 PM
8201	7/27/2022	07/31/2022	7/31/2022 11:59:59 PM
8202	8/2/2022	09/30/2022	9/30/2022 11:59:59 PM

id	date	bi_monthly_end	bi_monthly_end_timestamp
8203	8/8/2022	09/30/2022	9/30/2022 11:59:59 PM
8204	8/19/2022	09/30/2022	9/30/2022 11:59:59 PM
8205	9/26/2022	09/30/2022	9/30/2022 11:59:59 PM
8206	10/14/2022	11/30/2022	11/30/2022 11:59:59 PM
8207	10/29/2022	11/30/2022	11/30/2022 11:59:59 PM

Используя `4` в качестве аргумента `first_month_of_year`, функция `monthsend()` начинает год 1 апреля, а затем делит его на двухмесячные сегменты. Apr-May (апрель-май), Jun-Jul (июнь-июль), Aug-Sep (август-сентябрь), Oct-Nov (октябрь-ноябрь), Dec-Jan (декабрь-январь), Feb-Mar (февраль-март).

Диаграмма функции `monthsend`, где в качестве первого месяца года задан апрель.



Транзакция 8195 состоялась 22 мая и попадает в сегмент с 1 апреля по 31 мая. В результате функция возвращает последнюю миллисекунду этого сегмента, 05/31/2022 11:59:59 PM.

Пример 4. Пример объекта диаграммы

Скрипт загрузки и выражение диаграммы

Обзор

Используется тот же набор данных и сценарий, что в первом примере. Однако в этом примере в приложение загружается неизмененный набор данных.

В это примере стоит задача создать расчет, который группирует транзакции по двухмесячным сегментам и возвращает метку времени последней миллисекунды сегмента для каждой транзакции, в качестве меры в объекте диаграммы в приложении.

Скрипт загрузки

```
SET DateFormat='MM/DD/YYYY';

Transactions:
Load
*
Inline
```

```
[
id,date,amount
8188,2/19/2022,37.23
8189,3/7/2022,17.17
8190,3/30/2022,88.27
8191,4/5/2022,57.42
8192,4/16/2022,53.80
8193,5/1/2022,82.06
8194,5/7/2022,40.39
8195,5/22/2022,87.21
8196,6/15/2022,95.93
8197,6/26/2022,45.89
8198,7/9/2022,36.23
8199,7/22/2022,25.66
8200,7/23/2022,82.77
8201,7/27/2022,69.98
8202,8/2/2022,76.11
8203,8/8/2022,25.12
8204,8/19/2022,46.23
8205,9/26/2022,84.21
8206,10/14/2022,96.24
8207,10/29/2022,67.67
];
```

Результаты

Загрузите данные и откройте лист. Создайте новую таблицу и добавьте это поле как измерение:

```
date
```

Чтобы получить метку времени последней миллисекунды двухмесячного сегмента, в течение которого совершена транзакция, создайте следующие меры:

- `=monthsEnd(2,date)`
- `=timestamp(monthsend(2,date))`

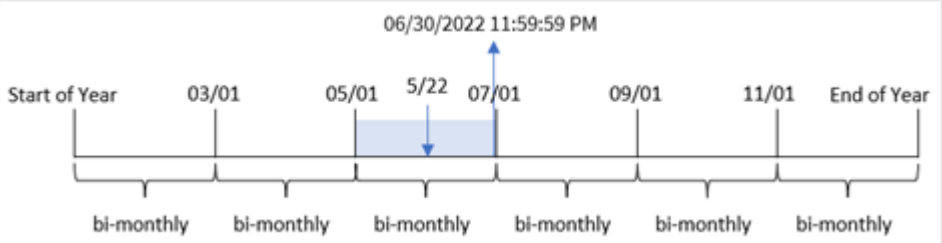
Результирующая таблица

id	date	=monthsend(2,date)	=timestamp(monthsend(2,date))
8188	1/7/2022	02/28/2022	2/28/2022 11:59:59 PM
8189	1/19/2022	02/28/2022	2/28/2022 11:59:59 PM
8190	2/5/2022	02/28/2022	2/28/2022 11:59:59 PM
8191	2/28/2022	02/28/2022	2/28/2022 11:59:59 PM
8192	3/16/2022	04/30/2022	4/30/2022 11:59:59 PM
8193	4/1/2022	04/30/2022	4/30/2022 11:59:59 PM
8194	5/7/2022	06/30/2022	6/30/2022 11:59:59 PM
8195	5/22/2022	06/30/2022	6/30/2022 11:59:59 PM

id	date	=monthsend(2,date)	=timestamp(monthsend(2,date))
8196	6/15/2022	06/30/2022	6/30/2022 11:59:59 PM
8197	6/26/2022	06/30/2022	6/30/2022 11:59:59 PM
8198	7/9/2022	08/31/2022	8/31/2022 11:59:59 PM
8199	7/22/2022	08/31/2022	8/31/2022 11:59:59 PM
8200	7/23/2022	08/31/2022	8/31/2022 11:59:59 PM
8201	7/27/2022	08/31/2022	8/31/2022 11:59:59 PM
8202	8/2/2022	08/31/2022	8/31/2022 11:59:59 PM
8203	8/8/2022	08/31/2022	8/31/2022 11:59:59 PM
8204	8/19/2022	08/31/2022	8/31/2022 11:59:59 PM
8205	9/26/2022	10/31/2022	10/31/2022 11:59:59 PM
8206	10/14/2022	10/31/2022	10/31/2022 11:59:59 PM
8207	10/29/2022	10/31/2022	10/31/2022 11:59:59 PM

Поле `bi_monthly_end` создается как мера в объекте диаграммы с помощью функции `monthsend()`. Первый предоставленный аргумент имеет значение `2`, что делит год на двухмесячные сегменты. Второй аргумент определяет, какое поле оценивается.

Диаграмма функции `monthsend` с двухмесячными сегментами.



Транзакция 8195 совершена 22 мая. Функция `monthsend()` изначально делит год на двухмесячные сегменты. Транзакция 8195 попадает в сегмент с 1 мая по 30 июня. В результате функция возвращает последнюю миллисекунду этого сегмента, 06/30/2022 11:59:59 PM.

Пример 5. Сценарий

Скрипт загрузки и результаты

Обзор

Откройте редактор загрузки данных и добавьте приведенный ниже скрипт загрузки на новую вкладку.

В этом примере набор данных загружается в таблицу под именем `Employee_Expenses`. Данная таблица содержит следующие поля:

- Employee IDs (Идентификаторы сотрудников)
- Employee names (Имена сотрудников)
- Средние ежедневные заявки на возмещение расходов каждого сотрудника.

Конечному пользователю требуется получить диаграмму, которая по идентификатору и имени сотрудника отображает ожидаемые расходы в течение оставшейся части выбранного периода. Финансовый год начинается в январе.

Скрипт загрузки

```
SET vPeriod = 1;

Employee_Expenses:
Load
*
Inline
[
employee_id,employee_name,avg_daily_claim
182,Mark, $15
183,Deryck, $12.5
184,Dexter, $12.5
185,Sydney,$27
186,Agatha,$18
];
```

Результаты

Загрузите данные и откройте новый лист.

В начале скрипта загрузки создана переменная `vPeriod`, которая будет привязана к элементу управления вводом переменной.

Выполните следующие действия.

1. На панели ресурсов щелкните **Пользовательские объекты**.
2. Выберите **Qlik Dashboard bundle** и создайте объект **Ввод переменной**.
3. Введите заголовок для объекта диаграммы.
4. В разделе **Переменная** выберите **vPeriod** в качестве имени и задайте для отображения объект **Раскрывающийся список**.
5. В списке **Значения** выберите **Динамическое**. Введите следующее:
   `='1~month|2~bi-month|3~quarter|4~tertial|6~half-year'.`

Создайте новую таблицу и добавьте эти поля как измерения:

- `employee_id`
- `employee_name`

Чтобы рассчитать накопленный процент, создайте эту меру:

```
=floor(monthsend($(vPeriod),today(1))-today(1))*avg_daily_claim
```

Эта мера является динамической и дает разные результаты в таблице в зависимости от даты загрузки данных.

Задайте параметру **Формат чисел** меры значение **Денежный**.

Результирующая таблица

employee_id	employee_ name	=floor(monthsend($(vPeriod),today(1))-today(1))*avg_daily_ claim
182	Mark	$1410.00
183	Deryck	$1175.00
184	Dexter	$1175.00
185	Sydney	$2538.00
186	Agatha	$1692.00

Функция `monthsend()` использует ввод пользователя в качестве первого аргумента и сегодняшнюю дату в качестве второго аргумента. Она возвращает дату окончания выбранного пользователем периода. Затем выражение возвращает количество дней, оставшихся в выбранном периоде, вычитая сегодняшнюю дату из даты окончания.

Затем это значение умножается на среднюю ежедневную заявку на возмещение расходов каждого сотрудника для расчета оценочной суммы заявок, которые каждый сотрудник должен подать за оставшиеся дни этого периода.

monthsname

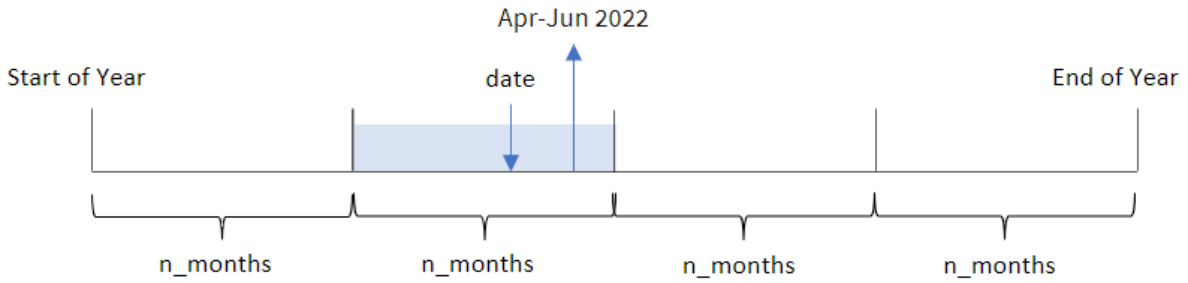
Эта функция возвращает значение, представляющее диапазон месяцев периода (форматированного согласно переменным скрипта **MonthNames**), а также года. Базовое числовое значение соответствует метке времени первой миллисекунды месяца, двухмесячного периода, квартала, трети года (четыре месяца) или полугодия, содержащих базовую дату.

Синтаксис:

```
MonthsName(n_months, date[, period_no[, first_month_of_year]])
```

Возвращаемые типы данных: двойное значение

Диаграмма функции monthsname



Функция `monthsname()` делит год на сегменты на основе предоставленного аргумента `n_months`. Затем она оценивает сегмент, к которому принадлежит каждый предоставленный элемент `date`, и возвращает названия начального и конечного месяцев этого сегмента, а также год. Функция также предоставляет возможность возвращать эти границы из предыдущих или следующих сегментов, а также переопределять первый месяц года.

В качестве аргументов `n_month` в функции доступны следующие отрезки года.

Возможные аргументы `n_month`

Периоды	Количество месяцев
месяц	1
два месяца	2
квартал	3
четыре месяца	4
полгода	6

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 (четыре месяца) или 6 (полугодие).
date	Дата или метка времени 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.

Аргумент	Описание
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**.

Когда это следует использовать

Функция `monthsname()` полезна, когда требуется предоставить пользователю возможность сравнивать агрегации по выбранному ими периоду. Например, вы можете указать входную переменную, чтобы пользователь мог просматривать общий объем продаж продуктов по месяцам, кварталам или полугодиям.

Эти измерения можно создать либо в скрипте загрузки, добавив функцию в виде поля в таблицу основного календаря или же создав измерение непосредственно в диаграмме как вычисляемое измерение.

Примеры функции

Пример	Результат
`monthsname(4, '10/19/2013')`	Возвращает Sep-Dec 2013. В этом и других примерах оператору **SET Monthnames** задается Jan;Feb;Mar и так далее.
`monthsname(4, '10/19/2013', -1)`	Возвращает May-Aug 2013.
`monthsname(4, '10/19/2013', 0, 2)`	Возвращает Oct-Jan 2014, так как указанный год начинается с месяца 2. Поэтому четырехмесячный период заканчивается в первом месяце следующего года.

Региональные настройки

Если не указано иное, в примерах, приведенных в данном разделе, используется следующий формат даты: ММ/ДД/ГГГГ. Формат даты указан в операторе `SET DateFormat` скрипта загрузки данных. В вашей системе может быть установлен другой формат даты по умолчанию в зависимости от региональных настроек и других факторов. Можно изменить формат в примерах в соответствии с потребностями. Или можно изменить форматы в скрипте загрузки в соответствии с этими примерами.

Региональные настройки по умолчанию в приложениях основаны на системных региональных настройках компьютера или сервера, где установлено ПО Qlik Sense. Если на сервере Qlik Sense, к которому обращается пользователь, выбран шведский язык, то редактор загрузки данных будет использовать шведские региональные настройки для даты, времени и валюты. Эти параметры регионального формата не связаны с языком, отображаемым в интерфейсе пользователя Qlik Sense. Qlik Sense будет отображаться на языке, который используется в браузере.

Пример 1. Базовый пример

Скрипт загрузки и результаты

Обзор

Откройте редактор загрузки данных и добавьте приведенный ниже скрипт загрузки на новую вкладку.

Скрипт загрузки содержит следующее:

- Набор данных, содержащий набор транзакций за 2022 год, загружается в таблицу под именем `Transactions`.
- Поле даты было предоставлено в формате системной переменной `DateFormat` (MM/DD/YYYY).
- Создание поля `bi_monthly_range`, которое группирует транзакции в двухмесячные сегменты и возвращает имена границ этого сегмента для каждой транзакции.

Скрипт загрузки

```
SET DateFormat='MM/DD/YYYY';

Transactions:
    Load
        *,
        monthsname(2,date) as bi_monthly_range
        ;
Load
*
Inline
[
id,date,amount
8188,2/19/2022,37.23
8189,3/7/2022,17.17
8190,3/30/2022,88.27
8191,4/5/2022,57.42
8192,4/16/2022,53.80
8193,5/1/2022,82.06
8194,5/7/2022,40.39
8195,5/22/2022,87.21
8196,6/15/2022,95.93
8197,6/26/2022,45.89
8198,7/9/2022,36.23
8199,7/22/2022,25.66
8200,7/23/2022,82.77
8201,7/27/2022,69.98
8202,8/2/2022,76.11
8203,8/8/2022,25.12
8204,8/19/2022,46.23
8205,9/26/2022,84.21
8206,10/14/2022,96.24
```

```
8207,10/29/2022,67.67
];
```

Результаты

Загрузите данные и откройте лист. Создайте новую таблицу и добавьте эти поля как измерения:

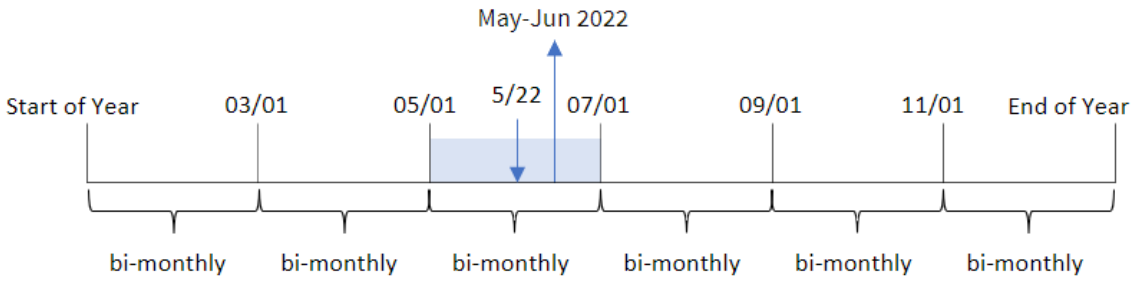
- `date`
- `bi_monthly_range`

Результирующая таблица

date	bi_monthly_range
2/19/2022	Jan-Feb 2022
3/7/2022	Mar-Apr 2022
3/30/2022	Mar-Apr 2022
4/5/2022	Mar-Apr 2022
4/16/2022	Mar-Apr 2022
5/1/2022	May-Jun 2022
5/7/2022	May-Jun 2022
5/22/2022	May-Jun 2022
6/15/2022	May-Jun 2022
6/26/2022	May-Jun 2022
7/9/2022	Jul-Aug 2022
7/22/2022	Jul-Aug 2022
7/23/2022	Jul-Aug 2022
7/27/2022	Jul-Aug 2022
8/2/2022	Jul-Aug 2022
8/8/2022	Jul-Aug 2022
8/19/2022	Jul-Aug 2022
9/26/2022	Sep-Oct 2022
10/14/2022	Sep-Oct 2022
10/29/2022	Sep-Oct 2022

Поле `bi_monthly_range` создано в предыдущем операторе load с помощью функции `monthsname()`. Первый предоставленный аргумент имеет значение 2, что делит год на двухмесячные сегменты. Второй аргумент определяет, какое поле оценивается.

Диаграмма функции monthsname, базовый пример



Транзакция 8195 совершена 22 мая. Функция `monthsname()` изначально делит год на двухмесячные сегменты. Транзакция 8195 попадает в сегмент с 1 мая по 30 июня. Поэтому функция возвращает эти месяцы в формате системной переменной `MonthNames`, а также год: May-Jun 2022.

Пример 2. Скрипт period_no

Скрипт загрузки и результаты

Обзор

Откройте редактор загрузки данных и добавьте приведенный ниже скрипт загрузки на новую вкладку.

Скрипт загрузки содержит следующее:

- Используется тот же встроенный набор данных и сценарий, что в первом примере.
- Создание поля `prev_bi_monthly_range`, которое группирует транзакции в двухмесячные сегменты и возвращает имена границ этого сегмента для каждой транзакции.

Добавьте сюда другой текст, если необходимо, со списками и т. д.

Скрипт загрузки

```
SET DateFormat='MM/DD/YYYY';

Transactions:
    Load
        *,
        MonthsName(2,date,-1) as prev_bi_monthly_range
        ;
Load
*
Inline
[
id,date,amount
8188,2/19/2022,37.23
8189,3/7/2022,17.17
8190,3/30/2022,88.27
8191,4/5/2022,57.42
```

```
8192,4/16/2022,53.80
8193,5/1/2022,82.06
8194,5/7/2022,40.39
8195,5/22/2022,87.21
8196,6/15/2022,95.93
8197,6/26/2022,45.89
8198,7/9/2022,36.23
8199,7/22/2022,25.66
8200,7/23/2022,82.77
8201,7/27/2022,69.98
8202,8/2/2022,76.11
8203,8/8/2022,25.12
8204,8/19/2022,46.23
8205,9/26/2022,84.21
8206,10/14/2022,96.24
8207,10/29/2022,67.67
];
```

Результаты

Загрузите данные и откройте лист. Создайте новую таблицу и добавьте эти поля как измерения:

- `date`
- `prev_bi_monthly_range`

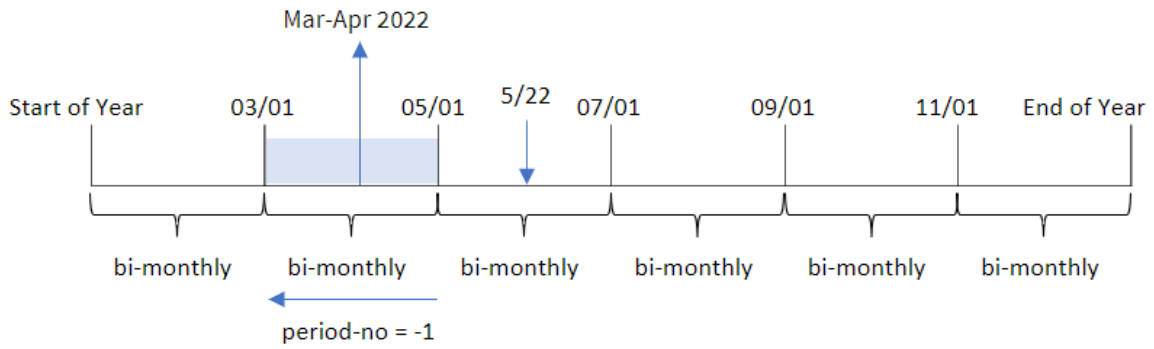
Результирующая таблица

date	prev_bi_monthly_range
2/19/2022	Nov-Dec 2021
3/7/2022	Jan-Feb 2022
3/30/2022	Jan-Feb 2022
4/5/2022	Jan-Feb 2022
4/16/2022	Jan-Feb 2022
5/1/2022	Mar-Apr 2022
5/7/2022	Mar-Apr 2022
5/22/2022	Mar-Apr 2022
6/15/2022	Mar-Apr 2022
6/26/2022	Mar-Apr 2022
7/9/2022	May-Jun 2022
7/22/2022	May-Jun 2022
7/23/2022	May-Jun 2022
7/27/2022	May-Jun 2022

date	prev_bi_monthly_range
8/2/2022	May-Jun 2022
8/8/2022	May-Jun 2022
8/19/2022	May-Jun 2022
9/26/2022	Jul-Aug 2022
10/14/2022	Jul-Aug 2022
10/29/2022	Jul-Aug 2022

В этом примере -1 используется в качестве аргумента `period_no` в функции `monthsname()`. Первоначально разделив год на двухмесячные сегменты, функция затем возвращает границы предыдущего сегмента для момента, когда происходит транзакция.

Диаграмма функции monthsname, пример с аргументом period_no



Транзакция 8195 совершена в сегменте с мая по июнь. Таким образом, предыдущий двухмесячный сегмент начинался 1 марта и заканчивался 30 апреля, поэтому функция возвращает Mar-Apr 2022.

Пример 3. Аргумент first_month_of_year

Скрипт загрузки и результаты

Обзор

Откройте редактор загрузки данных и добавьте приведенный ниже скрипт загрузки на новую вкладку.

Скрипт загрузки содержит следующее:

- Используется тот же встроенный набор данных и сценарий, что в первом примере.
- Создание другого поля, `bi_monthly_range`, которое группирует транзакции в двухмесячные сегменты и возвращает границы сегментов для каждой транзакции.

Однако в этом примере нам также нужно задать апрель в качестве первого месяца финансового года.

Скрипт загрузки

```
SET DateFormat='MM/DD/YYYY';

Transactions:
    Load
        *,
        MonthsName(2,date,0,4) as bi_monthly_range
        ;
Load
*
Inline
[
id,date,amount
8188,2/19/2022,37.23
8189,3/7/2022,17.17
8190,3/30/2022,88.27
8191,4/5/2022,57.42
8192,4/16/2022,53.80
8193,5/1/2022,82.06
8194,5/7/2022,40.39
8195,5/22/2022,87.21
8196,6/15/2022,95.93
8197,6/26/2022,45.89
8198,7/9/2022,36.23
8199,7/22/2022,25.66
8200,7/23/2022,82.77
8201,7/27/2022,69.98
8202,8/2/2022,76.11
8203,8/8/2022,25.12
8204,8/19/2022,46.23
8205,9/26/2022,84.21
8206,10/14/2022,96.24
8207,10/29/2022,67.67
];
```

Результаты

Загрузите данные и откройте лист. Создайте новую таблицу и добавьте эти поля как измерения:

- `date`
- `bi_monthly_range`

Результирующая таблица

date	bi_monthly_range
2/19/2022	Feb-Mar 2021
3/7/2022	Feb-Mar 2021

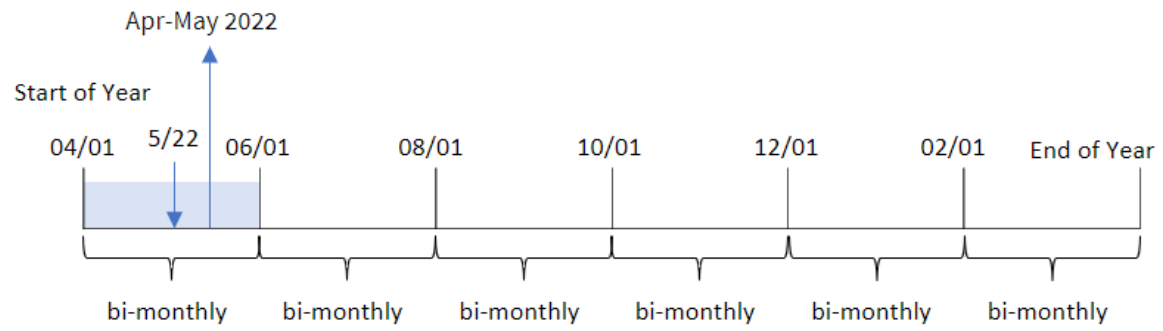
date	bi_monthly_range
3/30/2022	Feb-Mar 2021
4/5/2022	Apr-May 2022
4/16/2022	Apr-May 2022
5/1/2022	Apr-May 2022
5/7/2022	Apr-May 2022
5/22/2022	Apr-May 2022
6/15/2022	Jun-Jul 2022
6/26/2022	Jun-Jul 2022
7/9/2022	Jun-Jul 2022
7/22/2022	Jun-Jul 2022
7/23/2022	Jun-Jul 2022
7/27/2022	Jun-Jul 2022
8/2/2022	Aug-Sep 2022
8/8/2022	Aug-Sep 2022
8/19/2022	Aug-Sep 2022
9/26/2022	Aug-Sep 2022
10/14/2022	Oct-Nov 2022
10/29/2022	Oct-Nov 2022

Так как используется 4 в качестве аргумента `first_month_of_year` в функции `monthsname()`, функция начинает год 1 апреля, а затем делит год на двухмесячные сегменты. Apr-May,Jun-Jul,Aug-Sep,Oct-Nov,Dec-Jan,Feb-Mar.

Текст абзаца для результатов.

Транзакция 8195 состоялась 22 мая и попадает в сегмент с 1 апреля по 31 мая. Поэтому функция возвращает Apr-May 2022.

Диаграмма функции monthsname, пример с аргументом first_month_of_year



Пример 4. Пример объекта диаграммы

Скрипт загрузки и выражение диаграммы

Обзор

Откройте редактор загрузки данных и добавьте приведенный ниже скрипт загрузки на новую вкладку.

Скрипт загрузки содержит тот же встроенный набор данных и сценарий, что в первом примере. Однако в этом примере в приложение загружается неизмененный набор данных. Расчет, который группирует транзакции в двухмесячные сегменты и возвращает границы сегментов для каждой транзакции, создается как мера в объекте диаграммы приложения.

Скрипт загрузки

```
SET DateFormat='MM/DD/YYYY';

Transactions:
Load
*
Inline
[
id,date,amount
8188,2/19/2022,37.23
8189,3/7/2022,17.17
8190,3/30/2022,88.27
8191,4/5/2022,57.42
8192,4/16/2022,53.80
8193,5/1/2022,82.06
8194,5/7/2022,40.39
8195,5/22/2022,87.21
8196,6/15/2022,95.93
8197,6/26/2022,45.89
8198,7/9/2022,36.23
8199,7/22/2022,25.66
8200,7/23/2022,82.77
8201,7/27/2022,69.98
```

```
8202,8/2/2022,76.11
8203,8/8/2022,25.12
8204,8/19/2022,46.23
8205,9/26/2022,84.21
8206,10/14/2022,96.24
8207,10/29/2022,67.67
];
```

Результаты

Загрузите данные и откройте лист. Создайте новую таблицу и добавьте это поле как измерение:`date`.

Создайте следующую меру:

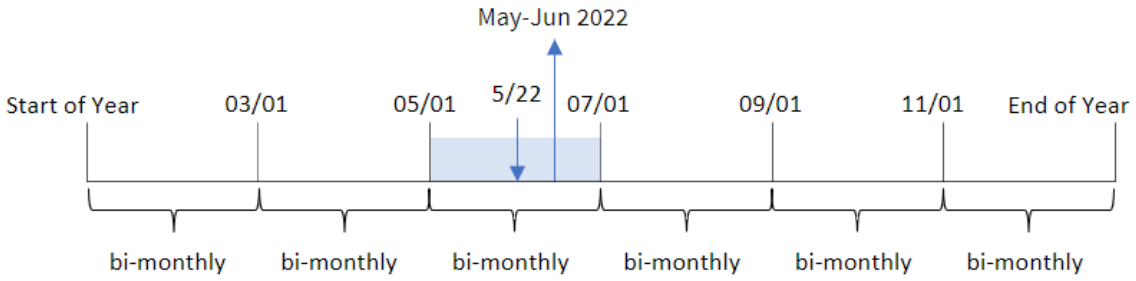
```
=monthsname(2,date)
```

Результирующая таблица

date	=monthsname(2,date)
2/19/2022	Jan-Feb 2022
3/7/2022	Mar-Apr 2022
3/30/2022	Mar-Apr 2022
4/5/2022	Mar-Apr 2022
4/16/2022	Mar-Apr 2022
5/1/2022	May-Jun 2022
5/7/2022	May-Jun 2022
5/22/2022	May-Jun 2022
6/15/2022	May-Jun 2022
6/26/2022	May-Jun 2022
7/9/2022	Jul-Aug 2022
7/22/2022	Jul-Aug 2022
7/23/2022	Jul-Aug 2022
7/27/2022	Jul-Aug 2022
8/2/2022	Jul-Aug 2022
8/8/2022	Jul-Aug 2022
8/19/2022	Jul-Aug 2022
9/26/2022	Sep-Oct 2022
10/14/2022	Sep-Oct 2022
10/29/2022	Sep-Oct 2022

Поле `bi_monthly_range` создается как мера в объекте диаграммы с помощью функции `monthsname()`. Первый предоставленный аргумент имеет значение 2, что делит год на двухмесячные сегменты. Второй аргумент определяет, какое поле оценивается.

Диаграмма функции monthsname, пример с объектом диаграммы



Транзакция 8195 совершена 22 мая. Функция `monthsname()` изначально делит год на двухмесячные сегменты. Транзакция 8195 попадает в сегмент с 1 мая по 30 июня. Поэтому функция возвращает эти месяцы в формате системной переменной `MonthNames`, а также год: May-Jun 2022.

Пример 5. Сценарий

Скрипт загрузки и выражение диаграммы

Обзор

Откройте редактор загрузки данных и добавьте приведенный ниже скрипт загрузки на новую вкладку.

Скрипт загрузки содержит следующее:

- Набор данных, содержащий набор транзакций за 2022 год, загружается в таблицу под именем `Transactions`.
- Поле даты было предоставлено в формате системной переменной `DateFormat` (MM/DD/YYYY).

Конечному пользователю требуется объект диаграммы, отображающий общий объем продаж за выбранный им период. Этого можно добиться, даже если это измерение недоступно в модели данных, используя функцию `monthsname()` в качестве вычисляемого измерения, которое динамически изменяется с помощью элемента управления вводом переменной.

Скрипт загрузки

```
SET vPeriod = 1;
SET DateFormat='MM/DD/YYYY';

Transactions:
Load
*
Inline
```

```
[
id,date,amount
8188,'1/7/2022',17.17
8189,'1/19/2022',37.23
8190,'2/28/2022',88.27
8191,'2/5/2022',57.42
8192,'3/16/2022',53.80
8193,'4/1/2022',82.06
8194,'5/7/2022',40.39
8195,'5/16/2022',87.21
8196,'6/15/2022',95.93
8197,'6/26/2022',45.89
8198,'7/9/2022',36.23
8199,'7/22/2022',25.66
8200,'7/23/2022',82.77
8201,'7/27/2022',69.98
8202,'8/2/2022',76.11
8203,'8/8/2022',25.12
8204,'8/19/2022',46.23
8205,'9/26/2022',84.21
8206,'10/14/2022',96.24
8207,'10/29/2022',67.67
];
```

Результаты

Загрузите данные и откройте лист.

В начале скрипта загрузки создана переменная (`vPeriod`), которая будет привязана к элементу управления вводом переменной. Затем настройте переменную как настраиваемый объект на листе.

Выполните следующие действия.

1. На панели ресурсов щелкните **Пользовательские объекты**.
2. Выберите **Qlik Dashboard bundle**и создайте объект **Ввод переменной**.
3. Введите заголовок для объекта диаграммы.
4. В разделе **Переменная** выберите **vPeriod** в качестве имени и задайте для отображения объект **Раскрывающийся список**.
5. В разделе **Значения** настройте для объекта использование динамических значений. Введите следующее:
   `='1~month|2~bi-month|3~quarter|4~tertial|6~half-year'`

Затем создайте таблицу результатов.

Выполните следующие действия.

1. Создайте новую таблицу и добавьте следующее вычисляемое измерение:
   `=monthsname($(vPeriod),date)`
2. Добавьте эту меру для расчета общего объема продаж:

```
=sum(amount)
```

3. Задайте параметру меры **Формат чисел** значение **Денежный**. Щелкните ✔ **Изменение завершено**. Теперь можно модифицировать данные, показанные в таблице, изменив сегмент времени в объекте переменной.

Вот как будет выглядеть таблица результатов при выборе варианта `tertial`:

Результирующая таблица

monthsname($(vPeriod),date)	sum(amount)
Jan-Apr 2022	253.89
Май-август 2022 г.	713.58
Sep-Dec 2022	248.12

monthsstart

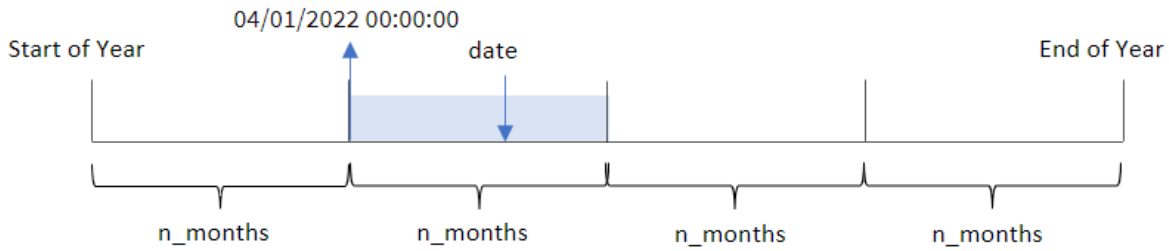
Эта функция возвращает значение, соответствующее метке времени первой миллисекунды месяца, двухмесячного периода, квартала, трети года (четыре месяца) или полугодия, содержащих базовую дату. Также можно найти метку времени для предыдущего или последующего временного периода.По умолчанию для вывода используется формат **DateFormat**, установленный в скрипте.

Синтаксис:

```
MonthsStart(n_months, date[, period_no [, first_month_of_year]])
```

Возвращаемые типы данных: двойное значение

Схема функции `monthsstart()`



Функция `monthsstart()` делит год на сегменты на основе предоставленного аргумента `n_months`. Затем она проверяет, к какому сегменту относится каждая переданная дата, и возвращает первую миллисекунду этого сегмента в формате даты. Функция также предоставляет возможность возвращать метку времени начала предыдущих или следующих сегментов, а также переопределять первый месяц года.

В качестве аргументов `n_month` в функции доступны следующие отрезки года.

Возможные аргументы `n_month`

Периоды	Количество месяцев
месяц	1
два месяца	2
квартал	3
четыре месяца	4
полгода	6

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 (четыре месяца) или 6 (полугодие).
date	Дата или метка времени 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**.

Когда это следует использовать

Функция `monthsstart()` широко используется в составе выражения, когда пользователю требуется учитывать в расчетах часть периода, которая еще не прошла. Например, это позволяет передавать входную переменную, чтобы пользователь мог вычислить общий процент, на данный момент начисленный за месяц, квартал или полугодие.

Примеры функции

Пример	Результат
`monthsstart(4, '10/19/2013')`	Возвращает 09/01/2013.
`monthsstart(4, '10/19/2013, -1)`	Возвращает 05/01/2013.
`monthsstart(4, '10/19/2013', 0, 2 )`	Возвращает 10/01/2013, поскольку началом года становится месяц 2.

Региональные настройки

Если не указано иное, в примерах, приведенных в данном разделе, используется следующий формат даты: MM/ДД/ГГГГ. Формат даты указан в операторе `SET DateFormat` скрипта загрузки данных. В вашей системе может быть установлен другой формат даты по умолчанию в зависимости от региональных настроек и других факторов. Можно изменить формат в примерах в соответствии с потребностями. Или можно изменить форматы в скрипте загрузки в соответствии с этими примерами.

Региональные настройки по умолчанию в приложениях основаны на системных региональных настройках компьютера или сервера, где установлено ПО Qlik Sense. Если на сервере Qlik Sense, к которому обращается пользователь, выбран шведский язык, то редактор загрузки данных будет использовать шведские региональные настройки для даты, времени и валюты. Эти параметры регионального формата не связаны с языком, отображаемым в интерфейсе пользователя Qlik Sense. Qlik Sense будет отображаться на языке, который используется в браузере.

Пример 1. Без дополнительных аргументов

Скрипт загрузки и результаты

Обзор

Откройте редактор загрузки данных и добавьте приведенный ниже скрипт загрузки на новую вкладку.

Скрипт загрузки содержит следующее:

- Набор данных, содержащий набор транзакций за 2022 год, загружается в таблицу под именем `Transactions`.
- Поле даты было предоставлено в формате системной переменной `DateFormat` (MM/DD/YYYY).
- Создание поля `bi_monthly_start`, которое группирует транзакции по двухмесячным сегментам и возвращает метку времени начала сегмента для каждой транзакции.

Скрипт загрузки

```
SET DateFormat='MM/DD/YYYY';

Transactions:
    Load
        *,
        monthsstart(2,date) as bi_monthly_start,
        timestamp(monthsstart(2,date)) as bi_monthly_start_timestamp
        ;
Load
*
Inline
[
id,date,amount
8188,2/19/2022,37.23
```

```
8189,3/7/2022,17.17
8190,3/30/2022,88.27
8191,4/5/2022,57.42
8192,4/16/2022,53.80
8193,5/1/2022,82.06
8194,5/7/2022,40.39
8195,5/22/2022,87.21
8196,6/15/2022,95.93
8197,6/26/2022,45.89
8198,7/9/2022,36.23
8199,7/22/2022,25.66
8200,7/23/2022,82.77
8201,7/27/2022,69.98
8202,8/2/2022,76.11
8203,8/8/2022,25.12
8204,8/19/2022,46.23
8205,9/26/2022,84.21
8206,10/14/2022,96.24
8207,10/29/2022,67.67
];
```

Результаты

Загрузите данные и откройте лист. Создайте новую таблицу и добавьте эти поля как измерения:

- `date`
- `bi_monthly_start`
- `bi_monthly_start_timestamp`

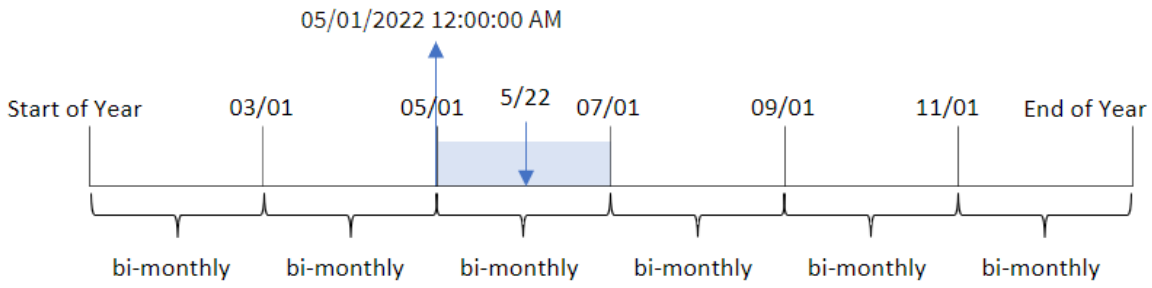
Результирующая таблица

date	bi_monthly_start	bi_monthly_start_timestamp
2/19/2022	01/01/2022	1/1/2022 12:00:00 AM
3/7/2022	03/01/2022	3/1/2022 12:00:00 AM
3/30/2022	03/01/2022	3/1/2022 12:00:00 AM
4/5/2022	03/01/2022	3/1/2022 12:00:00 AM
4/16/2022	03/01/2022	3/1/2022 12:00:00 AM
5/1/2022	05/01/2022	5/1/2022 12:00:00 AM
5/7/2022	05/01/2022	5/1/2022 12:00:00 AM
5/22/2022	05/01/2022	5/1/2022 12:00:00 AM
6/15/2022	05/01/2022	5/1/2022 12:00:00 AM
6/26/2022	05/01/2022	5/1/2022 12:00:00 AM
7/9/2022	07/01/2022	7/1/2022 12:00:00 AM
7/22/2022	07/01/2022	7/1/2022 12:00:00 AM

date	bi_monthly_start	bi_monthly_start_timestamp
7/23/2022	07/01/2022	7/1/2022 12:00:00 AM
7/27/2022	07/01/2022	7/1/2022 12:00:00 AM
8/2/2022	07/01/2022	7/1/2022 12:00:00 AM
8/8/2022	07/01/2022	7/1/2022 12:00:00 AM
8/19/2022	07/01/2022	7/1/2022 12:00:00 AM
9/26/2022	09/01/2022	9/1/2022 12:00:00 AM
10/14/2022	09/01/2022	9/1/2022 12:00:00 AM
10/29/2022	09/01/2022	9/1/2022 12:00:00 AM

Поле `bi_monthly_start` создано в предыдущем операторе load с помощью функции `monthsstart()`. Первый предоставленный аргумент имеет значение 2, что делит год на двухмесячные сегменты. Второй аргумент определяет, какое поле оценивается.

Диаграмма функции `monthsstart()`, пример без дополнительных аргументов



Транзакция 8195 совершена 22 мая. Функция `monthsstart()` изначально делит год на двухмесячные сегменты. Транзакция 8195 попадает в сегмент с 1 мая по 30 июня. Таким образом, функция возвращает первую миллисекунду этого сегмента, то есть 00:00:00 (12:00:00 AM) 1 мая 2022 года.

Пример 2. Скрипт period_no

Скрипт загрузки и результаты

Обзор

Откройте редактор загрузки данных и добавьте приведенный ниже скрипт загрузки на новую вкладку.

Скрипт загрузки содержит следующее:

- Используется тот же набор данных и сценарий, что в первом примере.
- Создание поля `prev_bi_monthly_start`, которое возвращает первую миллисекунду двухмесячного сегмента, предшествующего совершению транзакции.

Скрипт загрузки

```
SET DateFormat='MM/DD/YYYY';

Transactions:
    Load
        *,
        monthsstart(2,date,-1) as prev_bi_monthly_start,
           timestamp(monthsstart(2,date,-1)) as prev_bi_monthly_start_timestamp
           ;
Load
*
Inline
[
id,date,amount
8188,2/19/2022,37.23
8189,3/7/2022,17.17
8190,3/30/2022,88.27
8191,4/5/2022,57.42
8192,4/16/2022,53.80
8193,5/1/2022,82.06
8194,5/7/2022,40.39
8195,5/22/2022,87.21
8196,6/15/2022,95.93
8197,6/26/2022,45.89
8198,7/9/2022,36.23
8199,7/22/2022,25.66
8200,7/23/2022,82.77
8201,7/27/2022,69.98
8202,8/2/2022,76.11
8203,8/8/2022,25.12
8204,8/19/2022,46.23
8205,9/26/2022,84.21
8206,10/14/2022,96.24
8207,10/29/2022,67.67
];
```

Результаты

Загрузите данные и откройте лист. Создайте новую таблицу и добавьте эти поля как измерения:

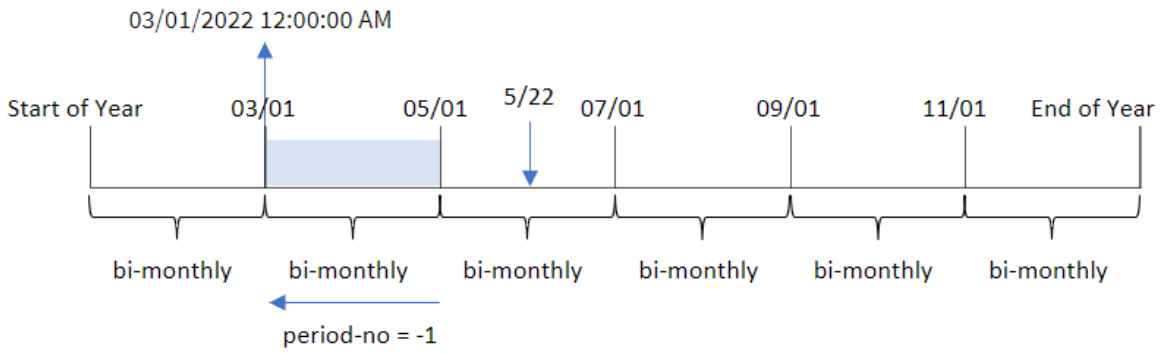
- `date`
- `prev_bi_monthly_start`
- `prev_bi_monthly_start_timestamp`

Результирующая таблица

date	prev_bi_monthly_start	prev_bi_monthly_start_timestamp
2/19/2022	11/01/2021	11/1/2021 12:00:00 AM
3/7/2022	01/01/2022	1/1/2022 12:00:00 AM
3/30/2022	01/01/2022	1/1/2022 12:00:00 AM
4/5/2022	01/01/2022	1/1/2022 12:00:00 AM
4/16/2022	01/01/2022	1/1/2022 12:00:00 AM
5/1/2022	03/01/2022	3/1/2022 12:00:00 AM
5/7/2022	03/01/2022	3/1/2022 12:00:00 AM
5/22/2022	03/01/2022	3/1/2022 12:00:00 AM
6/15/2022	03/01/2022	3/1/2022 12:00:00 AM
6/26/2022	03/01/2022	3/1/2022 12:00:00 AM
7/9/2022	05/01/2022	5/1/2022 12:00:00 AM
7/22/2022	05/01/2022	5/1/2022 12:00:00 AM
7/23/2022	05/01/2022	5/1/2022 12:00:00 AM
7/27/2022	05/01/2022	5/1/2022 12:00:00 AM
8/2/2022	05/01/2022	5/1/2022 12:00:00 AM
8/8/2022	05/01/2022	5/1/2022 12:00:00 AM
8/19/2022	05/01/2022	5/1/2022 12:00:00 AM
9/26/2022	07/01/2022	7/1/2022 12:00:00 AM
10/14/2022	07/01/2022	7/1/2022 12:00:00 AM
10/29/2022	07/01/2022	7/1/2022 12:00:00 AM

Используя значение -1 для аргумента `period_no`, функция `monthsstart()` первоначально делит год на двухмесячные сегменты, а затем возвращает первую миллисекунду предыдущего двухмесячного сегмента для момента, когда происходит транзакция.

Диаграмма функции monthsstart(), пример с аргументом period_no



Транзакция 8195 совершена в сегменте с мая по июнь. Таким образом, предшествующий двухмесячный сегмент был с 1 марта по 30 апреля, поэтому функция возвращает первую миллисекунду этого сегмента, то есть 00:00:00 (12:00:00 AM) 1 мая 2022.

Пример 3. Аргумент first_month_of_year

Скрипт загрузки и результаты

Обзор

Откройте редактор загрузки данных и добавьте приведенный ниже скрипт загрузки на новую вкладку.

Скрипт загрузки содержит следующее:

- Используется тот же набор данных и сценарий, что в первом примере.
- Создание поля `bi_monthly_start`, которое группирует транзакции по двухмесячным сегментам и возвращает метку времени начала сегмента для каждой транзакции.

Однако в этом примере нам также нужно задать апрель в качестве первого месяца финансового года.

Скрипт загрузки

```
SET DateFormat='MM/DD/YYYY';

Transactions:
    Load
        *,
        monthsstart(2,date,0,4) as bi_monthly_start,
        timestamp(monthsstart(2,date,0,4)) as bi_monthly_start_timestamp
        ;
Load
*
Inline
[
id,date,amount
```

```
8188,1/7/2022,17.17
8189,1/19/2022,37.23
8190,2/28/2022,88.27
8191,2/5/2022,57.42
8192,3/16/2022,53.80
8193,4/1/2022,82.06
8194,5/7/2022,40.39
8195,5/16/2022,87.21
8196,6/15/2022,95.93
8197,6/26/2022,45.89
8198,7/9/2022,36.23
8199,7/22/2022,25.66
8200,7/23/2022,82.77
8201,7/27/2022,69.98
8202,8/2/2022,76.11
8203,8/8/2022,25.12
8204,8/19/2022,46.23
8205,9/26/2022,84.21
8206,10/14/2022,96.24
8207,10/29/2022,67.67
];
```

Результаты

Загрузите данные и откройте лист. Создайте новую таблицу и добавьте эти поля как измерения:

- `date`
- `bi_monthly_start`
- `bi_monthly_start_timestamp`

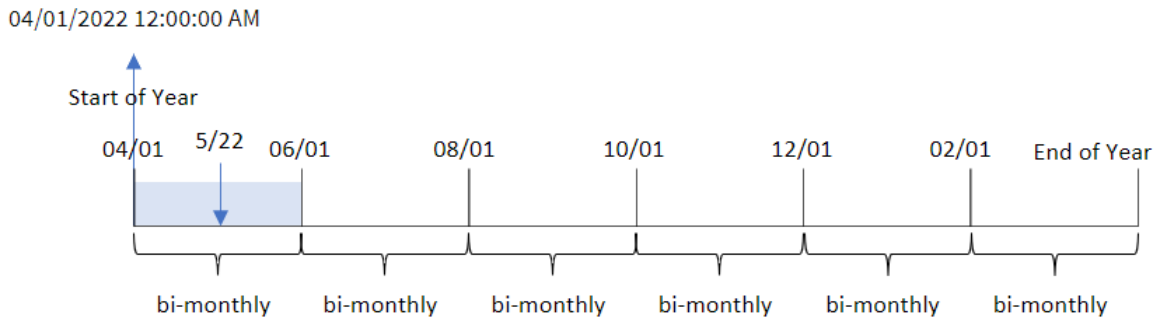
Результирующая таблица

date	bi_monthly_start	bi_monthly_start_timestamp
2/19/2022	02/01/2022	2/1/2022 12:00:00 AM
3/7/2022	02/01/2022	2/1/2022 12:00:00 AM
3/30/2022	02/01/2022	2/1/2022 12:00:00 AM
4/5/2022	04/01/2022	4/1/2022 12:00:00 AM
4/16/2022	04/01/2022	4/1/2022 12:00:00 AM
5/1/2022	04/01/2022	4/1/2022 12:00:00 AM
5/7/2022	04/01/2022	4/1/2022 12:00:00 AM
5/22/2022	04/01/2022	4/1/2022 12:00:00 AM
6/15/2022	06/01/2022	6/1/2022 12:00:00 AM
6/26/2022	06/01/2022	6/1/2022 12:00:00 AM
7/9/2022	06/01/2022	6/1/2022 12:00:00 AM

date	bi_monthly_start	bi_monthly_start_timestamp
7/22/2022	06/01/2022	6/1/2022 12:00:00 AM
7/23/2022	06/01/2022	6/1/2022 12:00:00 AM
7/27/2022	06/01/2022	6/1/2022 12:00:00 AM
8/2/2022	08/01/2022	8/1/2022 12:00:00 AM
8/8/2022	08/01/2022	8/1/2022 12:00:00 AM
8/19/2022	08/01/2022	8/1/2022 12:00:00 AM
9/26/2022	08/01/2022	8/1/2022 12:00:00 AM
10/14/2022	10/01/2022	10/1/2022 12:00:00 AM
10/29/2022	10/01/2022	10/1/2022 12:00:00 AM

Так как используется 4 в качестве аргумента `first_month_of_year` в функции `monthsstart()`, функция начинает год 1 апреля, а затем делит год на двухмесячные сегменты. Apr-May,Jun-Jul,Aug-Sep,Oct-Nov,Dec-Jan,Feb-Mar.

Диаграмма функции `monthsstart()`, пример с аргументом first_month_of_year



Транзакция 8195 состоялась 22 мая и попадает в сегмент с 1 апреля по 31 мая. Таким образом, функция возвращает первую миллисекунду этого сегмента, то есть в 00:00:00 (12:00:00 AM) 1 апреля 2022 года.

Пример 4. Пример объекта диаграммы

Скрипт загрузки и выражение диаграммы

Обзор

Откройте редактор загрузки данных и добавьте приведенный ниже скрипт загрузки на новую вкладку.

Скрипт загрузки содержит тот же набор данных и сценарий, что в первом примере.

Однако в этом примере в приложение загружается неизмененный набор данных. Расчет, который группирует транзакции в двухмесячные сегменты и возвращает начальную метку времени сегмента для каждой транзакции, создается как мера в объекте диаграммы приложения.

Скрипт загрузки

```
SET DateFormat='MM/DD/YYYY';

Transactions:
Load
*
Inline
[
id,date,amount
8188,2/19/2022,37.23
8189,3/7/2022,17.17
8190,3/30/2022,88.27
8191,4/5/2022,57.42
8192,4/16/2022,53.80
8193,5/1/2022,82.06
8194,5/7/2022,40.39
8195,5/22/2022,87.21
8196,6/15/2022,95.93
8197,6/26/2022,45.89
8198,7/9/2022,36.23
8199,7/22/2022,25.66
8200,7/23/2022,82.77
8201,7/27/2022,69.98
8202,8/2/2022,76.11
8203,8/8/2022,25.12
8204,8/19/2022,46.23
8205,9/26/2022,84.21
8206,10/14/2022,96.24
8207,10/29/2022,67.67
];
```

Результаты

Загрузите данные и откройте лист. Создайте новую таблицу и добавьте это поле как измерение: `date`.

Создайте следующие меры:

```
=monthsstart(2,date)
```

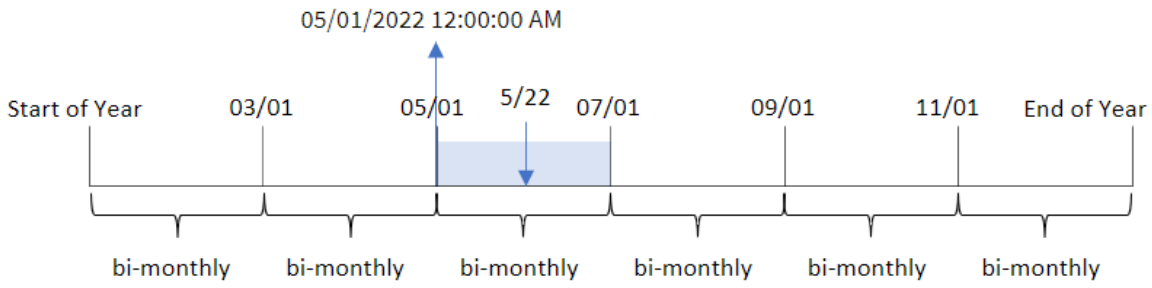
```
=timestamp(monthsstart(2,date))
```

Эти вычисления возвращают начальную метку времени двухмесячного сегмента, в течение которого совершена каждая транзакция.

Результирующая таблица

date	=monthsstart(2,date)	=timestamp(monthsstart(2,date))
9/26/2022	09/01/2022	9/1/2022 12:00:00 AM
10/14/2022	09/01/2022	9/1/2022 12:00:00 AM
10/29/2022	09/01/2022	9/1/2022 12:00:00 AM
7/9/2022	07/01/2022	7/1/2022 12:00:00 AM
7/22/2022	07/01/2022	7/1/2022 12:00:00 AM
7/23/2022	07/01/2022	7/1/2022 12:00:00 AM
7/27/2022	07/01/2022	7/1/2022 12:00:00 AM
8/2/2022	07/01/2022	7/1/2022 12:00:00 AM
8/8/2022	07/01/2022	7/1/2022 12:00:00 AM
8/19/2022	07/01/2022	7/1/2022 12:00:00 AM
5/1/2022	05/01/2022	5/1/2022 12:00:00 AM
5/7/2022	05/01/2022	5/1/2022 12:00:00 AM
5/22/2022	05/01/2022	5/1/2022 12:00:00 AM
6/15/2022	05/01/2022	5/1/2022 12:00:00 AM
6/26/2022	05/01/2022	5/1/2022 12:00:00 AM
3/7/2022	03/01/2022	3/1/2022 12:00:00 AM
3/30/2022	03/01/2022	3/1/2022 12:00:00 AM
4/5/2022	03/01/2022	3/1/2022 12:00:00 AM
4/16/2022	03/01/2022	3/1/2022 12:00:00 AM
2/19/2022	01/01/2022	1/1/2021 12:00:00 AM

Диаграмма функции `monthsstart()`, пример с объектом диаграммы



Транзакция 8195 совершена 22 мая. Функция `monthsstart()` изначально делит год на двухмесячные сегменты. Транзакция 8195 попадает в сегмент с 1 мая по 30 июня. Таким образом, функция возвращает первую миллисекунду этого сегмента, то есть 05/01/2022 12:00:00 AM.

Пример 5. Сценарий

Скрипт загрузки и выражение диаграммы

Обзор

Откройте редактор загрузки данных и добавьте приведенный ниже скрипт загрузки на новую вкладку.

Скрипт загрузки содержит следующее:

- Набор данных, содержащий набор остатков по кредитам, загруженный в таблицу под именем `Loans`.
- Данные, включая идентификаторы кредитов, остаток на начало месяца и простая процентная ставка, взимаемая по каждому кредиту за год.

Конечному пользователю требуется объект диаграммы, который будет отображать по идентификатору кредита текущий процент, начисленный по каждому кредиту за выбранный период. Финансовый год начинается в январе.

Скрипт загрузки

```
SET DateFormat='MM/DD/YYYY';

Loans:
Load
*
Inline
[
loan_id,start_balance,rate
8188,$10000.00,0.024
8189,$15000.00,0.057
8190,$17500.00,0.024
8191,$21000.00,0.034
8192,$90000.00,0.084
];
```

Результаты

Загрузите данные и откройте лист.

В начале скрипта загрузки создана переменная (`vPeriod`), которая будет привязана к элементу управления вводом переменной. Затем настройте переменную как настраиваемый объект на листе.

Выполните следующие действия.

1. На панели ресурсов щелкните **Пользовательские объекты**.
2. Выберите **Qlik Dashboard bundle**и создайте объект **Ввод переменной**.

3. Введите заголовок для объекта диаграммы.
4. В разделе **Переменная** выберите **vPeriod** в качестве имени и задайте для отображения объект **Раскрывающийся список**.
5. В разделе **Значения** настройте для объекта использование динамических значений. Введите следующее:
   `='1~month|2~bi-month|3~quarter|4~tertial|6~half-year'`

Затем создайте таблицу результатов.

Выполните следующие действия.

1. Создайте новую таблицу. Добавьте следующие поля как измерения:
   - `employee_id`
   - `employee_name`
2. Чтобы рассчитать накопленный процент, создайте меру:
   `=start_balance*(rate*(today(1)-monthsstart($(vPeriod),today(1)))/365)`
3. Задайте параметру меры **Формат чисел** значение **Денежный**. Щелкните ✔ **Изменение завершено**. Теперь можно модифицировать данные, показанные в таблице, изменив сегмент времени в объекте переменной.

Вот как будет выглядеть таблица результатов при выборе периода `month`:

Результирующая таблица

loan_id	start_balance	=start_balance*(rate*(today(1)-monthsstart($(vPeriod),today(1)))/365)
8188	$10000.00	$7.95
8189	$15000.00	$67.93
8190	$17500.00	$33.37
8191	$21000.00	$56.73
8192	$90000.00	$600.66

Функция `monthsstart()`, используя ввод пользователя в качестве первого аргумента и сегодняшнюю дату в качестве второго аргумента, возвращает дату начала выбранного пользователем периода. Вычитая этот результат из текущей даты, выражение возвращает количество дней, прошедших до сих пор в течение этого периода.

Затем это значение умножается на процентную ставку и делится на 365, чтобы получить эффективную процентную ставку начисленную за этот период. После этого результат умножается на начальный остаток кредита, чтобы вернуть проценты, начисленные до сих пор в течение этого периода.

monthstart

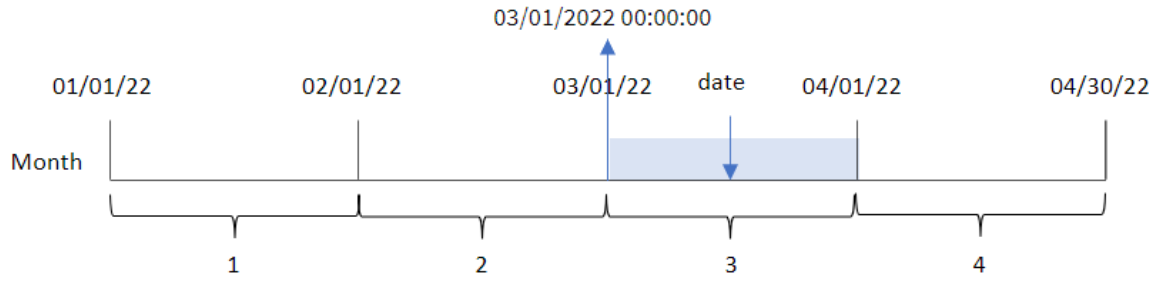
Эта функция возвращает значение, соответствующее метке времени, включающей первую миллисекунду первого дня месяца, содержащего значение, указанное в поле **date**. По умолчанию для вывода используется формат **DateFormat**, установленный в скрипте.

Синтаксис:

```
MonthStart(date[, period_no])
```

Возвращаемые типы данных: двойное значение

Схема функции `monthstart()`



Функция `monthstart()` определяет, на какой месяц приходится дата. Затем она возвращает метку времени в формате даты для первой миллисекунды этого месяца.

Аргументы

Аргумент	Описание
date	Дата или метка времени 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.

Когда это следует использовать

Функция `monthstart()` широко используется в составе выражения, когда пользователю требуется учитывать в расчетах часть месяца, которая уже прошла. Например, можно ее использовать, если требуется рассчитать проценты, накопленные в течение месяца до определенной даты.

Примеры функции

Пример	Результат
`monthstart('10/19/2001')`	Возвращает 10/01/2001.
`monthstart('10/19/2001', -1)`	Возвращает 09/01/2001.

Региональные настройки

Если не указано иное, в примерах, приведенных в данном разделе, используется следующий формат даты: MM/ДД/ГГГГ. Формат даты указан в операторе `SET DateFormat` скрипта загрузки данных. В вашей системе может быть установлен другой формат даты по умолчанию в зависимости от региональных настроек и других факторов. Можно изменить формат в примерах в соответствии с потребностями. Или можно изменить форматы в скрипте загрузки в соответствии с этими примерами.

Региональные настройки по умолчанию в приложениях основаны на системных региональных настройках компьютера или сервера, где установлено ПО Qlik Sense. Если на сервере Qlik Sense, к которому обращается пользователь, выбран шведский язык, то редактор загрузки данных будет использовать шведские региональные настройки для даты, времени и валюты. Эти параметры регионального формата не связаны с языком, отображаемым в интерфейсе пользователя Qlik Sense. Qlik Sense будет отображаться на языке, который используется в браузере.

Пример 1. Без дополнительных аргументов

Скрипт загрузки и результаты

Обзор

Откройте редактор загрузки данных и добавьте приведенный ниже скрипт загрузки на новую вкладку.

Скрипт загрузки содержит следующее:

- Набор данных, содержащий набор транзакций за 2022 год, загружается в таблицу под именем `Transactions`.
- Поле даты было предоставлено в формате системной переменной `DateFormat` (MM/DD/YYYY).
- Создание поля `start_of_month`, возвращающего метку времени начала месяца, в течение которого совершены транзакции.

Скрипт загрузки

```
SET DateFormat='MM/DD/YYYY';

Transactions:
    Load
        *,
        monthstart(date) as start_of_month,
        timestamp(monthstart(date)) as start_of_month_timestamp
        ;
Load
*
Inline
[
id,date,amount
8188,1/7/2022,17.17
```

```
8189,1/19/2022,37.23
8190,2/28/2022,88.27
8191,2/5/2022,57.42
8192,3/16/2022,53.80
8193,4/1/2022,82.06
8194,5/7/2022,40.39
8195,5/16/2022,87.21
8196,6/15/2022,95.93
8197,6/26/2022,45.89
8198,7/9/2022,36.23
8199,7/22/2022,25.66
8200,7/23/2022,82.77
8201,7/27/2022,69.98
8202,8/2/2022,76.11
8203,8/8/2022,25.12
8204,8/19/2022,46.23
8205,9/26/2022,84.21
8206,10/14/2022,96.24
8207,10/29/2022,67.67
];
```

Результаты

Загрузите данные и откройте лист. Создайте новую таблицу и добавьте эти поля как измерения:

- `date`
- `start_of_month`
- `start_of_month_timestamp`

Результирующая таблица

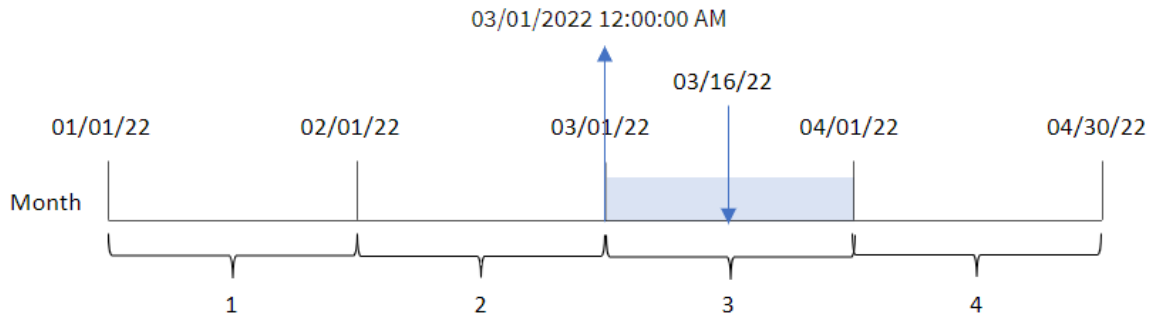
date	start_of_month	start_of_month_timestamp
1/7/2022	01/01/2022	1/1/2022 12:00:00 AM
1/19/2022	01/01/2022	1/1/2022 12:00:00 AM
2/5/2022	02/01/2022	2/1/2022 12:00:00 AM
2/28/2022	02/01/2022	2/1/2022 12:00:00 AM
3/16/2022	03/01/2022	3/1/2022 12:00:00 AM
4/1/2022	04/01/2022	4/1/2022 12:00:00 AM
5/7/2022	05/01/2022	5/1/2022 12:00:00 AM
5/16/2022	05/01/2022	5/1/2022 12:00:00 AM
6/15/2022	06/01/2022	6/1/2022 12:00:00 AM
6/26/2022	07/01/2022	6/1/2022 12:00:00 AM
7/9/2022	07/01/2022	7/1/2022 12:00:00 AM
7/22/2022	07/01/2022	7/1/2022 12:00:00 AM

date	start_of_month	start_of_month_timestamp
7/23/2022	07/01/2022	7/1/2022 12:00:00 AM
7/27/2022	07/01/2022	7/1/2022 12:00:00 AM
8/2/2022	08/01/2022	8/1/2022 12:00:00 AM
8/8/2022	08/01/2022	8/1/2022 12:00:00 AM
8/19/2022	08/01/2022	8/1/2022 12:00:00 AM
9/26/2022	09/01/2022	9/1/2022 12:00:00 AM
10/14/2022	10/01/2022	10/1/2022 12:00:00 AM
10/29/2022	10/01/2022	10/1/2022 12:00:00 AM

Поле `start_of_month` создано предшествующим оператором load с использованием функции `monthstart()`, в качестве аргумента которой передано поле даты.

Функция `monthstart()` определяет, к какому месяцу относится значение даты, и возвращает метку времени для первой миллисекунды этого месяца.

Диаграмма функции `monthstart()`, пример без дополнительных аргументов



Транзакция 8192 совершена 16 марта. Функция `monthstart()` возвращает первую миллисекунду этого месяца, то есть 00:00:00 (12:00:00 AM) 1 марта.

Пример 2. Скрипт period_no

Скрипт загрузки и результаты

Обзор

Откройте редактор загрузки данных и добавьте приведенный ниже скрипт загрузки на новую вкладку.

Скрипт загрузки содержит следующее:

- Используется тот же набор данных и сценарий, что в первом примере.
- Создание поля `previous_month_start`, которое возвращает метку времени начала месяца, предшествующего месяцу совершения транзакции.

Скрипт загрузки

```
SET DateFormat='MM/DD/YYYY';

Transactions:
    Load
        *,
        monthstart(date,-1) as previous_month_start,
        timestamp(monthstart(date,-1)) as previous_month_start_timestamp
        ;
Load
*
Inline
[
id,date,amount
8188,1/7/2022,17.17
8189,1/19/2022,37.23
8190,2/28/2022,88.27
8191,2/5/2022,57.42
8192,3/16/2022,53.80
8193,4/1/2022,82.06
8194,5/7/2022,40.39
8195,5/16/2022,87.21
8196,6/15/2022,95.93
8197,6/26/2022,45.89
8198,7/9/2022,36.23
8199,7/22/2022,25.66
8200,7/23/2022,82.77
8201,7/27/2022,69.98
8202,8/2/2022,76.11
8203,8/8/2022,25.12
8204,8/19/2022,46.23
8205,9/26/2022,84.21
8206,10/14/2022,96.24
8207,10/29/2022,67.67
];
```

Результаты

Загрузите данные и откройте лист. Создайте новую таблицу и добавьте эти поля как измерения:

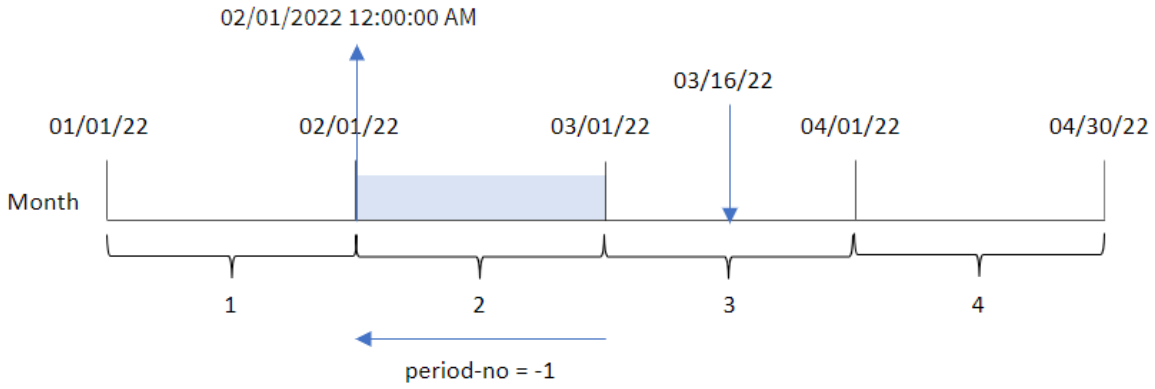
- `date`
- `previous_month_start`
- `previous_month_start_timestamp`

Результирующая таблица

date	previous_month_start	previous_month_start_timestamp
1/7/2022	12/01/2021	12/1/2021 12:00:00 AM
1/19/2022	12/01/2021	12/1/2021 12:00:00 AM
2/5/2022	01/01/2022	1/1/2022 12:00:00 AM
2/28/2022	01/01/2022	1/1/2022 12:00:00 AM
3/16/2022	02/01/2022	2/1/2022 12:00:00 AM
4/1/2022	03/01/2022	3/1/2022 12:00:00 AM
5/7/2022	04/01/2022	4/1/2022 12:00:00 AM
5/16/2022	04/01/2022	4/1/2022 12:00:00 AM
6/15/2022	05/01/2022	5/1/2022 12:00:00 AM
6/26/2022	05/01/2022	5/1/2022 12:00:00 AM
7/9/2022	06/01/2022	6/1/2022 12:00:00 AM
7/22/2022	06/01/2022	6/1/2022 12:00:00 AM
7/23/2022	06/01/2022	6/1/2022 12:00:00 AM
7/27/2022	06/01/2022	6/1/2022 12:00:00 AM
8/2/2022	07/01/2022	7/1/2022 12:00:00 AM
8/8/2022	07/01/2022	7/1/2022 12:00:00 AM
8/19/2022	07/01/2022	7/1/2022 12:00:00 AM
9/26/2022	08/01/2022	8/1/2022 12:00:00 AM
10/14/2022	09/01/2022	9/1/2022 12:00:00 AM
10/29/2022	09/01/2022	9/1/2022 12:00:00 AM

В этом случае, так как в качестве аргумента смещения в функции `monthstart()` использовалось `period_no` = -1, функция сначала определяет месяц, в котором совершаются транзакции. Затем она возвращается на месяц назад и определяет первую миллисекунду этого месяца.

Диаграмма функции `monthstart()`, пример с аргументом period_no



Транзакция 8192 совершена 16 марта. Функция `monthstart()` определяет, что месяцем, предшествующим транзакции, является февраль. Затем она возвращает первую миллисекунду этого месяца, 1 февраля в 00:00:00 (12:00:00 AM).

Пример 3. Пример объекта диаграммы

Скрипт загрузки и выражение диаграммы

Обзор

Откройте редактор загрузки данных и добавьте приведенный ниже скрипт загрузки на новую вкладку.

Скрипт загрузки содержит тот же набор данных и сценарий, что в первом примере.

Однако в этом примере в приложение загружается неизмененный набор данных. Расчет, возвращающий метку времени начала месяца, в котором совершены транзакции, создается как мера в объекте диаграммы в приложении.

Скрипт загрузки

```
SET DateFormat='MM/DD/YYYY';

Transactions:
Load
*
Inline
[
id,date,amount
8188,1/7/2022,17.17
8189,1/19/2022,37.23
8190,2/28/2022,88.27
8191,2/5/2022,57.42
8192,3/16/2022,53.80
8193,4/1/2022,82.06
8194,5/7/2022,40.39
```

```
8195,5/16/2022,87.21
8196,6/15/2022,95.93
8197,6/26/2022,45.89
8198,7/9/2022,36.23
8199,7/22/2022,25.66
8200,7/23/2022,82.77
8201,7/27/2022,69.98
8202,8/2/2022,76.11
8203,8/8/2022,25.12
8204,8/19/2022,46.23
8205,9/26/2022,84.21
8206,10/14/2022,96.24
8207,10/29/2022,67.67
];
```

Результаты

Загрузите данные и откройте лист. Создайте новую таблицу и добавьте это поле как измерение: `date`.

Чтобы рассчитать дату начала месяца, когда была совершена транзакция, создайте следующие меры:

- `=monthstart(date)`
- `=timestamp(monthstart(date))`

Результирующая таблица

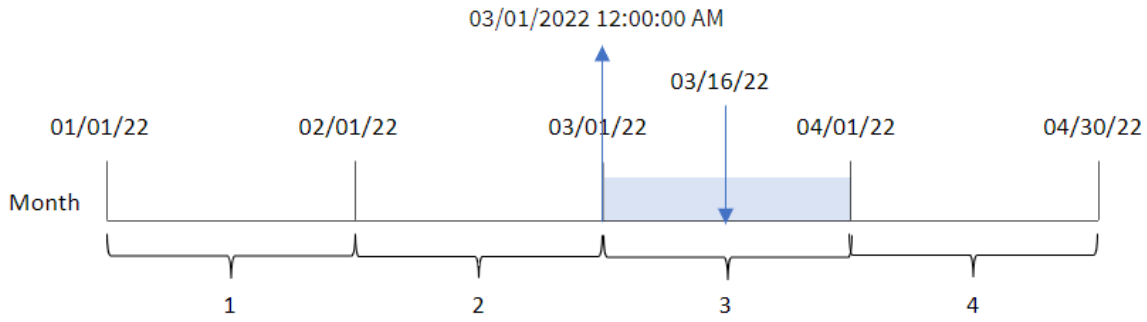
date	=monthstart(date)	=timestamp(monthstart(date))
10/14/2022	10/01/2022	10/1/2022 12:00:00 AM
10/29/2022	10/01/2022	10/1/2022 12:00:00 AM
9/26/2022	09/01/2022	9/1/2022 12:00:00 AM
8/2/2022	08/01/2022	8/1/2022 12:00:00 AM
8/8/2022	08/01/2022	8/1/2022 12:00:00 AM
8/19/2022	08/01/2022	8/1/2022 12:00:00 AM
7/9/2022	07/01/2022	7/1/2022 12:00:00 AM
7/22/2022	07/01/2022	7/1/2022 12:00:00 AM
7/23/2022	07/01/2022	7/1/2022 12:00:00 AM
7/27/2022	07/01/2022	7/1/2022 12:00:00 AM
6/15/2022	06/01/2022	6/1/2022 12:00:00 AM
6/26/2022	06/01/2022	6/1/2022 12:00:00 AM
5/7/2022	05/01/2022	5/1/2022 12:00:00 AM
5/16/2022	05/01/2022	5/1/2022 12:00:00 AM

date	=monthstart(date)	=timestamp(monthstart(date))
4/1/2022	04/01/2022	4/1/2022 12:00:00 AM
3/16/2022	03/01/2022	3/1/2022 12:00:00 AM
2/5/2022	02/01/2022	2/1/2022 12:00:00 AM
2/28/2022	02/01/2022	2/1/2022 12:00:00 AM
1/7/2022	01/01/2022	1/1/2022 12:00:00 AM
1/19/2022	01/01/2022	1/1/2022 12:00:00 AM

Мера `start_of_month` создана в объекте диаграммы с использованием функции `monthstart()`, где в качестве аргумента функции передано поле даты.

Функция `monthstart()` определяет, к какому месяцу относится значение даты, и возвращает метку времени для первой миллисекунды этого месяца.

Диаграмма функции `monthstart()`, пример с объектом диаграммы



Транзакция 8192 совершена 16 марта. Функция `monthstart()` определяет, что транзакция совершена в марте и возвращает первую миллисекунду этого месяца, то есть 00:00:00 (12:00:00 AM) 1 марта.

Пример 4. Сценарий

Скрипт загрузки и выражение диаграммы

Обзор

Откройте редактор загрузки данных и добавьте приведенный ниже скрипт загрузки на новую вкладку.

Скрипт загрузки содержит следующее:

- Набор данных, содержащий набор остатков по кредитам, загруженный в таблицу под именем `Loans`.

- Данные, включая идентификаторы кредитов, остаток на начало месяца и простая процентная ставка, взимаемая по каждому кредиту за год.

Конечному пользователю требуется объект диаграммы, который будет отображать по идентификатору кредита текущий процент, начисленный по каждому кредиту в течение месяца до текущей даты.

Скрипт загрузки

```
SET DateFormat='MM/DD/YYYY';

Loans:
Load
*
Inline
[
loan_id,start_balance,rate
8188,$10000.00,0.024
8189,$15000.00,0.057
8190,$17500.00,0.024
8191,$21000.00,0.034
8192,$90000.00,0.084
];
```

Результаты

Выполните следующие действия.

1. Загрузите данные и откройте лист. Создайте новую таблицу и добавьте эти поля как измерения:
   - `loan_id`
   - `start_balance`
2. Затем создайте меру, чтобы рассчитать накопленный процент:
   `=start_balance*(rate*(today(1)-monthstart(today(1)))/365)`
3. Задайте параметру меры **Формат чисел** значение **Денежный**.

Результирующая таблица

loan_id	start_balance	=start_balance*(rate*(today(1)-monthstart(today(1)))/365)
8188	$10000.00	$16.44
8189	$15000.00	$58.56
8190	$17500.00	$28.77
8191	$21000.00	$48.90
8192	$90000.00	$517.81

Используя сегодняшнюю дату в качестве единственного аргумента, функция `monthstart()` возвращает дату начала текущего месяца. Вычитая этот результат из текущей даты, выражение возвращает количество дней, прошедших до сих пор в течение этого месяца.

Затем это значение умножается на процентную ставку и делится на 365, чтобы получить эффективную процентную ставку начисленную за этот период. После этого результат умножается на начальный остаток кредита, чтобы вернуть проценты, начисленные до сих пор в течение этого месяца.

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

Диаграмма календаря с диапазоном дат, возвращенным функцией networkdays

Sun	Mon	Tue	Wed	Thu	Fri	Sat
	1	2	3	4	5	6
7	8	9	10 start_date	11	12	13
14	15	16	17	18	19	20
21	22	23	24	25	26 end_date	27
28	29	30	31			

Функция `networkdays` имеет следующие ограничения:

- Отсутствует возможность изменять рабочие дни. Другими словами, не предусмотрено способа изменять функцию для регионов или ситуаций, когда рабочая неделя отличается от стандартных рабочих дней, с понедельника по пятницу.
- Параметр `holiday` должен быть постоянной строкой. Выражения не поддерживаются.

Аргументы

Аргумент	Описание
start_date	Начальная дата для вычисления.
end_date	Конечная дата для вычисления.
holiday	Периоды выходных дней для исключения из рабочих дней. Праздник обозначен как постоянная строка даты. Можно указать несколько дат праздников, разделенных запятыми. **Пример:** `'12/25/2013', '12/26/2013', '12/31/2013', '01/01/2014'`

Когда это следует использовать

Функция `networkdays()` широко используется в составе выражения, когда пользователю требуется учитывать в расчетах количество рабочих дней между двумя датами. Например, если пользователю требуется вычислить совокупную заработную плату, заработанную сотрудником по договору PAYE (отчисление подоходного налога из зарплаты).

Примеры функции

Пример	Результат
`networkdays ('12/19/2013', '01/07/2014')`	Возвращает 14. В этом примере выходные дни не учитываются.
`networkdays ('12/19/2013', '01/07/2014', '12/25/2013', '12/26/2013')`	Возвращает 12. В этом примере учитываются выходные в периоде с 12/25/2013 по 12/26/2013.
`networkdays ('12/19/2013', '01/07/2014', '12/25/2013', '12/26/2013', '12/31/2013', '01/01/2014')`	Возвращает 10. В этом примере учитываются двухдневные периоды выходных дней.

Региональные настройки

Если не указано иное, в примерах, приведенных в данном разделе, используется следующий формат даты: ММ/ДД/ГГГГ. Формат даты указан в операторе `SET DateFormat` скрипта загрузки данных. В вашей системе может быть установлен другой формат даты по умолчанию в зависимости от региональных настроек и других факторов. Можно изменить формат в примерах в соответствии с потребностями. Или можно изменить форматы в скрипте загрузки в соответствии с этими примерами.

Региональные настройки по умолчанию в приложениях основаны на системных региональных настройках компьютера или сервера, где установлено ПО Qlik Sense. Если на сервере Qlik Sense, к которому обращается пользователь, выбран шведский язык, то редактор загрузки данных будет использовать шведские региональные настройки для даты, времени и валюты. Эти параметры регионального формата не связаны с языком, отображаемым в интерфейсе пользователя Qlik Sense. Qlik Sense будет отображаться на языке, который используется в браузере.

Пример 1. Базовый пример

Скрипт загрузки и результаты

Обзор

Откройте редактор загрузки данных и добавьте приведенный ниже скрипт загрузки на новую вкладку.

Скрипт загрузки содержит следующее:

- Набор данных, содержащий идентификаторы проектов, их даты начала и окончания. Эта информация загружается в таблицу под именем `Projects`.
- Поле даты было предоставлено в формате системной переменной `DateFormat` (MM/DD/YYYY).
- Создание дополнительного поля `net_work_days` для вычисления количества рабочих дней, необходимых для выполнения каждого проекта.

Скрипт загрузки

```
SET DateFormat='MM/DD/YYYY';

Projects:
    Load
        *,
        networkdays(start_date,end_date) as net_work_days
        ;
Load
id,
start_date,
end_date
Inline
[
id,start_date,end_date
1,01/01/2022,01/18/2022
2,02/10/2022,02/17/2022
3,05/17/2022,07/05/2022
4,06/01/2022,06/12/2022
5,08/10/2022,08/26/2022
];
```

Результаты

Загрузите данные и откройте лист. Создайте новую таблицу и добавьте эти поля как измерения:

- `id`
- `start_date`
- `end_date`
- `net_work_days`

Результирующая таблица

id	start_date	end_date	net_work_days
1	01/01/2022	01/18/2022	12
2	02/10/2022	02/17/2022	6
3	05/17/2022	07/05/2022	36
4	06/01/2022	06/12/2022	8
5	08/10/2022	08/26/2022	13

Так как на этот период не запланированы праздники (они бы указывались в третьем аргументе функции `networkdays()`), функция вычитает `start_date` из `end_date`, а также выходные дни, чтобы вычислить количество рабочих дней между двумя датами.

Диаграмма календаря, в котором выделены рабочие дни для проекта 5 (без праздников)

Sun	Mon	Tue	Wed	Thu	Fri	Sat
	1	2	3	4	5	6
7	8	9	10	11	12	13
14	15	16	17	18	19	20
21	22	23	24	25	26	27
28	29	30	31			

Приведенный выше календарь наглядно показывает проект с `id` = 5. Проект 5 начинается в среду, 10 августа 2022 года, и заканчивается 26 августа 2022 года. Так как все субботы и воскресенья игнорируются, между этими двумя датами насчитывается 13 рабочих дней.

Пример 2. Один праздник

Скрипт загрузки и результаты

Обзор

Откройте редактор загрузки данных и добавьте приведенный ниже скрипт загрузки на новую вкладку.

Скрипт загрузки содержит следующее:

- Используется тот же набор данных и сценарий, что и в предыдущем примере.
- Поле даты было предоставлено в формате системной переменной `DateFormat` (MM/DD/YYYY).
- Создание дополнительного поля `net_work_days` для вычисления количества рабочих дней, необходимых для выполнения каждого проекта.

В этом примере на 19 августа 2022 года запланирован один праздничный день.

Скрипт загрузки

```
SET DateFormat='MM/DD/YYYY';

Projects:
    Load
        *,
        networkdays(start_date,end_date,'08/19/2022') as net_work_days
        ;
Load
id,
start_date,
end_date
Inline
[
id,start_date,end_date
1,01/01/2022,01/18/2022
2,02/10/2022,02/17/2022
3,05/17/2022,07/05/2022
4,06/01/2022,06/12/2022
5,08/10/2022,08/26/2022
];
```

Результаты

Загрузите данные и откройте лист. Создайте новую таблицу и добавьте эти поля как измерения:

- `id`
- `start_date`
- `end_date`
- `net_work_days`

Результирующая таблица

id	start_date	end_date	net_work_days
1	01/01/2022	01/18/2022	12
2	02/10/2022	02/17/2022	6
3	05/17/2022	07/05/2022	36
4	06/01/2022	06/12/2022	8
5	08/10/2022	08/26/2022	12

Один запланированный праздник передан в качестве третьего аргумента функции `networkdays()`.

Диаграмма календаря, в котором выделены рабочие дни для проекта 5 (один праздник)

Приведенный выше календарь наглядно показывает проект 5, демонстрируя эту корректировку с учетом праздника. Этот праздник запланирован в период выполнения проекта 5 – в пятницу, 19 августа 2022 года. В результате этого, общее значение `net_work_days` для проекта 5 уменьшается на один день: с 13 до 12 дней.

Пример 3. Несколько праздников

Скрипт загрузки и результаты

Обзор

Откройте редактор загрузки данных и добавьте приведенный ниже скрипт загрузки на новую вкладку.

Скрипт загрузки содержит следующее:

- Используется тот же набор данных и сценарий из первого примера.
- Поле даты было предоставлено в формате системной переменной `DateFormat` (MM/DD/YYYY).
- Создание дополнительного поля `net_work_days` для вычисления количества рабочих дней, необходимых для выполнения каждого проекта.

Однако в этом примере запланировано четыре праздничных дня с 18 по 21 августа 2022 года.

Скрипт загрузки

```
SET DateFormat='MM/DD/YYYY';

Projects:
    Load
        *,
        networkdays(start_date,end_date,'08/18/2022','08/19/2022','08/20/2022','08/21/2022')
as net_work_days
        ;
Load
id,
start_date,
end_date
Inline
[
id,start_date,end_date
1,01/01/2022,01/18/2022
2,02/10/2022,02/17/2022
3,05/17/2022,07/05/2022
4,06/01/2022,06/12/2022
5,08/10/2022,08/26/2022
];
```

Результаты

Загрузите данные и откройте лист. Создайте новую таблицу и добавьте эти поля как измерения:

- `id`
- `start_date`

- `end_date`
- `net_work_days`

Результирующая таблица

id	start_date	end_date	net_work_days
1	01/01/2022	01/18/2022	12
2	02/10/2022	02/17/2022	6
3	05/17/2022	07/05/2022	36
4	06/01/2022	06/12/2022	8
5	08/10/2022	08/26/2022	11

Четыре запланированных праздничных дня вводятся в виде списка, разделенного запятыми, начиная с третьего аргумента в функции `networkdays()`.

Диаграмма календаря, в котором выделены рабочие дни для проекта 5 (несколько праздников)

Sun	Mon	Tue	Wed	Thu	Fri	Sat
	1	2	3	4	5	6
7	8	9	10	11	12	13
14	15	16	17	18 Holiday	19 Holiday	20
21	22	23	24	25	26	27
28	29	30	31			

Приведенный выше календарь наглядно показывает проект 5, демонстрируя эту корректировку с учетом этих праздничных дней. Эти запланированные праздники попадают в период выполнения проекта 5, два из них выпадают на четверг и пятницу. В результате этого общее значение `net_work_days` для проекта 5 уменьшается на два дня: с 13 до 11 дней.

Пример 4. Один праздник

Скрипт загрузки и выражение диаграммы

Обзор

Откройте редактор загрузки данных и добавьте приведенный ниже скрипт загрузки на новую вкладку.

Скрипт загрузки содержит следующее:

- Используется тот же набор данных и сценарий из первого примера.
- Поле даты было предоставлено в формате системной переменной `DateFormat` (MM/DD/YYYY).

На 19 августа 2022 года запланирован один праздничный день.

Однако в этом примере в приложение загружается неизмененный набор данных. Значение поля `net_work_days` вычисляется с использованием меры в объекте диаграммы.

Скрипт загрузки

```
SET DateFormat='MM/DD/YYYY';

Projects:
Load
id,
start_date,
end_date
Inline
[
id,start_date,end_date
1,01/01/2022,01/18/2022
2,02/10/2022,02/17/2022
3,05/17/2022,07/05/2022
4,06/01/2022,06/12/2022
5,08/10/2022,08/26/2022
];
```

Результаты

Загрузите данные и откройте лист. Создайте новую таблицу и добавьте эти поля как измерения:

- `id`
- `start_date`
- `end_date`

Создайте следующую меру:

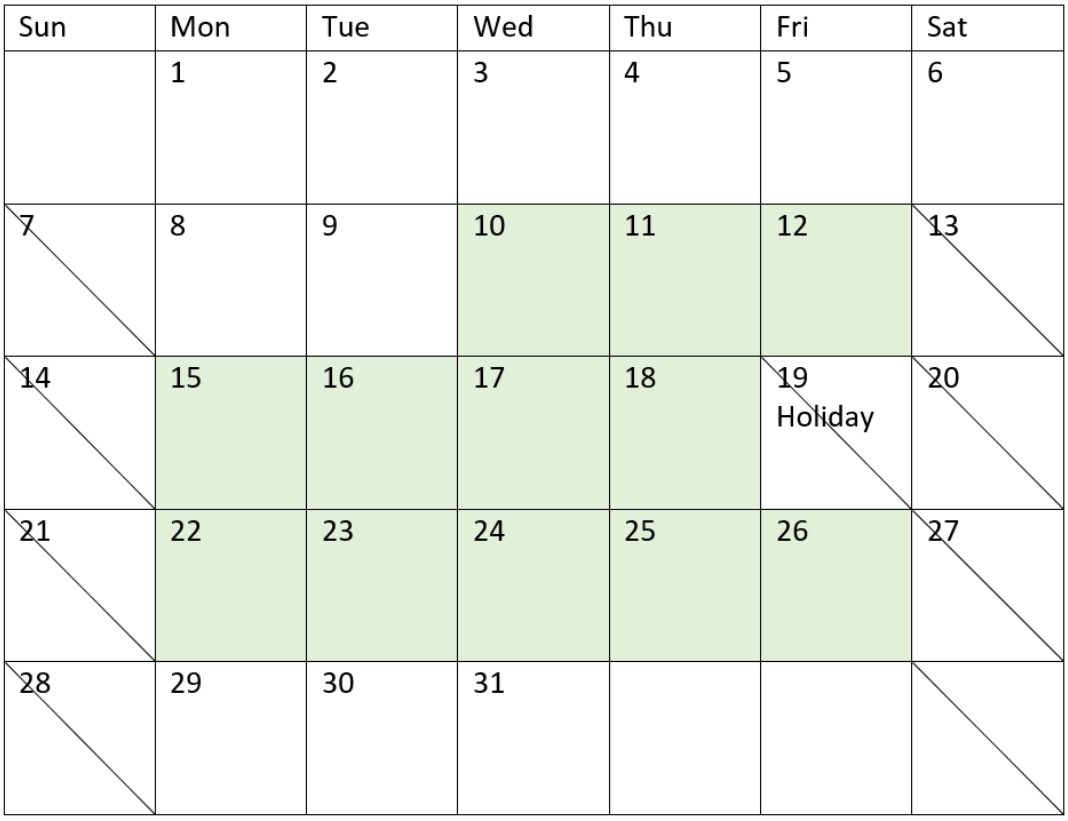
```
= networkdays(start_date,end_date,'08/19/2022')
```

Результирующая таблица

id	start_date	end_date	net_work_days
1	01/01/2022	01/18/2022	12
2	02/10/2022	02/17/2022	6
3	05/17/2022	07/05/2022	36
4	06/01/2022	06/12/2022	8
5	08/10/2022	08/26/2022	12

Один запланированный праздник передан в качестве третьего аргумента функции `networkdays()`.

Диаграмма календаря, демонстрирующая полезные рабочие дни с учетом одного праздника (объект диаграммы)



Приведенный выше календарь наглядно показывает проект 5, демонстрируя эту корректировку с учетом праздника. Этот праздник запланирован в период выполнения проекта 5 — в пятницу, 19 августа 2022 года. В результате этого общее значение `net_work_days` для проекта 5 уменьшается на один день: с 13 до 12 дней.

now

Эта функция возвращает метку текущего времени. Эта функция возвращает значения в формате системной переменной **TimeStamp**. Значение **timer_mode** по умолчанию – 1.

Синтаксис:

```
now([ timer_mode])
```

Возвращаемые типы данных: двойное значение

Функцию `now()` можно использовать в скрипте загрузки или в объектах диаграммы.

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.*

Когда это следует использовать

Функция `now()` часто используется как компонент внутри выражения. Например, ее можно использовать для вычисления оставшегося времени жизненного цикла продукта. Функцию `now()` рекомендуется использовать вместо функции `today()`, когда в выражении требуется учитывать часть дня.

В следующей таблице приводится объяснение результата, возвращаемого функцией `now()`, в зависимости от различных значений аргумента `timer_mode`:

Примеры функции

Значение timer_mode	Результат при использовании в скрипте загрузки	Результат при использовании в объекте диаграммы
0	Возвращает метку времени в формате системной переменной `TimeStamp` для даты последней успешной перезагрузки, предшествующей самой последней перезагрузке данных.	Возвращает метку времени в формате системной переменной `TimeStamp` для даты последней перезагрузки данных.
1	Возвращает метку времени в формате системной переменной `TimeStamp` для самой последней перезагрузки данных.	Возвращает метку времени вызова функции в формате системной переменной `TimeStamp`.
2	Возвращает метку времени в формате системной переменной `TimeStamp` для начала сеанса пользователя в приложении. Это значение не будет обновляться, если пользователь не перезагрузит скрипт.	Возвращает метку времени в формате системной переменной `TimeStamp` для начала сеанса пользователя в приложении. Это значение будет обновляться в случае начала нового сеанса или перезагрузки данных в приложение.

Региональные настройки

Если не указано иное, в примерах, приведенных в данном разделе, используется следующий формат даты: ММ/ДД/ГГГГ. Формат даты указан в операторе `SET DateFormat` скрипта загрузки данных. В вашей системе может быть установлен другой формат даты по умолчанию в зависимости от региональных настроек и других факторов. Можно изменить формат в примерах в соответствии с потребностями. Или можно изменить форматы в скрипте загрузки в соответствии с этими примерами.

Региональные настройки по умолчанию в приложениях основаны на системных региональных настройках компьютера или сервера, где установлено ПО Qlik Sense. Если на сервере Qlik Sense, к которому обращается пользователь, выбран шведский язык, то редактор загрузки данных будет использовать шведские региональные настройки для даты, времени и валюты. Эти параметры регионального формата не связаны с языком, отображаемым в интерфейсе пользователя Qlik Sense. Qlik Sense будет отображаться на языке, который используется в браузере.

Пример 1. Создание объектов с использованием скрипта загрузки

Скрипт загрузки и результаты

Обзор

Откройте редактор загрузки данных и добавьте приведенный ниже скрипт загрузки на новую вкладку.

Этот пример создает три переменные с использованием функции now(). Каждая переменная использует один из параметров timer_mode для демонстрации эффекта.

Чтобы продемонстрировать назначение переменных, перезагрузите скрипт, а затем, через короткий период времени, выполните вторую перезагрузку. Это приведет к тому, что переменные now(0) и now(1) будут показывать разные значения, тем самым правильно демонстрируя свое назначение.

Скрипт загрузки

```
LET vPreviousDataLoad = now(0);
LET vCurrentDataLoad = now(1);
LET vApplicationOpened = now(2);
```

Результаты

После загрузки данных во второй раз создайте три текстовых поля, следуя приведенным ниже инструкциям.

Сначала создайте текстовое поле для данных, загруженных ранее.

Выполните следующие действия.

1. Создайте текстовое поле, используя объект диаграммы **Текст и изображение**.
2. Добавьте в объект следующую меру:
   `=vPreviousDataLoad`
3. В области **Вид** выберите **Show titles** и добавьте в объект заголовок «Время предыдущей загрузки».

Затем создайте текстовое поле для данных, которые загружены в данный момент.

Выполните следующие действия.

1. Создайте текстовое поле, используя объект диаграммы **Текст и изображение**.
2. Добавьте в объект следующую меру:
   `=vCurrentDataLoad`
3. В области **Вид** выберите **Show titles** и добавьте в объект заголовок «Время текущей загрузки».

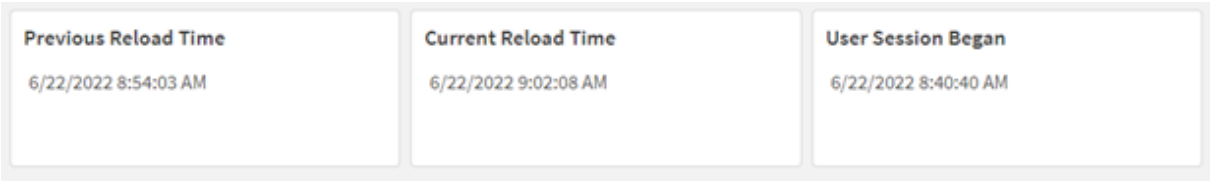
Создайте последнее текстовое поле, в котором будет отображаться время начала сеанса пользователя в приложении.

Выполните следующие действия.

1. Создайте текстовое поле, используя объект диаграммы **Текст и изображение**.
2. Добавьте в объект следующую меру:
   `=vApplicationOpened`

3. В области **Вид** выберите **Show titles** и добавьте в объект заголовок «Начало сеанса пользователя».

Переменные скрипта загрузки now()



В приведенной выше диаграмме показаны примерные значения для каждой из созданных переменных. Например, это могут быть следующие значения:

- Время предыдущей перезагрузки: 6/22/2022 8:54:03 AM
- Время текущей перезагрузки: 6/22/2022 9:02:08 AM
- Начало сеанса пользователя: 6/22/2022 8:40:40 AM

Пример 2. Создание объектов без использования скрипта загрузки

Скрипт загрузки и выражение диаграммы

Обзор

В этом примере используются три объекта диаграммы с помощью функции `now()` и без загрузки переменных или данных в приложение. Каждый объект диаграммы использует один из параметров `timer_mode` для демонстрации своего эффекта.

Для этого примера не предусмотрен скрипт загрузки.

Выполните следующие действия.

1. Откройте Редактор загрузки данных.
2. Не изменяя существующий скрипт загрузки, щелкните **Загрузить данные**.
3. Через короткий период времени загрузите скрипт повторно.

Результаты

После загрузки данных во второй раз создайте три текстовых поля.

Сначала создайте текстовое поле для последней перезагрузки данных.

Выполните следующие действия.

1. Создайте текстовое поле, используя объект диаграммы **Текст и изображение**.
2. Добавьте следующую меру:
   ```
   =now(0)
   ```

3. В области **Вид** выберите **Show titles** и добавьте в объект заголовок «Последняя перезагрузка данных».

Затем создайте текстовое поле для отображения текущего времени.

Выполните следующие действия.

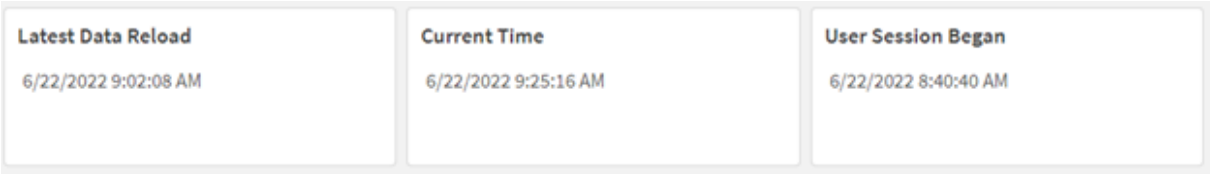
1. Создайте текстовое поле, используя объект диаграммы **Текст и изображение**.
2. Добавьте следующую меру:
   `=now(1)`
3. В области **Вид** выберите **Show titles** и добавьте в объект заголовок «Текущее время».

Создайте последнее текстовое поле, в котором будет отображаться время начала сеанса пользователя в приложении.

Выполните следующие действия.

1. Создайте текстовое поле, используя объект диаграммы **Текст и изображение**.
2. Добавьте следующую меру:
   `=now(2)`
3. В области **Вид** выберите **Show titles** и добавьте в объект заголовок «Начало сеанса пользователя».

Примеры объектов диаграммы `now()`



В приведенной выше диаграмме показаны примерные значения для каждого из созданных объектов. Например, это могут быть следующие значения:

- Последняя перезагрузка данных: 6/22/2022 9:02:08 AM
- Текущее время: 6/22/2022 9:25:16 AM
- Начало сеанса пользователя: 6/22/2022 8:40:40 AM

Объект диаграммы «Последняя перезагрузка данных» использует значение `timer_mode` = 0. В результате этого возвращается метка времени для последней успешной перезагрузки данных.

Объект диаграммы «Текущее время» использует значение `timer_mode` = 1. В результате этого возвращается текущее время по системным часам. В случае обновления листа или объекта это значение обновляется.

Объект диаграммы «Начало сеанса пользователя» использует `timer_mode` = 2. В результате этого возвращается метка времени, когда пользователь открыл приложение и начал сеанс.

Пример 3. Сценарий

Скрипт загрузки и выражение диаграммы

Обзор

Откройте редактор загрузки данных и добавьте приведенный ниже скрипт загрузки на новую вкладку.

Скрипт загрузки содержит следующее:

- Набор данных, содержащий инвентарь для операции майнинга криптовалюты, загруженный в таблицу под именем `Inventory`.
- Данные со следующими полями: `id`, `purchase_date` и `wph` (ватт в час).

Пользователю нужна таблица, в которой по `id` отображается общая стоимость каждой фермы для майнинга, начисленная за месяц до текущего момента с точки зрения энергопотребления.

Это значение должно обновляться при каждом обновлении объекта диаграммы. Текущая стоимость электричества составляет $0.0678 за кВт· ч.

Скрипт загрузки

```
SET DateFormat='MM/DD/YYYY';

Inventory:
Load
*
Inline
[
id,purchase_date,wph
8188,1/7/2022,1123
8189,1/19/2022,1432
8190,2/28/2022,1227
8191,2/5/2022,1322
8192,3/16/2022,1273
8193,4/1/2022,1123
8194,5/7/2022,1342
8195,5/16/2022,2342
8196,6/15/2022,1231
8197,6/26/2022,1231
8198,7/9/2022,1123
8199,7/22/2022,1212
8200,7/23/2022,1223
8201,7/27/2022,1232
8202,8/2/2022,1232
8203,8/8/2022,1211
8204,8/19/2022,1243
8205,9/26/2022,1322
8206,10/14/2022,1133
8207,10/29/2022,1231
];
```

Результаты

Загрузите данные и откройте лист. Создайте новую таблицу и добавьте это поле как измерение: `id`.

Создайте следующую меру:

```
=(now(1)-monthstart(now(1)))*24*wph/1000*0.0678
```

Если объект диаграммы был обновлен в 6/22/2022 10:39:05 AM, он должен вернуть следующие результаты:

Результирующая таблица

id	=(now(1)-monthstart(now(1)))*24*wph/1000*0.0678
8188	$39.18
8189	$49.97
8190	$42.81
8191	$46.13
8192	$44.42
8193	$39.18
8194	$46.83
8195	$81.72
8196	$42.95
8197	$42.95
8198	$39.18
8199	$42.29
8200	$42.67
8201	$42.99
8202	$42.99
8203	$42.25
8204	$43.37
8205	$46.13
8206	$39.53

Пользователю требуется, чтобы результаты объекта обновлялись каждый раз при обновлении самого объекта. Поэтому аргумент `timer_mode` предоставлен для экземпляров функции `now()` в приложении. Метка времени для начала месяца, определенного с использованием функции `now()` в качестве аргумента метки времени в функции `monthstart()`, вычитается из текущего времени, определенного с помощью функции `now()`. Это дает общее количество прошедшего времени в этом месяце, выраженное в днях.

Это значение умножается на 24 (количество часов в сутках), а затем на значение в поле `wph`.

Чтобы преобразовать значение из Вт· ч в кВт· ч, результат делится на 1000 перед заключительным умножением на тариф оплаты за кВт· ч.

quarterend

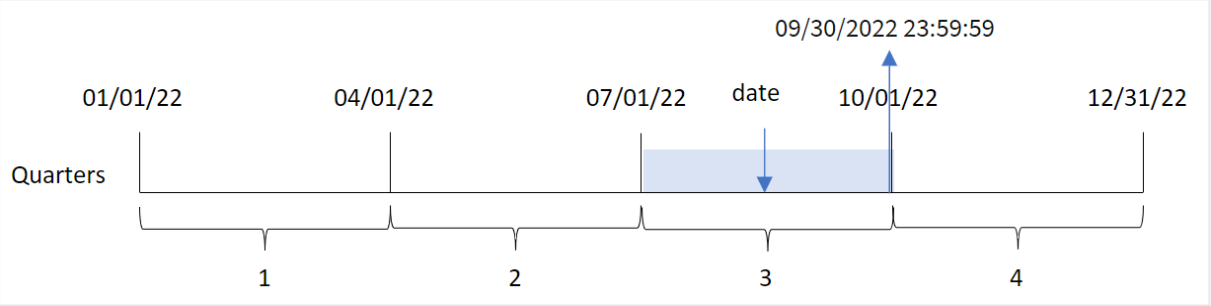
Эта функция возвращает значение, соответствующее метке времени, включающей последнюю миллисекунду квартала, содержащего значение, указанное в поле **date**. По умолчанию для вывода используется формат **DateFormat**, установленный в скрипте.

Синтаксис:

```
QuarterEnd(date[, period_no[, first_month_of_year]])
```

Возвращаемые типы данных: двойное значение

Диаграмма функции `quarterend()`



Функция `quarterend()` определяет, на какой квартал приходится дата. Затем она возвращает метку времени в формате даты для последней миллисекунды последнего месяца этого квартала. По умолчанию первым месяцем года является январь. Однако также можно изменить первый месяц, используя аргумент `first_month_of_year` в функции `quarterend()`.

> *Функция `quarterend()` не учитывает системную переменную `FirstMonthOfYear`. Год начинается 1 января, если для его изменения не используется аргумент `first_month_of_year`.*

Когда это следует использовать

Функция `quarterend()` широко используется в составе выражения, когда в расчетах требуется учитывать часть квартала, которая еще не прошла, например, если требуется рассчитать общую сумму процентов, еще не начисленных в течение квартала.

Аргументы

Аргумент	Описание
date	Дата или метка времени 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**.

Можно использовать следующие значения, чтобы задать первый месяц года в аргументе `first_month_of_year`:

значения first_month_of_year

Месяц	Значение
Февраль	2
Март	3
Апрель	4
Мау	5
Июнь	6
Июль	7
Август	8
Сентябрь	9
Октябрь	10
Ноябрь	11
Декабрь	12

Региональные настройки

Если не указано иное, в примерах, приведенных в данном разделе, используется следующий формат даты: ММ/ДД/ГГГГ. Формат даты указан в операторе `SET DateFormat` скрипта загрузки данных. В вашей системе может быть установлен другой формат даты по умолчанию в зависимости от региональных настроек и других факторов. Можно изменить формат в примерах в соответствии с потребностями. Или можно изменить форматы в скрипте загрузки в соответствии с этими примерами.

Региональные настройки по умолчанию в приложениях основаны на системных региональных настройках компьютера или сервера, где установлено ПО Qlik Sense. Если на сервере Qlik Sense, к которому обращается пользователь, выбран шведский язык, то редактор загрузки данных будет

использовать шведские региональные настройки для даты, времени и валюты. Эти параметры регионального формата не связаны с языком, отображаемым в интерфейсе пользователя Qlik Sense. Qlik Sense будет отображаться на языке, который используется в браузере.

Примеры функции

Пример	Результат
`quarterend('10/29/2005')`	Returns 12/31/2005 23:59:59.
`quarterend('10/29/2005', -1)`	Returns 09/30/2005 23:59:59.
`quarterend('10/29/2005', 0, 3)`	Returns 11/30/2005 23:59:59.

Пример 1. Базовый пример

Скрипт загрузки и результаты

Обзор

Откройте редактор загрузки данных и добавьте приведенный ниже скрипт загрузки на новую вкладку.

Скрипт загрузки содержит следующее:

- Набор данных, содержащий набор транзакций за 2022 год, загруженный в таблицу под именем `Transactions`.
- Предыдущая загрузка, которая содержит следующее:
  - Функция `quarterend()`, заданная как поле `end_of_quarter`, которая возвращает метку времени конца квартала, в течение которого совершены транзакции.
  - Функция `timestamp()`, заданная как поле `end_of_quarter_timestamp`, которая возвращает точную метку времени конца выбранного квартала.

Скрипт загрузки

```
SET DateFormat='MM/DD/YYYY';

Transactions:
    Load
       *,
       quarterend(date) as end_of_quarter,
       timestamp(quarterend(date)) as end_of_quarter_timestamp
       ;
Load
*
Inline
[
id,date,amount
8188,1/7/2022,17.17
8189,1/19/2022,37.23
8190,2/28/2022,88.27
8191,2/5/2022,57.42
```

```
8192,3/16/2022,53.80
8193,4/1/2022,82.06
8194,5/7/2022,40.39
8195,5/16/2022,87.21
8196,6/15/2022,95.93
8197,6/26/2022,45.89
8198,7/9/2022,36.23
8199,7/22/2022,25.66
8200,7/23/2022,82.77
8201,7/27/2022,69.98
8202,8/2/2022,76.11
8203,8/8/2022,25.12
8204,8/19/2022,46.23
8205,9/26/2022,84.21
8206,10/14/2022,96.24
8207,10/29/2022,67.67
];
```

Результаты

Загрузите данные и откройте лист. Создайте новую таблицу и добавьте эти поля как измерения:

- id
- date
- end_of_quarter
- end_of_quarter_timestamp

Результирующая таблица

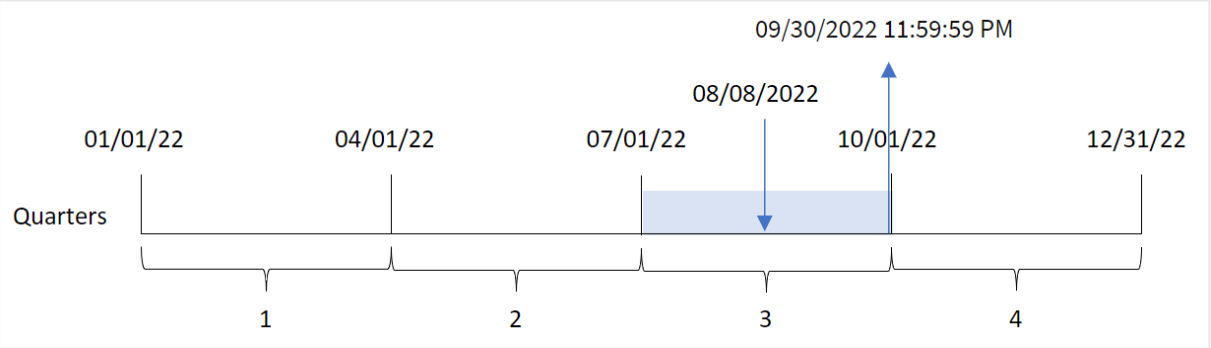
id	date	end_of_quarter	end_of_quarter_timestamp
8188	1/7/2022	03/31/2022	3/31/2022 11:59:59 PM
8189	1/19/2022	03/31/2022	3/31/2022 11:59:59 PM
8190	2/5/2022	03/31/2022	3/31/2022 11:59:59 PM
8191	2/28/2022	03/31/2022	3/31/2022 11:59:59 PM
8192	3/16/2022	03/31/2022	3/31/2022 11:59:59 PM
8193	4/1/2022	06/30/2022	6/30/2022 11:59:59 PM
8194	5/7/2022	06/30/2022	6/30/2022 11:59:59 PM
8195	5/16/2022	06/30/2022	6/30/2022 11:59:59 PM
8196	6/15/2022	06/30/2022	6/30/2022 11:59:59 PM
8197	6/26/2022	06/30/2022	6/30/2022 11:59:59 PM
8198	7/9/2022	09/30/2022	9/30/2022 11:59:59 PM
8199	7/22/2022	09/30/2022	9/30/2022 11:59:59 PM
8200	7/23/2022	09/30/2022	9/30/2022 11:59:59 PM

id	date	end_of_quarter	end_of_quarter_timestamp
8201	7/27/2022	09/30/2022	9/30/2022 11:59:59 PM
8202	8/2/2022	09/30/2022	9/30/2022 11:59:59 PM
8203	8/8/2022	09/30/2022	9/30/2022 11:59:59 PM
8204	8/19/2022	09/30/2022	9/30/2022 11:59:59 PM
8205	9/26/2022	09/30/2022	9/30/2022 11:59:59 PM
8206	10/14/2022	12/31/2022	12/31/2022 11:59:59 PM
8207	10/29/2022	12/31/2022	12/31/2022 11:59:59 PM

Поле `end_of_quarter` создано предшествующим оператором load с использованием функции `quarterend()`, где в качестве аргумента функции передано поле даты.

Функция `quarterend()` первоначально определяет, к какому кварталу относится значение даты, и затем возвращает метку времени для последней миллисекунды этого квартала.

Диаграмма функции `quarterend()`, определяющей конец квартала для транзакции 8203



Транзакция 8203 совершена 8 августа. Функция `quarterend()` определяет, что транзакция совершена в третьем квартале, и возвращает последнюю миллисекунду этого квартала, то есть 23:59:59 (11:59:59 PM) 30 сентября.

Пример 2. Аргумент period_no

Скрипт загрузки и результаты

Обзор

Откройте редактор загрузки данных и добавьте приведенный ниже скрипт загрузки на новую вкладку.

Скрипт загрузки содержит следующее:

- Набор данных, содержащий набор транзакций за 2022 год, загруженный в таблицу под именем `Transactions`.

- Предыдущая загрузка, которая содержит следующее:
    - Функция `quarterend()`, заданная как поле `previous_quarter_end`, которая возвращает метку времени конца квартала, предшествующего совершению транзакции.
    - Функция `timestamp()`, заданная как поле `previous_end_of_quarter_timestamp`, которая возвращает точную метку времени конца квартала, предшествующего совершению транзакции.

Скрипт загрузки

```
SET DateFormat='MM/DD/YYYY';

Transactions:
    Load
       *,
       quarterend(date, -1) as previous_quarter_end,
       timestamp(quarterend(date, -1)) as previous_quarter_end_timestamp
       ;
Load
*
Inline
[
id,date,amount
8188,1/7/2022,17.17
8189,1/19/2022,37.23
8190,2/28/2022,88.27
8191,2/5/2022,57.42
8192,3/16/2022,53.80
8193,4/1/2022,82.06
8194,5/7/2022,40.39
8195,5/16/2022,87.21
8196,6/15/2022,95.93
8197,6/26/2022,45.89
8198,7/9/2022,36.23
8199,7/22/2022,25.66
8200,7/23/2022,82.77
8201,7/27/2022,69.98
8202,8/2/2022,76.11
8203,8/8/2022,25.12
8204,8/19/2022,46.23
8205,9/26/2022,84.21
8206,10/14/2022,96.24
8207,10/29/2022,67.67
];
```

Результаты

Загрузите данные и откройте лист. Создайте новую таблицу и добавьте эти поля как измерения:

- `id`
- `date`

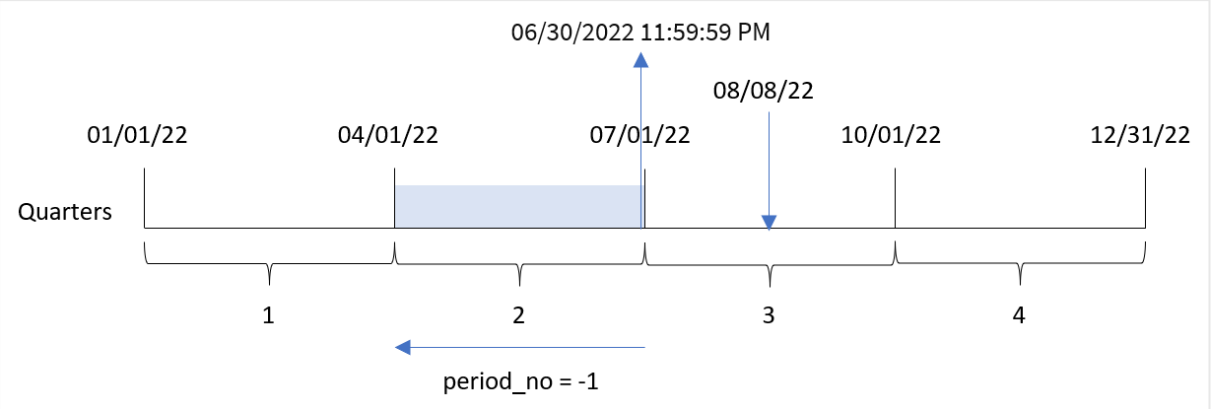
- previous_quarter_end
- previous_quarter_end_timestamp

Результирующая таблица

id	date	previous_quarter_end	previous_quarter_end_timestamp
8188	1/7/2022	12/31/2021	12/31/2021 11:59:59 PM
8189	1/19/2022	12/31/2021	12/31/2021 11:59:59 PM
8190	2/5/2022	12/31/2021	12/31/2021 11:59:59 PM
8191	2/28/2022	12/31/2021	12/31/2021 11:59:59 PM
8192	3/16/2022	12/31/2021	12/31/2021 11:59:59 PM
8193	4/1/2022	03/31/2022	3/31/2022 11:59:59 PM
8194	5/7/2022	03/31/2022	3/31/2022 11:59:59 PM
8195	5/16/2022	03/31/2022	3/31/2022 11:59:59 PM
8196	6/15/2022	03/31/2022	3/31/2022 11:59:59 PM
8197	6/26/2022	03/31/2022	3/31/2022 11:59:59 PM
8198	7/9/2022	06/30/2022	6/30/2022 11:59:59 PM
8199	7/22/2022	06/30/2022	6/30/2022 11:59:59 PM
8200	7/23/2022	06/30/2022	6/30/2022 11:59:59 PM
8201	7/27/2022	06/30/2022	6/30/2022 11:59:59 PM
8202	8/2/2022	06/30/2022	6/30/2022 11:59:59 PM
8203	8/8/2022	06/30/2022	6/30/2022 11:59:59 PM
8204	8/19/2022	06/30/2022	6/30/2022 11:59:59 PM
8205	9/26/2022	06/30/2022	6/30/2022 11:59:59 PM
8206	10/14/2022	09/30/2022	9/30/2022 11:59:59 PM
8207	10/29/2022	09/30/2022	9/30/2022 11:59:59 PM

Так как в качестве аргумента смещения используется `period_no = -1`, функция `quarterend()` сначала определяет квартал, в течение которого совершены транзакции. Затем она возвращается на квартал назад и определяет последнюю миллисекунду предыдущего квартала.

Диаграмма функции `quarterend()` с period_no = -1.



Транзакция 8203 совершена 8 августа. Функция `quarterend()` определяет, что квартал, предшествующий совершению транзакции, начинается 1 апреля и заканчивается 30 июня. Затем функция возвращает последнюю миллисекунду этого квартала, 23:59:59 (11:59:59 PM) 30 июля.

Пример 3. Аргумент first_month_of_year

Скрипт загрузки и результаты

Обзор

Откройте редактор загрузки данных и добавьте приведенный ниже скрипт загрузки на новую вкладку.

Скрипт загрузки содержит следующее:

- Набор данных, содержащий набор транзакций за 2022 год, загруженный в таблицу под именем `Transactions`.
- Предыдущая загрузка, которая содержит следующее:
  - Функция `quarterend()`, заданная как поле `end_of_quarter`, которая возвращает метку времени конца квартала, в течение которого совершены транзакции.
  - Функция `timestamp()`, заданная как поле `end_of_quarter_timestamp`, которая возвращает точную метку времени конца выбранного квартала.

Однако в этом примере согласно политике компании финансовый год начинается 1 марта.

Скрипт загрузки

```
SET DateFormat='MM/DD/YYYY';

Transactions:
    Load
       *,
       quarterend(date, 0, 3) as end_of_quarter,
       timestamp(quarterend(date, 0, 3)) as end_of_quarter_timestamp
       ;
Load
```

```
*
Inline
[
id,date,amount
8188,1/7/2022,17.17
8189,1/19/2022,37.23
8190,2/28/2022,88.27
8191,2/5/2022,57.42
8192,3/16/2022,53.80
8193,4/1/2022,82.06
8194,5/7/2022,40.39
8195,5/16/2022,87.21
8196,6/15/2022,95.93
8197,6/26/2022,45.89
8198,7/9/2022,36.23
8199,7/22/2022,25.66
8200,7/23/2022,82.77
8201,7/27/2022,69.98
8202,8/2/2022,76.11
8203,8/8/2022,25.12
8204,8/19/2022,46.23
8205,9/26/2022,84.21
8206,10/14/2022,96.24
8207,10/29/2022,67.67
];
```

Результаты

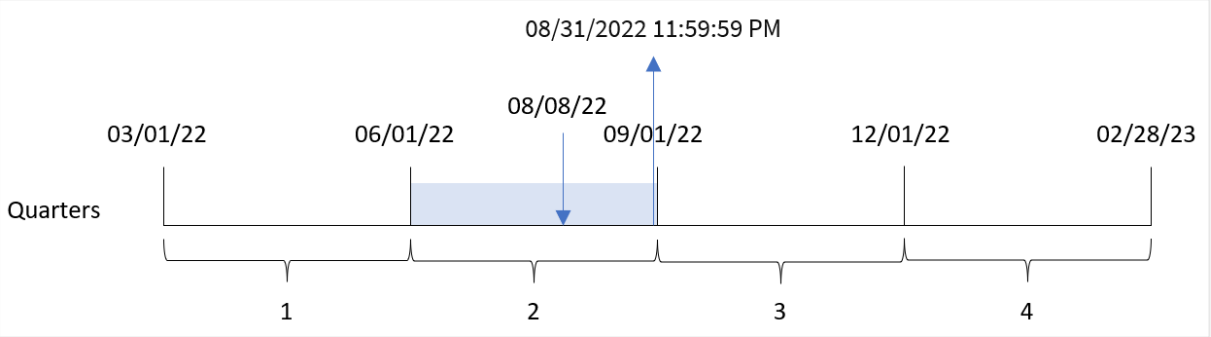
Результирующая таблица

id	date	end_of_quarter	end_of_quarter_timestamp
8188	1/7/2022	02/28/2022	2/28/2022 11:59:59 PM
8189	1/19/2022	02/28/2022	2/28/2022 11:59:59 PM
8190	2/5/2022	02/28/2022	2/28/2022 11:59:59 PM
8191	2/28/2022	02/28/2022	2/28/2022 11:59:59 PM
8192	3/16/2022	05/31/2022	5/31/2022 11:59:59 PM
8193	4/1/2022	05/31/2022	5/31/2022 11:59:59 PM
8194	5/7/2022	05/31/2022	5/31/2022 11:59:59 PM
8195	5/16/2022	05/31/2022	5/31/2022 11:59:59 PM
8196	6/15/2022	08/31/2022	8/31/2022 11:59:59 PM
8197	6/26/2022	08/31/2022	8/31/2022 11:59:59 PM
8198	7/9/2022	08/31/2022	8/31/2022 11:59:59 PM
8199	7/22/2022	08/31/2022	8/31/2022 11:59:59 PM
8200	7/23/2022	08/31/2022	8/31/2022 11:59:59 PM

id	date	end_of_quarter	end_of_quarter_timestamp
8201	7/27/2022	08/31/2022	8/31/2022 11:59:59 PM
8202	8/2/2022	08/31/2022	8/31/2022 11:59:59 PM
8203	8/8/2022	08/31/2022	8/31/2022 11:59:59 PM
8204	8/19/2022	08/31/2022	8/31/2022 11:59:59 PM
8205	9/26/2022	11/30/2022	11/30/2022 11:59:59 PM
8206	10/14/2022	11/30/2022	11/30/2022 11:59:59 PM
8207	10/29/2022	11/30/2022	11/30/2022 11:59:59 PM

Так как используется аргумент `first_month_of_year = 3` в функции `quarterend()`, начало года переносится с 1 января на 1 марта.

Диаграмма функции `quarterend()`, где в качестве первого месяца года задан март.



Транзакция 8203 совершена 8 августа. Так как год начинается 1 марта, он делится на кварталы март-май, июнь-август, сентябрь-ноябрь и декабрь-февраль.

Транзакция 8203 совершена 8 августа. Функция `quarterend()` определяет, что транзакция совершена в квартале с июня по август и возвращает последнюю миллисекунду этого квартала, то есть 23:59:59 (11:59:59 PM) 31 августа.

Пример 4. Пример объекта диаграммы

Скрипт загрузки и выражение диаграммы

Обзор

Используется тот же набор данных и сценарий, что в первом примере.

Однако в этом примере в приложение загружается неизмененный набор данных. Расчет, возвращающий метку времени окончания квартала, в котором совершены транзакции, создается как мера в диаграмме приложения.

Скрипт загрузки

```
SET DateFormat='MM/DD/YYYY';

Transactions:
Load
*
Inline
[
id,date,amount
8188,1/7/2022,17.17
8189,1/19/2022,37.23
8190,2/28/2022,88.27
8191,2/5/2022,57.42
8192,3/16/2022,53.80
8193,4/1/2022,82.06
8194,5/7/2022,40.39
8195,5/16/2022,87.21
8196,6/15/2022,95.93
8197,6/26/2022,45.89
8198,7/9/2022,36.23
8199,7/22/2022,25.66
8200,7/23/2022,82.77
8201,7/27/2022,69.98
8202,8/2/2022,76.11
8203,8/8/2022,25.12
8204,8/19/2022,46.23
8205,9/26/2022,84.21
8206,10/14/2022,96.24
8207,10/29/2022,67.67
];
```

Результаты

Загрузите данные и откройте лист. Создайте новую таблицу и добавьте эти поля как измерения:

- `id`
- `date`

Чтобы рассчитать дату окончания квартала, в течение которого совершена транзакция, создайте следующие меры:

- `=quarterend(date)`
- `=timestamp(quarterend(date))`

Результирующая таблица

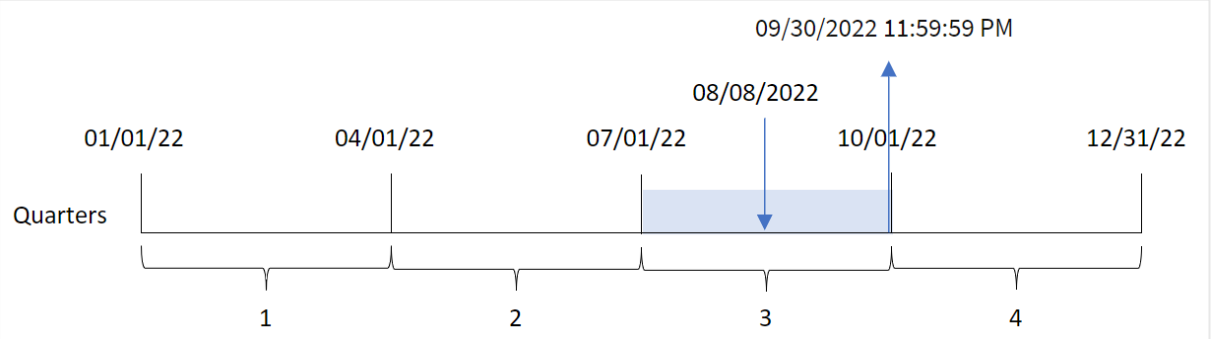
id	date	=quarterend(date)	=timestamp(quarterend(date))
8188	1/7/2022	03/31/2022	3/31/2022 11:59:59 PM
8189	1/19/2022	03/31/2022	3/31/2022 11:59:59 PM

id	date	=quarterend(date)	=timestamp(quarterend(date))
8190	2/5/2022	03/31/2022	3/31/2022 11:59:59 PM
8191	2/28/2022	03/31/2022	3/31/2022 11:59:59 PM
8192	3/16/2022	03/31/2022	3/31/2022 11:59:59 PM
8193	4/1/2022	06/30/2022	6/30/2022 11:59:59 PM
8194	5/7/2022	06/30/2022	6/30/2022 11:59:59 PM
8195	5/16/2022	06/30/2022	6/30/2022 11:59:59 PM
8196	6/15/2022	06/30/2022	6/30/2022 11:59:59 PM
8197	6/26/2022	06/30/2022	6/30/2022 11:59:59 PM
8198	7/9/2022	09/30/2022	9/30/2022 11:59:59 PM
8199	7/22/2022	09/30/2022	9/30/2022 11:59:59 PM
8200	7/23/2022	09/30/2022	9/30/2022 11:59:59 PM
8201	7/27/2022	09/30/2022	9/30/2022 11:59:59 PM
8202	8/2/2022	09/30/2022	9/30/2022 11:59:59 PM
8203	8/8/2022	09/30/2022	9/30/2022 11:59:59 PM
8204	8/19/2022	09/30/2022	9/30/2022 11:59:59 PM
8205	9/26/2022	09/30/2022	9/30/2022 11:59:59 PM
8206	10/14/2022	12/31/2022	12/31/2022 11:59:59 PM
8207	10/29/2022	12/31/2022	12/31/2022 11:59:59 PM

Поле `end_of_quarter` создано предшествующим оператором load с использованием функции `quarterend()`, где в качестве аргумента функции передано поле даты.

Функция `quarterend()` первоначально определяет, к какому кварталу относится значение даты, и затем возвращает метку времени для последней миллисекунды этого квартала.

Диаграмма функции `quarterend()`, определяющей конец квартала для транзакции 8203



Транзакция 8203 совершена 8 августа. Функция `quarterend()` определяет, что транзакция совершена в третьем квартале, и возвращает последнюю миллисекунду этого квартала, то есть 23:59:59 (11:59:59 PM) 30 сентября.

Пример 5. Сценарий

Скрипт загрузки и результаты

Обзор

Откройте редактор загрузки данных и добавьте приведенный ниже скрипт загрузки на новую вкладку.

Скрипт загрузки содержит следующее:

- Набор данных загружается в таблицу под именем `Employee_Expenses`. Данная таблица содержит следующие поля:
    - Employee IDs (Идентификаторы сотрудников)
    - Employee names (Имена сотрудников)
    - Средние ежедневные заявки на возмещение расходов каждого сотрудника.

Конечному пользователю требуется получить объект диаграммы, отображающий по идентификатору и имени сотрудника расчетные расходы, которые еще предстоит понести в течение оставшейся части квартала. Финансовый год начинается в январе.

Скрипт загрузки

```
Employee_Expenses:
Load
*
Inline
[
employee_id,employee_name,avg_daily_claim
182,Mark, $15
183,Deryck, $12.5
184,Dexter, $12.5
185,Sydney,$27
186,Agatha,$18
];
```

Результаты

Загрузите данные и откройте лист. Создайте новую таблицу и добавьте эти поля как измерения:

- `employee_id`
- `employee_name`

Чтобы рассчитать накопленный процент, создайте следующую меру:

- `=(quarterend(today(1))-today(1))*avg_daily_claim`

Задайте параметру **Формат чисел** меры значение **Денежный**.

Результирующая таблица

employee_id	employee_name	=(quarterend(today(1))-today(1))*avg_daily_claim
182	Mark	$480.00
183	Deryck	$400.00
184	Dexter	$400.00
185	Sydney	$864.00
186	Agatha	$576.00

Функция `quarterend()` возвращает дату окончания текущего месяца, используя сегодняшнюю дату в качестве единственного аргумента. Затем она вычитает сегодняшнюю дату из даты окончания, и выражение возвращает количество дней, оставшихся в этом месяце.

Затем это значение умножается на среднюю ежедневную заявку на возмещение расходов каждого сотрудника для расчета оценочной суммы заявок, которые каждый сотрудник должен подать до конца квартала.

quartername

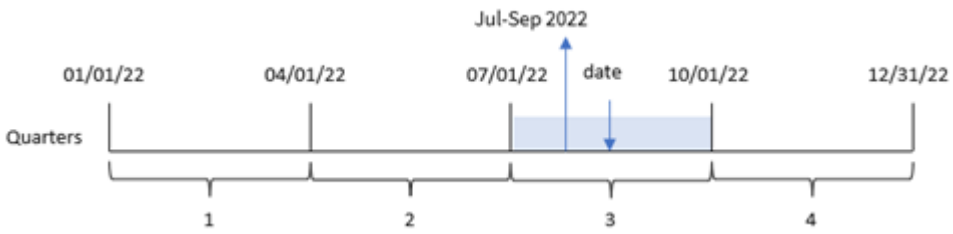
Эта функция возвращает значение, отображающее месяцы квартала (в формате переменной **MonthNames** скрипта) и год с базовым числовым значением, соответствующим метке времени, включающей первую миллисекунду первого дня квартала.

Синтаксис:

```
QuarterName(date[, period_no[, first_month_of_year]])
```

Возвращаемые типы данных: двойное значение

Схема функции `quartername()`



Функция `quartername()` определяет, на какой квартал приходится дата. Затем она возвращает начальный и конечный месяцы этого квартала, а также указывает год. Базовым числовым значением в этом результате является первая миллисекунда квартала.

Аргументы

Аргумент	Описание
date	Дата или метка времени 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**.

Когда это следует использовать

Функция `quartername()` полезна, когда требуется сравнить агрегации по кварталам, например, если требуется увидеть общий объем продаж продуктов по кварталам.

С помощью этой функции в скрипте загрузки можно создать поле в таблице основного календаря. Ее также можно использовать непосредственно в диаграмме в качестве вычисляемого измерения.

В этих примерах используется формат даты MM/DD/YYYY. Формат даты указан в операторе `SET DateFormat` в верхней части скрипта загрузки данных. Измените формат в примерах согласно своим пожеланиям.

Примеры функции

Пример	Результат
`quartername('10/29/2013')`	Возвращает Oct-Dec 2013.
`quartername('10/29/2013', -1)`	Возвращает Jul-Sep 2013.
`quartername('10/29/2013', 0, 3)`	Возвращает Sep-Nov 2013.

Региональные настройки

Если не указано иное, в примерах, приведенных в данном разделе, используется следующий формат даты: ММ/ДД/ГГГГ. Формат даты указан в операторе `SET DateFormat` скрипта загрузки данных. В вашей системе может быть установлен другой формат даты по умолчанию в зависимости от региональных настроек и других факторов. Можно изменить формат в примерах в соответствии с потребностями. Или можно изменить форматы в скрипте загрузки в соответствии с этими примерами.

Региональные настройки по умолчанию в приложениях основаны на системных региональных настройках компьютера или сервера, где установлено ПО Qlik Sense. Если на сервере Qlik Sense, к которому обращается пользователь, выбран шведский язык, то редактор загрузки данных будет использовать шведские региональные настройки для даты, времени и валюты. Эти параметры регионального формата не связаны с языком, отображаемым в интерфейсе пользователя Qlik Sense. Qlik Sense будет отображаться на языке, который используется в браузере.

Пример 1. Дата без дополнительных аргументов

Скрипт загрузки и результаты

Обзор

Откройте редактор загрузки данных и добавьте приведенный ниже скрипт загрузки на новую вкладку.

Скрипт загрузки содержит следующее:

- Набор данных, содержащий набор транзакций за 2022 год, загружается в таблицу под именем `Transactions`.
- Поле даты было предоставлено в формате системной переменной `DateFormat` (MM/DD/YYYY).
- Создание поля `transaction_quarter`, возвращающего квартал, в котором совершены транзакции.

Добавьте сюда другой текст, если необходимо, со списками и т. д.

Скрипт загрузки

```
SET DateFormat='MM/DD/YYYY';
SET MonthNames='Jan;Feb;Mar;Apr;May;Jun;Jul;Aug;Sep;Oct;Nov;Dec';

Transactions:
    Load
        *,
        quartername(date) as transaction_quarter
        ;
Load
*
Inline
[
id,date,amount
8188,1/7/2022,17.17
8189,1/19/2022,37.23
8190,2/28/2022,88.27
8191,2/5/2022,57.42
8192,3/16/2022,53.80
8193,4/1/2022,82.06
8194,5/7/2022,40.39
8195,5/16/2022,87.21
8196,6/15/2022,95.93
8197,6/26/2022,45.89
8198,7/9/2022,36.23
8199,7/22/2022,25.66
8200,7/23/2022,82.77
8201,7/27/2022,69.98
8202,8/2/2022,76.11
8203,8/8/2022,25.12
8204,8/19/2022,46.23
```

```
8205,9/26/2022,84.21
8206,10/14/2022,96.24
8207,10/29/2022,67.67
];
```

Результаты

Загрузите данные и откройте лист. Создайте новую таблицу и добавьте эти поля как измерения:

- `date`
- `transaction_quarter`

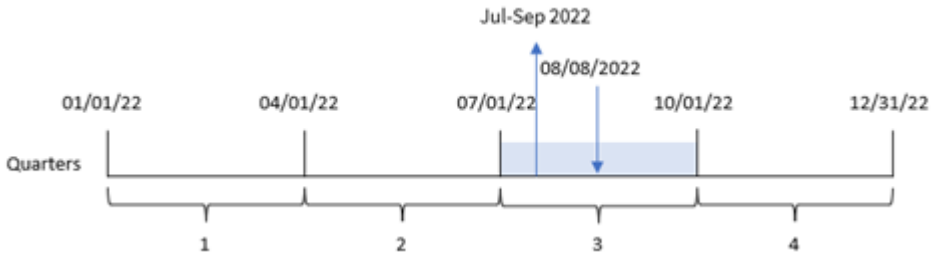
Результирующая таблица

date	transaction_quarter
1/7/2022	Jan-Mar 2022
1/19/2022	Jan-Mar 2022
2/5/2022	Jan-Mar 2022
2/28/2022	Jan-Mar 2022
3/16/2022	Jan-Mar 2022
4/1/2022	Apr-Jun 2022
5/7/2022	Apr-Jun 2022
5/16/2022	Apr-Jun 2022
6/15/2022	Apr-Jun 2022
6/26/2022	Apr-Jun 2022
7/9/2022	Jul-Sep 2022
7/22/2022	Jul-Sep 2022
7/23/2022	Jul-Sep 2022
7/27/2022	Jul-Sep 2022
8/2/2022	Jul-Sep 2022
8/8/2022	Jul-Sep 2022
8/19/2022	Jul-Sep 2022
9/26/2022	Jul-Sep 2022
10/14/2022	Oct-Dec 2022
10/29/2022	Oct-Dec 2022

Поле `transaction_quarter` создано предшествующим оператором load с использованием функции `quartername()`, в качестве аргумента которой передано поле даты.

Функция `quartername()` сначала определяет квартал, к которому относится значение даты. Затем она возвращает начальный и конечный месяцы этого квартала, а также указывает год.

Диаграмма функции `quartername()`, пример без дополнительных аргументов



Транзакция 8203 совершена 8 августа 2022 года. Функция `quartername()` определяет, что транзакция совершена в третьем квартале, и поэтому возвращает значение Jul-Sep 2022 (июль-сентябрь 2022 года). Месяцы отображаются в том же формате, который задан для системной переменной `MonthNames`.

Пример 2. Дата с аргументом period_no

Скрипт загрузки и результаты

Обзор

Откройте редактор загрузки данных и добавьте приведенный ниже скрипт загрузки на новую вкладку.

Скрипт загрузки содержит следующее:

- Используется тот же набор данных и сценарий, что в первом примере.
- Создание поля `previous_quarter`, возвращающего квартал, который предшествует кварталу, когда совершены транзакции.

Скрипт загрузки

```
SET DateFormat='MM/DD/YYYY';
SET MonthNames='Jan;Feb;Mar;Apr;May;Jun;Jul;Aug;Sep;Oct;Nov;Dec';

Transactions:
    Load
        *,
        quartername(date,-1) as previous_quarter
        ;
Load
*
Inline
[
id,date,amount
8188,1/7/2022,17.17
8189,1/19/2022,37.23
```

```
8190,2/28/2022,88.27
8191,2/5/2022,57.42
8192,3/16/2022,53.80
8193,4/1/2022,82.06
8194,5/7/2022,40.39
8195,5/16/2022,87.21
8196,6/15/2022,95.93
8197,6/26/2022,45.89
8198,7/9/2022,36.23
8199,7/22/2022,25.66
8200,7/23/2022,82.77
8201,7/27/2022,69.98
8202,8/2/2022,76.11
8203,8/8/2022,25.12
8204,8/19/2022,46.23
8205,9/26/2022,84.21
8206,10/14/2022,96.24
8207,10/29/2022,67.67
];
```

Результаты

Загрузите данные и откройте лист. Создайте новую таблицу и добавьте эти поля как измерения:

- `date`
- `previous_quarter`

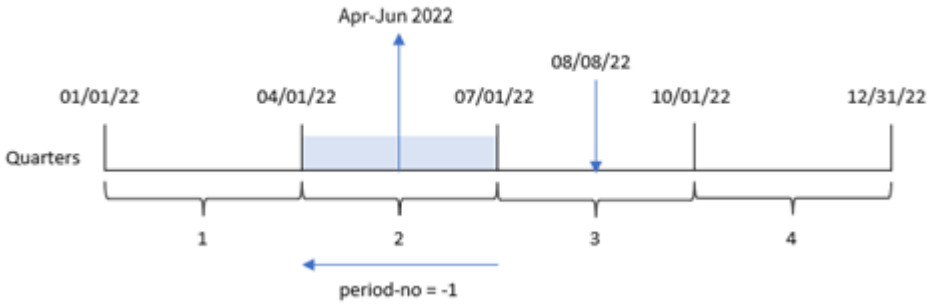
Результирующая таблица

date	previous_quarter
1/7/2022	Oct-Dec 2021
1/19/2022	Oct-Dec 2021
2/5/2022	Oct-Dec 2021
2/28/2022	Oct-Dec 2021
3/16/2022	Oct-Dec 2021
4/1/2022	Jan-Mar 2022
5/7/2022	Jan-Mar 2022
5/16/2022	Jan-Mar 2022
6/15/2022	Jan-Mar 2022
6/26/2022	Jan-Mar 2022
7/9/2022	Apr-Jun 2022
7/22/2022	Apr-Jun 2022
7/23/2022	Apr-Jun 2022

date	previous_quarter
7/27/2022	Apr-Jun 2022
8/2/2022	Apr-Jun 2022
8/8/2022	Apr-Jun 2022
8/19/2022	Apr-Jun 2022
9/26/2022	Apr-Jun 2022
10/14/2022	Jul-Sep 2022
10/29/2022	Jul-Sep 2022

В этом случае, так как в качестве аргумента смещения использовалось `period_no` = -1, функция `quartername()` сначала определяет, что транзакции совершены в третьем квартале. Затем она переходит к предыдущему кварталу и возвращает значение, отображающее начальный и конечный месяцы этого квартала, а также год.

Диаграмма функции `quartername()`, пример с аргументом period_no



Транзакция 8203 совершена 8 августа. Функция `quartername()` определяет, что квартал, предшествующий совершению транзакции, начинается 1 апреля и заканчивается 30 июня. Таким образом, функция возвращает значение Apr-Jun 2022 (апрель-июнь 2022).

Пример 3. Дата с аргументом first_week_day

Скрипт загрузки и результаты

Обзор

Откройте редактор загрузки данных и добавьте приведенный ниже скрипт загрузки на новую вкладку.

Скрипт загрузки содержит тот же набор данных и сценарий, что в первом примере. Однако в этом примере нам также нужно задать 1 марта в качестве начала финансового года.

Скрипт загрузки

```
SET DateFormat='MM/DD/YYYY';
SET MonthNames='Jan;Feb;Mar;Apr;May;Jun;Jul;Aug;Sep;Oct;Nov;Dec';

Transactions:
    Load
        *,
        quartername(date,0,3) as transaction_quarter
        ;
Load
*
Inline
[
id,date,amount
8188,1/7/2022,17.17
8189,1/19/2022,37.23
8190,2/28/2022,88.27
8191,2/5/2022,57.42
8192,3/16/2022,53.80
8193,4/1/2022,82.06
8194,5/7/2022,40.39
8195,5/16/2022,87.21
8196,6/15/2022,95.93
8197,6/26/2022,45.89
8198,7/9/2022,36.23
8199,7/22/2022,25.66
8200,7/23/2022,82.77
8201,7/27/2022,69.98
8202,8/2/2022,76.11
8203,8/8/2022,25.12
8204,8/19/2022,46.23
8205,9/26/2022,84.21
8206,10/14/2022,96.24
8207,10/29/2022,67.67
];
```

Результаты

Загрузите данные и откройте лист. Создайте новую таблицу и добавьте эти поля как измерения:

- `date`
- `transaction_quarter`

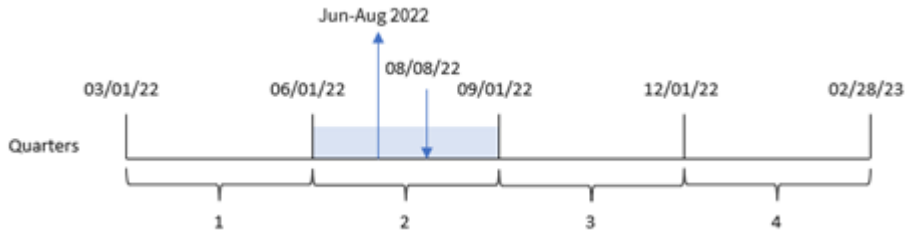
Результирующая таблица

date	transaction_quarter
1/7/2022	Dec-Feb 2021
1/19/2022	Dec-Feb 2021
2/5/2022	Dec-Feb 2021

date	transaction_quarter
2/28/2022	Dec-Feb 2021
3/16/2022	Mar-May 2022
4/1/2022	Mar-May 2022
5/7/2022	Mar-May 2022
5/16/2022	Mar-May 2022
6/15/2022	Jun-Aug 2022
6/26/2022	Jun-Aug 2022
7/9/2022	Jun-Aug 2022
7/22/2022	Jun-Aug 2022
7/23/2022	Jun-Aug 2022
7/27/2022	Jun-Aug 2022
8/2/2022	Jun-Aug 2022
8/8/2022	Jun-Aug 2022
8/19/2022	Jun-Aug 2022
9/26/2022	Sep-Nov 2022
10/14/2022	Sep-Nov 2022
10/29/2022	Sep-Nov 2022

В данном случае, так как используется аргумент `first_month_of_year` = 3 в функции `quartername()`, начало года переносится с 1 января на 1 марта. Таким образом, год делится на следующие кварталы: март-май, июнь-август, сентябрь-ноябрь и декабрь-февраль.

Диаграмма функции `quartername()` , пример с аргументом first_week_day



Транзакция 8203 совершена 8 августа. Функция `quartername()` определяет, что транзакция совершена во втором квартале, который начинается 1 июня и заканчивается 31 августа. Таким образом, функция возвращает значение Jun-Aug 2022 (июнь-август 2022).

Пример 4. Пример объекта диаграммы

Скрипт загрузки и выражение диаграммы

Обзор

Откройте редактор загрузки данных и добавьте приведенный ниже скрипт загрузки на новую вкладку.

Скрипт загрузки содержит тот же набор данных и сценарий, что в первом примере.

Однако в этом примере в приложение загружается неизмененный набор данных. Расчет, возвращающий метку времени окончания квартала, в котором совершены транзакции, создается как мера в объекте диаграммы в приложении.

Скрипт загрузки

```
Transactions:
Load
*
Inline
[
id,date,amount
8188,1/7/2022,17.17
8189,1/19/2022,37.23
8190,2/28/2022,88.27
8191,2/5/2022,57.42
8192,3/16/2022,53.80
8193,4/1/2022,82.06
8194,5/7/2022,40.39
8195,5/16/2022,87.21
8196,6/15/2022,95.93
8197,6/26/2022,45.89
8198,7/9/2022,36.23
8199,7/22/2022,25.66
8200,7/23/2022,82.77
8201,7/27/2022,69.98
8202,8/2/2022,76.11
8203,8/8/2022,25.12
8204,8/19/2022,46.23
8205,9/26/022,84.21
8206,10/14/2022,96.24
8207,10/29/2022,67.67
];
```

Результаты

Загрузите данные и откройте лист. Создайте новую таблицу и добавьте это поле как измерение: `date`.

Создайте следующую меру:

```
=quartername(date)
```

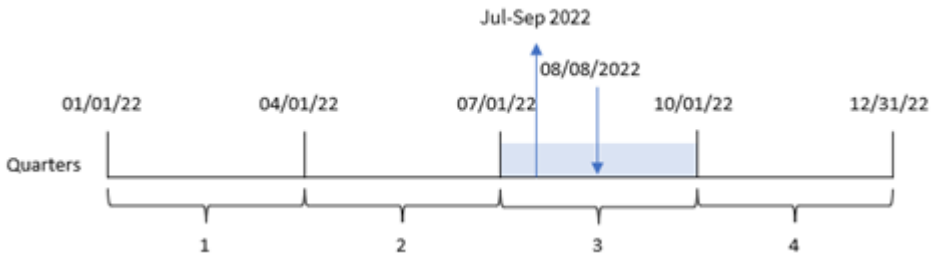
Результирующая таблица

date	=quartername(date)
1/7/2022	Jan-Mar 2022
1/19/2022	Jan-Mar 2022
2/5/2022	Jan-Mar 2022
2/28/2022	Jan-Mar 2022
3/16/2022	Jan-Mar 2022
4/1/2022	Apr-Jun 2022
5/7/2022	Apr-Jun 2022
5/16/2022	Apr-Jun 2022
6/15/2022	Apr-Jun 2022
6/26/2022	Apr-Jun 2022
7/9/2022	Jul-Sep 2022
7/22/2022	Jul-Sep 2022
7/23/2022	Jul-Sep 2022
7/27/2022	Jul-Sep 2022
8/2/2022	Jul-Sep 2022
8/8/2022	Jul-Sep 2022
8/19/2022	Jul-Sep 2022
9/26/2022	Jul-Sep 2022
10/14/2022	Oct-Dec 2022
10/29/2022	Oct-Dec 2022

Мера `transaction_quarter` создана в объекте диаграммы с использованием функции `quartername()`, где в качестве аргумента функции передано поле `date`.

Функция `quartername()` сначала определяет квартал, к которому относится значение даты. Затем она возвращает начальный и конечный месяцы этого квартала, а также указывает год.

Диаграмма функции quartername(), пример с объектом диаграммы



Транзакция 8203 совершена 8 августа 2022 года. Функция `quartername()` определяет, что транзакция совершена в третьем квартале, и поэтому возвращает значение Jul-Sep 2022 (июль-сентябрь 2022 года). Месяцы отображаются в том же формате, который задан для системной переменной `MonthNames`.

Пример 5. Сценарий

Скрипт загрузки и выражение диаграммы

Обзор

Откройте редактор загрузки данных и добавьте приведенный ниже скрипт загрузки на новую вкладку.

Скрипт загрузки содержит следующее:

- Набор данных, содержащий набор транзакций за 2022 год, загружается в таблицу под именем `Transactions`.
- Поле даты было предоставлено в формате системной переменной `DateFormat` (MM/DD/YYYY).

Конечному пользователю нужен объект диаграммы, на котором представлены общие продажи для транзакций по кварталам. Этого можно добиться, даже если это измерение недоступно в модели данных, используя в диаграмме функцию `quartername()` в качестве вычисляемого измерения.

Скрипт загрузки

```
SET DateFormat='MM/DD/YYYY';

Transactions:
Load
*
Inline
[
id,date,amount
8188,'1/7/2022',17.17
8189,'1/19/2022',37.23
8190,'2/28/2022',88.27
8191,'2/5/2022',57.42
8192,'3/16/2022',53.80
```

```
8193,'4/1/2022',82.06
8194,'5/7/2022',40.39
8195,'5/16/2022',87.21
8196,'6/15/2022',95.93
8197,'6/26/2022',45.89
8198,'7/9/2022',36.23
8199,'7/22/2022',25.66
8200,'7/23/2022',82.77
8201,'7/27/2022',69.98
8202,'8/2/2022',76.11
8203,'8/8/2022',25.12
8204,'8/19/2022',46.23
8205,'9/26/2022',84.21
8206,'10/14/2022',96.24
8207,'10/29/2022',67.67
];
```

Результаты

Выполните следующие действия.

1. Загрузите данные и откройте лист. Создайте новую таблицу.
2. Создайте вычисляемое измерение, используя следующее выражение:
   `=quartername(date)`
3. Затем вычислите общий объем продаж, используя следующую меру агрегирования:
   `=sum(amount)`
4. Задайте параметру меры **Формат чисел** значение **Денежный**.

   Результирующая таблица

=quartername(date)	=sum(amount)
Jul-Sep 2022	$446.31
Apr-Jun 2022	$351.48
Jan-Mar 2022	$253.89
Oct-Dec 2022	$163.91

quarterstart

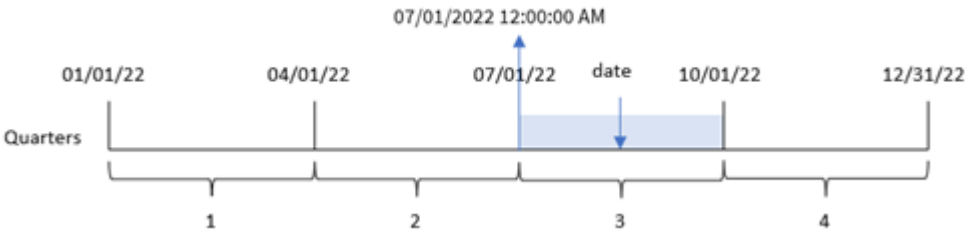
Эта функция возвращает значение, соответствующее метке времени, включающей первую миллисекунду квартала, содержащего значение, указанное в поле **date**. По умолчанию для вывода используется формат **DateFormat**, установленный в скрипте.

Синтаксис:

```
QuarterStart(date[, period_no[, first_month_of_year]])
```

Возвращаемые типы данных: двойное значение

Схема функции `quarterstart()`



Функция `quarterstart()` определяет, на какой квартал приходится `date`. Затем она возвращает метку времени в формате даты для первой миллисекунды первого месяца этого квартала.

Аргументы

Аргумент	Описание
date	Дата или метка времени 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**.

Когда это следует использовать

Функция `quarterstart()` широко используется в составе выражения, когда пользователю требуется учитывать в расчетах часть квартала, которая уже прошла. Например, можно ее использовать, если требуется рассчитать проценты, накопленные в течение квартала до текущей даты.

Примеры функции

Пример	Результат
`quarterstart('10/29/2005')`	Возвращает 10/01/2005.
`quarterstart('10/29/2005', -1 )`	Возвращает 07/01/2005.
`quarterstart('10/29/2005', 0, 3)`	Возвращает 09/01/2005.

Региональные настройки

Если не указано иное, в примерах, приведенных в данном разделе, используется следующий формат даты: ММ/ДД/ГГГГ. Формат даты указан в операторе `SET DateFormat` скрипта загрузки данных. В вашей системе может быть установлен другой формат даты по умолчанию в

зависимости от региональных настроек и других факторов. Можно изменить формат в примерах в соответствии с потребностями. Или можно изменить форматы в скрипте загрузки в соответствии с этими примерами.

Региональные настройки по умолчанию в приложениях основаны на системных региональных настройках компьютера или сервера, где установлено ПО Qlik Sense. Если на сервере Qlik Sense, к которому обращается пользователь, выбран шведский язык, то редактор загрузки данных будет использовать шведские региональные настройки для даты, времени и валюты. Эти параметры регионального формата не связаны с языком, отображаемым в интерфейсе пользователя Qlik Sense. Qlik Sense будет отображаться на языке, который используется в браузере.

Пример 1. Без дополнительных аргументов

Скрипт загрузки и результаты

Обзор

Откройте редактор загрузки данных и добавьте приведенный ниже скрипт загрузки на новую вкладку.

Скрипт загрузки содержит следующее:

- Набор данных, содержащий набор транзакций за 2022 год, загружается в таблицу под именем `Transactions`.
- Поле даты было предоставлено в формате системной переменной `DateFormat` (MM/DD/YYYY).
- Создание поля `start_of_quarter`, возвращающего метку времени начала квартала, в течение которого совершены транзакции.

Скрипт загрузки

```
SET DateFormat='MM/DD/YYYY';

Transactions:
    Load
        *,
        quarterstart(date) as start_of_quarter,
        timestamp(quarterstart(date)) as start_of_quarter_timestamp
        ;
Load
*
Inline
[
id,date,amount
8188,1/7/2022,17.17
8189,1/19/2022,37.23
8190,2/28/2022,88.27
8191,2/5/2022,57.42
8192,3/16/2022,53.80
8193,4/1/2022,82.06
8194,5/7/2022,40.39
```

```
8195,5/16/2022,87.21
8196,6/15/2022,95.93
8197,6/26/2022,45.89
8198,7/9/2022,36.23
8199,7/22/2022,25.66
8200,7/23/2022,82.77
8201,7/27/2022,69.98
8202,8/2/2022,76.11
8203,8/8/2022,25.12
8204,8/19/2022,46.23
8205,9/26/2022,84.21
8206,10/14/2022,96.24
8207,10/29/2022,67.67
];
```

Результаты

Загрузите данные и откройте лист. Создайте новую таблицу и добавьте эти поля как измерения:

- `date`
- `start_of_quarter`
- `start_of_quarter_timestamp`

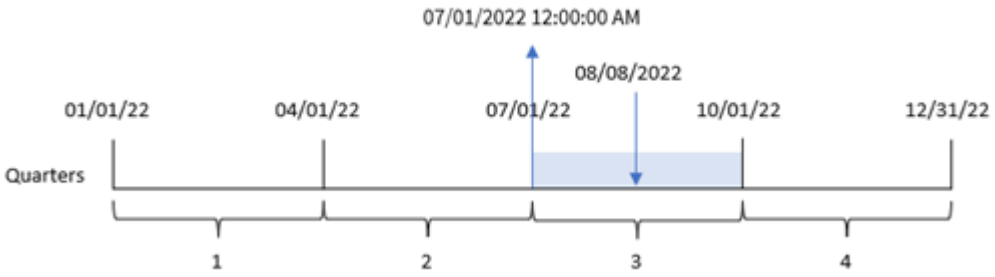
Результирующая таблица

date	start_of_quarter	start_of_quarter_timestamp
1/7/2022	01/01/2022	1/1/2022 12:00:00 AM
1/19/2022	01/01/2022	1/1/2022 12:00:00 AM
2/5/2022	01/01/2022	1/1/2022 12:00:00 AM
2/28/2022	01/01/2022	1/1/2022 12:00:00 AM
3/16/2022	01/01/2022	1/1/2022 12:00:00 AM
4/1/2022	04/01/2022	4/1/2021 12:00:00 AM
5/7/2022	04/01/2022	4/1/2021 12:00:00 AM
5/16/2022	04/01/2022	4/1/2021 12:00:00 AM
6/15/2022	04/01/2022	4/1/2021 12:00:00 AM
6/26/2022	04/01/2022	4/1/2021 12:00:00 AM
7/9/2022	07/01/2022	7/1/2021 12:00:00 AM
7/22/2022	07/01/2022	7/1/2021 12:00:00 AM
7/23/2022	07/01/2022	7/1/2021 12:00:00 AM
7/27/2022	07/01/2022	7/1/2021 12:00:00 AM
8/2/2022	07/01/2022	7/1/2021 12:00:00 AM

date	start_of_quarter	start_of_quarter_timestamp
8/8/2022	07/01/2022	7/1/2021 12:00:00 AM
8/19/2022	07/01/2022	7/1/2021 12:00:00 AM
9/26/2022	07/01/2022	7/1/2021 12:00:00 AM
10/14/2022	10/01/2022	10/1/2022 12:00:00 AM
10/29/2022	10/01/2022	10/1/2022 12:00:00 AM

Поле `start_of_quarter` создано предшествующим оператором load с использованием функции `quarterstart()`, в качестве аргумента которой передано поле даты. Функция `uarterstart()` сначала определяет квартал, к которому относится значение даты. Затем она возвращает метку времени для первой миллисекунды этого квартала.

Диаграмма функции `quarterstart()`, пример без дополнительных аргументов



Транзакция 8203 совершена 8 августа. Функция `quarterstart()` определяет, что транзакция совершена в третьем квартале, и возвращает первую миллисекунду этого квартала, то есть 00:00:00 (12:00:00 AM) 1 июля.

Пример 2. Скрипт period_no

Скрипт загрузки и результаты

Обзор

Откройте редактор загрузки данных и добавьте приведенный ниже скрипт загрузки на новую вкладку.

Скрипт загрузки содержит следующее:

- Используется тот же набор данных и сценарий, что в первом примере.
- Создание поля `previous_quarter_start`, возвращающего метку времени начала квартала, который предшествует совершению транзакции.

Скрипт загрузки

```
SET DateFormat='MM/DD/YYYY';
```

```
Transactions:
    Load
        *,
        quarterstart(date,-1) as previous_quarter_start,
        timestamp(quarterstart(date,-1)) as previous_quarter_start_timestamp
        ;
Load
*
Inline
[
id,date,amount
8188,1/7/2022,17.17
8189,1/19/2022,37.23
8190,2/28/2022,88.27
8191,2/5/2022,57.42
8192,3/16/2022,53.80
8193,4/1/2022,82.06
8194,5/7/2022,40.39
8195,5/16/2022,87.21
8196,6/15/2022,95.93
8197,6/26/2022,45.89
8198,7/9/2022,36.23
8199,7/22/2022,25.66
8200,7/23/2022,82.77
8201,7/27/2022,69.98
8202,8/2/2022,76.11
8203,8/8/2022,25.12
8204,8/19/2022,46.23
8205,9/26/2022,84.21
8206,10/14/2022,96.24
8207,10/29/2022,67.67
];
```

Результаты

Загрузите данные и откройте лист. Создайте новую таблицу и добавьте эти поля как измерения:

- `date`
- `previous_quarter_start`
- `previous_quarter_start_timestamp`

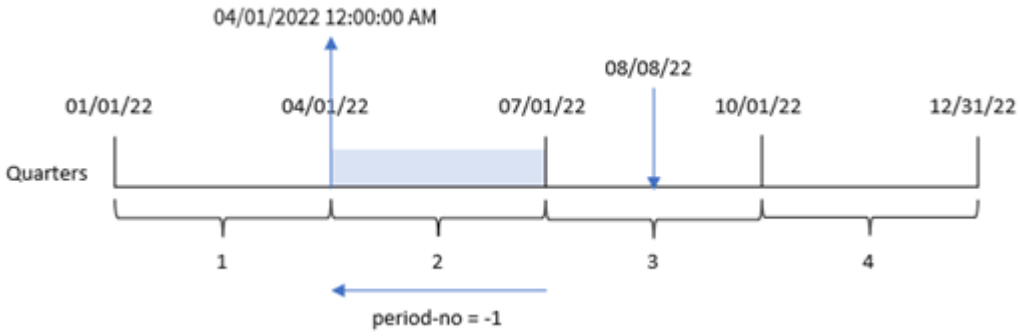
Результирующая таблица

date	previous_quarter_start	previous_quarter_start_timestamp
1/7/2022	10/01/2021	10/1/2021 12:00:00 AM
1/19/2022	10/01/2021	10/1/2021 12:00:00 AM
2/5/2022	10/01/2021	10/1/2021 12:00:00 AM
2/28/2022	10/01/2021	10/1/2021 12:00:00 AM
3/16/2022	10/01/2021	10/1/2021 12:00:00 AM

date	previous_quarter_start	previous_quarter_start_timestamp
4/1/2022	01/01/2022	1/1/2022 12:00:00 AM
5/7/2022	01/01/2022	1/1/2022 12:00:00 AM
5/16/2022	01/01/2022	1/1/2022 12:00:00 AM
6/15/2022	01/01/2022	1/1/2022 12:00:00 AM
6/26/2022	01/01/2022	1/1/2022 12:00:00 AM
7/9/2022	04/01/2022	4/1/2021 12:00:00 AM
7/22/2022	04/01/2022	4/1/2021 12:00:00 AM
7/23/2022	04/01/2022	4/1/2021 12:00:00 AM
7/27/2022	04/01/2022	4/1/2021 12:00:00 AM
8/2/2022	04/01/2022	4/1/2021 12:00:00 AM
8/8/2022	04/01/2022	4/1/2021 12:00:00 AM
8/19/2022	04/01/2022	4/1/2021 12:00:00 AM
9/26/2022	04/01/2022	4/1/2021 12:00:00 AM
10/14/2022	07/01/2022	7/1/2022 12:00:00 AM
10/29/2022	07/01/2022	7/1/2022 12:00:00 AM

В этом случае, так как в качестве аргумента смещения использовалось `period_no` = -1, функция `quarterstart()` сначала определяет квартал, в котором совершаются транзакции. Затем она возвращается на квартал назад и определяет первую миллисекунду предыдущего квартала.

Диаграмма функции `quarterstart()`, пример с аргументом period_no



Транзакция 8203 совершена 8 августа. Функция `quarterstart()` определяет, что квартал, предшествующий совершению транзакции, начинается 1 апреля и заканчивается 30 июня. Затем она возвращает первую миллисекунду этого квартала, 00:00:00 (12:00:00 AM) 1 апреля.

Пример 3. Аргумент first_month_of_year

Скрипт загрузки и результаты

Обзор

Откройте редактор загрузки данных и добавьте приведенный ниже скрипт загрузки на новую вкладку.

Скрипт загрузки содержит тот же набор данных и сценарий, что в первом примере. Однако в этом примере нам также нужно задать 1 марта в качестве начала финансового года.

Скрипт загрузки

```
SET DateFormat='MM/DD/YYYY';

Transactions:
    Load
        *,
        quarterstart(date,0,3) as start_of_quarter,
        timestamp(quarterstart(date,0,3)) as start_of_quarter_timestamp
        ;
Load
*
Inline
[
id,date,amount
8188,1/7/2022,17.17
8189,1/19/2022,37.23
8190,2/28/2022,88.27
8191,2/5/2022,57.42
8192,3/16/2022,53.80
8193,4/1/2022,82.06
8194,5/7/2022,40.39
8195,5/16/2022,87.21
8196,6/15/2022,95.93
8197,6/26/2022,45.89
8198,7/9/2022,36.23
8199,7/22/2022,25.66
8200,7/23/2022,82.77
8201,7/27/2022,69.98
8202,8/2/2022,76.11
8203,8/8/2022,25.12
8204,8/19/2022,46.23
8205,9/26/2022,84.21
8206,10/14/2022,96.24
8207,10/29/2022,67.67
];
```

Результаты

Загрузите данные и откройте лист. Создайте новую таблицу и добавьте эти поля как измерения:

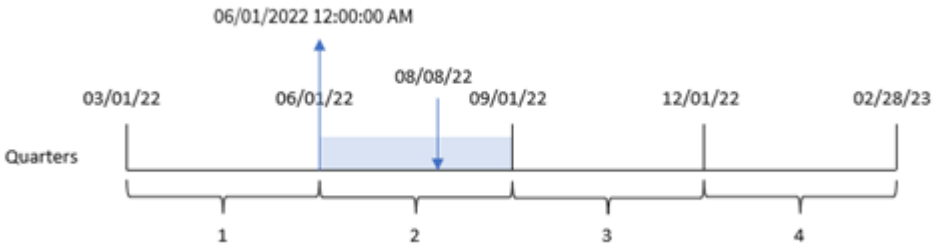
- date
- start_of_quarter
- start_of_quarter_timestamp

Результирующая таблица

date	start_of_quarter	start_of_quarter_timestamp
1/7/2022	12/01/2021	12/1/2021 12:00:00 AM
1/19/2022	12/01/2021	12/1/2021 12:00:00 AM
2/5/2022	12/01/2021	12/1/2021 12:00:00 AM
2/28/2022	12/01/2021	12/1/2021 12:00:00 AM
3/16/2022	03/01/2022	3/1/2022 12:00:00 AM
4/1/2022	03/01/2022	3/1/2022 12:00:00 AM
5/7/2022	03/01/2022	3/1/2022 12:00:00 AM
5/16/2022	03/01/2022	3/1/2022 12:00:00 AM
6/15/2022	06/01/2022	6/1/2022 12:00:00 AM
6/26/2022	06/01/2022	6/1/2022 12:00:00 AM
7/9/2022	06/01/2022	6/1/2022 12:00:00 AM
7/22/2022	06/01/2022	6/1/2022 12:00:00 AM
7/23/2022	06/01/2022	6/1/2022 12:00:00 AM
7/27/2022	06/01/2022	6/1/2022 12:00:00 AM
8/2/2022	06/01/2022	6/1/2022 12:00:00 AM
8/8/2022	06/01/2022	6/1/2022 12:00:00 AM
8/19/2022	06/01/2022	6/1/2022 12:00:00 AM
9/26/2022	09/01/2022	9/1/2022 12:00:00 AM
10/14/2022	09/01/2022	9/1/2022 12:00:00 AM
10/29/2022	09/01/2022	9/1/2022 12:00:00 AM

В этом примере, так как используется аргумент `first_month_of_year`= 3 в функции `quarterstart()`, начало года переносится с 1 января на 1 марта.

Диаграмма функции `quarterstart()`, пример с аргументом first_month_of_year



Транзакция 8203 совершена 8 августа. Так как год начинается 1 марта, он делится на кварталы март-май, июнь-август, сентябрь-ноябрь и декабрь-февраль. Функция `quarterstart()` определяет, что транзакция совершена в квартале с июня по август и возвращает первую миллисекунду этого квартала, то есть 00:00:00 (12:00:00 AM) 1 июня.

Пример 4. Пример объекта диаграммы

Скрипт загрузки и выражение диаграммы

Обзор

Откройте редактор загрузки данных и добавьте приведенный ниже скрипт загрузки на новую вкладку.

Скрипт загрузки содержит тот же набор данных и сценарий, что в первом примере.

Однако в этом примере в приложение загружается неизмененный набор данных. Расчет, возвращающий метку времени окончания квартала, в котором совершены транзакции, создается как мера в объекте диаграммы в приложении.

Скрипт загрузки

```
Transactions:
Load
*
Inline
[
id,date,amount
8188,1/7/2022,17.17
8189,1/19/2022,37.23
8190,2/28/2022,88.27
8191,2/5/2022,57.42
8192,3/16/2022,53.80
8193,4/1/2022,82.06
8194,5/7/2022,40.39
8195,5/16/2022,87.21
8196,6/15/2022,95.93
8197,6/26/2022,45.89
8198,7/9/2022,36.23
8199,7/22/2022,25.66
8200,7/23/2022,82.77
```

```
8201,7/27/2022,69.98
8202,8/2/2022,76.11
8203,8/8/2022,25.12
8204,8/19/2022,46.23
8205,9/26/2022,84.21
8206,10/14/2022,96.24
8207,10/29/2022,67.67
];
```

Результаты

Загрузите данные и откройте лист. Создайте новую таблицу и добавьте это поле как измерение: `date`.

Добавьте следующие меры:

- `=quarterstart(date)`
- `=timestamp(quarterstart(date))`

Результирующая таблица

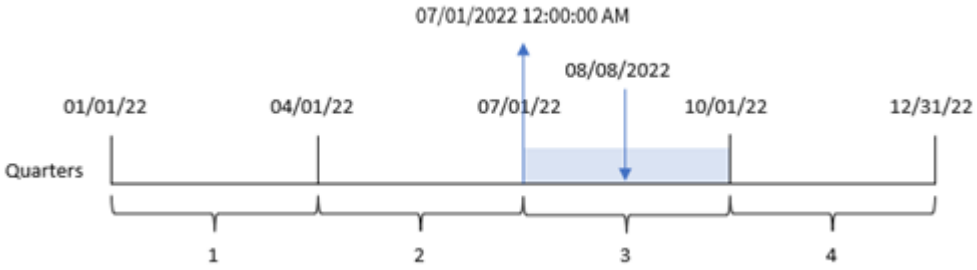
date	=quarterstart(date)	=timestamp(quarterstart(date))
10/14/2022	10/01/2022	10/1/2022 12:00:00 AM
10/29/2022	10/01/2022	10/1/2022 12:00:00 AM
7/9/2022	07/01/2022	7/1/2022 12:00:00 AM
7/22/2022	07/01/2022	7/1/2022 12:00:00 AM
7/23/2022	07/01/2022	7/1/2022 12:00:00 AM
7/27/2022	07/01/2022	7/1/2022 12:00:00 AM
8/2/2022	07/01/2022	7/1/2022 12:00:00 AM
8/8/2022	07/01/2022	7/1/2022 12:00:00 AM
8/19/2022	07/01/2022	7/1/2022 12:00:00 AM
9/26/2022	07/01/2022	7/1/2022 12:00:00 AM
4/1/2022	04/01/2022	4/1/2022 12:00:00 AM
5/7/2022	04/01/2022	4/1/2022 12:00:00 AM
5/16/2022	04/01/2022	4/1/2022 12:00:00 AM
6/15/2022	04/01/2022	4/1/2022 12:00:00 AM
6/26/2022	04/01/2022	4/1/2022 12:00:00 AM
1/7/2022	01/01/2022	1/1/2022 12:00:00 AM
1/19/2022	01/01/2022	1/1/2022 12:00:00 AM
2/5/2022	01/01/2022	1/1/2022 12:00:00 AM

date	=quarterstart(date)	=timestamp(quarterstart(date))
2/28/2022	01/01/2022	1/1/2022 12:00:00 AM
3/16/2022	01/01/2022	1/1/2022 12:00:00 AM

Мера `start_of_quarter` создана в объекте диаграммы с использованием функции `quarterstart()`, где в качестве аргумента функции передано поле `date`.

Функция `quarterstart()` определяет, к какому кварталу относится дата, и возвращает метку времени для первой миллисекунды этого квартала.

Диаграмма функции `quarterstart()`, пример с объектом диаграммы



Транзакция 8203 совершена 8 августа. Функция `quarterstart()` определяет, что транзакция совершена в третьем квартале, и возвращает первую миллисекунду этого квартала, то есть 00:00:00 (12:00:00 AM) 1 июля.

Пример 5. Сценарий

Скрипт загрузки и выражение диаграммы

Обзор

Откройте редактор загрузки данных и добавьте приведенный ниже скрипт загрузки на новую вкладку.

Скрипт загрузки содержит следующее:

- Набор данных, содержащий набор остатков по кредитам, загруженный в таблицу под именем `Loans`.
- Данные, включая идентификаторы кредитов, остаток на начало квартала и простую процентную ставку, взимаемую по каждому кредиту за год.

Конечному пользователю требуется объект диаграммы, который будет отображать по идентификатору кредита текущий процент, начисленный по каждому кредиту в течение квартала до текущей даты.

Скрипт загрузки

```
Loans:
Load
*
Inline
[
loan_id,start_balance,rate
8188,$10000.00,0.024
8189,$15000.00,0.057
8190,$17500.00,0.024
8191,$21000.00,0.034
8192,$90000.00,0.084
];
```

Результаты

Выполните следующие действия.

1. Загрузите данные и откройте лист. Создайте новую таблицу и добавьте эти поля как измерения:
   - `loan_id`
   - `start_balance`
2. Затем создайте эту меру, чтобы рассчитать накопленный процент:
   `=start_balance*(rate*(today(1)-quarterstart(today(1)))/365)`
3. Задайте параметру меры **Формат чисел** значение **Денежный**.

Результирующая таблица

loan_id	start_balance	=start_balance*(rate*(today(1)-quarterstart(today(1)))/365)
8188	$10000.00	$15.07
8189	$15000.00	$128.84
8190	$17500.00	$63.29
8191	$21000.00	$107.59
8192	$90000.00	$1139.18

Используя сегодняшнюю дату в качестве единственного аргумента, функция `quarterstart()` возвращает дату начала текущего года. Вычитая этот результат из текущей даты, выражение возвращает количество дней, прошедших до сих пор в этом квартале.

Затем это значение умножается на процентную ставку и делится на 365, чтобы получить эффективную процентную ставку начисленную за этот период. После этого результат умножается на начальный остаток кредита, чтобы вернуть проценты, начисленные до сих пор в течение этого квартала.

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

Когда это следует использовать

Функция `second()` полезна, когда требуется сравнить агрегирования по секундам. Например, эту функцию можно использовать, если требуется просмотреть распределения количества действий по секундам.

Эти измерения можно создать либо в скрипте загрузки, используя функцию для создания поля в таблице основного календаря или используя функции непосредственно в диаграмме в качестве вычисляемого измерения.

Примеры функции

Пример	Результат
`second( '09:14:36' )`	возвращает 36
`second( '0.5555' )`	возвращает 55 (так как 0,5555 = 13:19:55)

Региональные настройки

Если не указано иное, в примерах, приведенных в данном разделе, используется следующий формат даты: ММ/ДД/ГГГГ. Формат даты указан в операторе `SET DateFormat` скрипта загрузки данных. В вашей системе может быть установлен другой формат даты по умолчанию в зависимости от региональных настроек и других факторов. Можно изменить формат в примерах в соответствии с потребностями. Или можно изменить форматы в скрипте загрузки в соответствии с этими примерами.

Региональные настройки по умолчанию в приложениях основаны на системных региональных настройках компьютера или сервера, где установлено ПО Qlik Sense. Если на сервере Qlik Sense, к которому обращается пользователь, выбран шведский язык, то редактор загрузки данных будет использовать шведские региональные настройки для даты, времени и валюты. Эти параметры регионального формата не связаны с языком, отображаемым в интерфейсе пользователя Qlik Sense. Qlik Sense будет отображаться на языке, который используется в браузере.

Пример 1. Переменная

Скрипт загрузки и результаты

Обзор

Откройте редактор загрузки данных и добавьте приведенный ниже скрипт загрузки на новую вкладку.

Скрипт загрузки содержит следующее:

- Набор данных, содержащий транзакции по метке времени, загруженный в таблицу под именем `Transactions`.
- Используется системная переменная `TimeStamp` со значением по умолчанию `M/D/YYYY h:mm:ss[.fff] TT`.
- Создание поля `second` для вычисления времени, когда совершены покупки.

Скрипт загрузки

```
SET TimestampFormat='M/D/YYYY h:mm:ss[.fff] TT';

Transactions:
    Load
        *,
        second(date) as second
        ;
Load
*
Inline
[
id,date,amount
9497,'01/05/2022 7:04:57 PM',47.25
9498,'01/03/2022 2:21:53 PM',51.75
9499,'01/03/2022 5:40:49 AM',73.53
9500,'01/04/2022 6:49:38 PM',15.35
9501,'01/01/2022 10:10:22 PM',31.43
9502,'01/05/2022 7:34:46 PM',13.24
9503,'01/06/2022 10:58:34 PM',74.34
9504,'01/06/2022 11:29:38 AM',50.00
9505,'01/02/2022 8:35:54 AM',36.34
9506,'01/06/2022 8:49:09 AM',74.23
];
```

Результаты

Загрузите данные и откройте лист. Создайте новую таблицу и добавьте эти поля как измерения:

- `date`
- `second`

Результирующая таблица

date	second
01/01/2022 10:10:22 PM	22
01/02/2022 8:35:54 AM	54
01/03/2022 5:40:49 AM	49
01/03/2022 2:21:53 PM	53
01/04/2022 6:49:38 PM	38
01/05/2022 7:04:57 PM	57
01/05/2022 7:34:46 PM	46
01/06/2022 8:49:09 AM	9
01/06/2022 11:29:38 AM	38
01/06/2022 10:58:34 PM	34

Значение в поле `second` создаются с использованием функции `second()` и путем передачи даты в качестве выражения в предшествующем операторе load.

Пример 2. Объект диаграммы

Скрипт загрузки и выражение диаграммы

Обзор

Откройте редактор загрузки данных и добавьте приведенный ниже скрипт загрузки на новую вкладку.

Скрипт загрузки содержит тот же набор данных и сценарий, что в первом примере. Однако в этом примере в приложение загружается неизмененный набор данных. Значения `second` рассчитывается с использованием меры в объекте диаграммы.

Скрипт загрузки

```
SET TimestampFormat='M/D/YYYY h:mm:ss[.fff] TT';

Transactions:
Load
*
Inline
[
id,date,amount
9497,'01/05/2022 7:04:57 PM',47.25
9498,'01/03/2022 2:21:53 PM',51.75
9499,'01/03/2022 5:40:49 AM',73.53
9500,'01/04/2022 6:49:38 PM',15.35
9501,'01/01/2022 10:10:22 PM',31.43
9502,'01/05/2022 7:34:46 PM',13.24
```

```
9503,'01/06/2022 10:58:34 PM',74.34
9504,'01/06/2022 11:29:38 AM',50.00
9505,'01/02/2022 8:35:54 AM',36.34
9506,'01/06/2022 8:49:09 AM',74.23
];
```

Результаты

Загрузите данные и откройте лист. Создайте новую таблицу и добавьте это поле как измерение:`date`.

Создайте следующую меру:

```
=second(date)
```

Результирующая таблица

date	=second(date)
01/01/2022 10:10:22 PM	22
01/02/2022 8:35:54 AM	54
01/03/2022 5:40:49 AM	49
01/03/2022 2:21:53 PM	53
01/04/2022 6:49:38 PM	38
01/05/2022 7:04:57 PM	57
01/05/2022 7:34:46 PM	46
01/06/2022 8:49:09 AM	9
01/06/2022 11:29:38 AM	38
01/06/2022 10:58:34 PM	34

Значения для `second` создаются с использованием функции `second()`, где дата передается в виде выражения в мере для объекта диаграммы.

Пример 3. Сценарий

Скрипт загрузки и выражения диаграммы

Обзор

Откройте редактор загрузки данных и добавьте приведенный ниже скрипт загрузки на новую вкладку.

Скрипт загрузки содержит следующее:

- Набор данных с метками времени, созданный для представления трафика на веб-сайт продажи билетов на определенный фестиваль. Эти метки времени и соответствующие значения `id` загружаются в таблицу под именем `Web_Traffic`.
- Используется системная переменная `TimeStamp` (`M/D/YYYY h:mm:ss[.fff] TT`).

В этом сценарии 10000 билетов поступили в продажу в 9:00 утра 20 мая 2021 года. Уже через минуту все билеты были распроданы.

Пользователю требуется объект диаграммы, который показывает количество посещений веб-сайта по секундам.

Скрипт загрузки

```
SET TimestampFormat='M/D/YYYY h:mm:ss[.fff] TT';

tmpTimeStampCreator:
load
    makedate(2022,05,20) as date
AutoGenerate 1;

join load
    maketime(9+floor(rand()*2),0,floor(rand()*59)) as time
autogenerate 10000;

Web_Traffic:
load
    recno() as id,
    timestamp(date + time) as timestamp
resident tmpTimeStampCreator;

drop table tmpTimeStampCreator;
```

Результаты

Выполните следующие действия.

1. Загрузите данные и откройте лист. Создайте новую таблицу.
2. Затем создайте вычисляемое измерение, используя следующее выражение:
   `=second(timestamp)`
3. Создайте меру агрегирования для вычисления общего количества входов:
   `=count(id)`

Таблица результатов может выглядеть, как показано ниже, но содержать другие значения меры агрегирования:

Результирующая таблица

second(timestamp)	=count(id)
0	150

second(timestamp)	=count(id)
1	184
2	163
3	178
4	179
5	158
6	177
7	169
8	149
9	186
10	169
11	179
12	186
13	182
14	180
15	153
16	191
17	203
18	158
19	159
20	163
+ 39 строк	

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 часовой пояс, определенный на компьютере, на котором работает подсистема Qlik.

Синтаксис:

```
TimeZone( )
```

Возвращаемые типы данных: двойное значение

Пример:

```
timezone( )
```

Если необходимо показать другой часовой пояс в мере в приложении, можно использовать функцию `localtime()` в мере.

today

Эта функция возвращает текущую дату. Эта функция возвращает значения в формате системной переменной `DateFormat`.

Синтаксис:

```
today([ timer_mode])
```

Возвращаемые типы данных: двойное значение

Функцию `today()` можно использовать в сценарии загрузки или в объектах диаграммы.

Значение `timer_mode` по умолчанию – 1.

Аргументы

Аргумент	Описание
timer_mode	Может иметь следующие значения: 0 (день последней завершенной загрузки данных) 1 (день вызова функции) 2 (день открытия приложения) *Если функция используется в скрипте загрузки данных, функция **timer_mode=0** выдает день последней завершенной загрузки данных, а **timer_mode=1** выдает день текущей загрузки данных.*

Примеры функции

Значение timer_mode	Результат при использовании в скрипте загрузки	Результат при использовании в объекте диаграммы
0	Возвращает дату в формате системной переменной `DateFormat` для даты последней успешной перезагрузки, предшествующей самой последней перезагрузке данных.	Возвращает дату в формате системной переменной `DateFormat` для даты самой последней перезагрузки данных.
1	Возвращает дату в формате системной переменной `DateFormat` для самой последней перезагрузки данных.	Возвращает дату вызова функции в формате системной переменной `DateFormat`.
2	Возвращает дату в формате системной переменной `DateFormat` для начала сеанса пользователя в приложении. Это значение не будет обновляться, если пользователь не перезагрузит скрипт.	Возвращает дату в формате системной переменной `DateFormat` для начала сеанса пользователя в приложении. Это значение будет обновляться в случае начала нового сеанса или перезагрузки данных в приложение.

Когда это следует использовать

Функция `today()` часто используется как компонент внутри выражения. Например, можно ее использовать, если требуется рассчитать проценты, накопленные в течение месяца до текущей даты.

В следующей таблице приводится объяснение результата, возвращаемого функцией `today()`, в зависимости от различных значений аргумента `timer_mode`:

Региональные настройки

Если не указано иное, в примерах, приведенных в данном разделе, используется следующий формат даты: ММ/ДД/ГГГГ. Формат даты указан в операторе `SET DateFormat` скрипта загрузки данных. В вашей системе может быть установлен другой формат даты по умолчанию в зависимости от региональных настроек и других факторов. Можно изменить формат в примерах в соответствии с потребностями. Или можно изменить форматы в скрипте загрузки в соответствии с этими примерами.

Региональные настройки по умолчанию в приложениях основаны на системных региональных настройках компьютера или сервера, где установлено ПО Qlik Sense. Если на сервере Qlik Sense, к которому обращается пользователь, выбран шведский язык, то редактор загрузки данных будет использовать шведские региональные настройки для даты, времени и валюты. Эти параметры регионального формата не связаны с языком, отображаемым в интерфейсе пользователя Qlik Sense. Qlik Sense будет отображаться на языке, который используется в браузере.

Пример 1. Создание объектов с использованием скрипта загрузки

Скрипт загрузки и результаты

Обзор

Следующий пример создает три переменные с использованием функции `today()`. Каждая переменная использует один из параметров `timer_mode` для демонстрации эффекта.

Чтобы продемонстрировать назначение переменных, перезагрузите скрипт, а затем через 24 часа выполните вторую перезагрузку. Это приведет к тому, что переменные `today(0)` и `today(1)` будут показывать разные значения, тем самым правильно демонстрируя свое назначение.

Скрипт загрузки

```
LET vPreviousDataLoad = today(0);
LET vCurrentDataLoad = today(1);
LET vApplicationOpened = today(2);
```

Результаты

После загрузки данных во второй раз создайте три текстовых поля, следуя приведенным ниже инструкциям.

Сначала создайте текстовое поле для данных, загруженных ранее.

Выполните следующие действия.

1. Создайте текстовое поле, используя объект диаграммы **Текст и изображение**.
2. Добавьте в объект следующую меру:
   `=vPreviousDataLoad`

3. В области **Вид** выберите **Show titles** и добавьте в объект заголовок «Время предыдущей загрузки».

Затем создайте текстовое поле для данных, которые загружены в данный момент.

Выполните следующие действия.

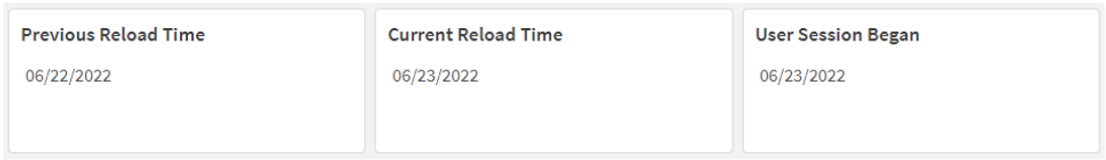
1. Создайте текстовое поле, используя объект диаграммы **Текст и изображение**.
2. Добавьте в объект следующую меру:
   `=vCurrentDataLoad`
3. В области **Вид** выберите **Show titles** и добавьте в объект заголовок «Время текущей загрузки».

Создайте последнее текстовое поле, в котором будет отображаться время начала сеанса пользователя в приложении.

Выполните следующие действия.

1. Создайте текстовое поле, используя объект диаграммы **Текст и изображение**.
2. Добавьте в объект следующую меру:
   `=vApplicationOpened`
3. В области **Вид** выберите **Show titles** и добавьте в объект заголовок «Начало сеанса пользователя».

Диаграмма с переменными, созданными с использованием функции `today()` в скрипте загрузки



В приведенной выше диаграмме показаны примерные значения для каждой из созданных переменных. Например, это могут быть следующие значения:

- Время предыдущей перезагрузки: 6/22/2022
- Время текущей перезагрузки: 06/23/2022
- Начало сеанса пользователя: 06/23/2022

Пример 2. Создание объектов без использования скрипта загрузки

Скрипт загрузки и выражение диаграммы

Обзор

Следующий пример создает три объекта диаграммы с использованием функции `today()`. Каждый объект диаграммы использует один из параметров `timer_mode` для демонстрации своего эффекта.

Для этого примера не предусмотрен скрипт загрузки.

Результаты

После загрузки данных во второй раз создайте три текстовых поля.

Сначала создайте текстовое поле для последней перезагрузки данных.

Выполните следующие действия.

1. Создайте текстовое поле, используя объект диаграммы **Текст и изображение**.
2. Добавьте следующую меру:
   `=today(0)`
3. В области **Вид** выберите **Show titles** и добавьте в объект заголовок «Последняя перезагрузка данных».

Затем создайте текстовое поле для отображения текущего времени.

Выполните следующие действия.

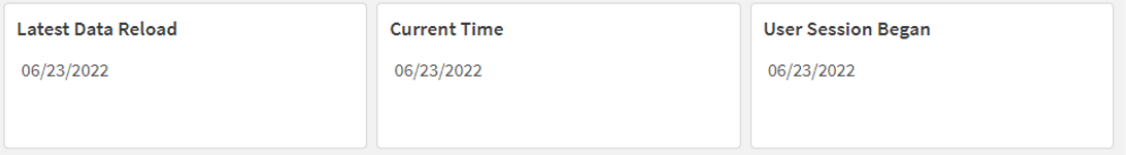
1. Создайте текстовое поле, используя объект диаграммы **Текст и изображение**.
2. Добавьте следующую меру:
   `=today(1)`
3. В области **Вид** выберите **Show titles** и добавьте в объект заголовок «Текущее время».

Создайте последнее текстовое поле, в котором будет отображаться время начала сеанса пользователя в приложении.

Выполните следующие действия.

1. Создайте текстовое поле, используя объект диаграммы **Текст и изображение**.
2. Добавьте следующую меру:
   `=today(2)`
3. В области **Вид** выберите **Show titles** и добавьте в объект заголовок «Начало сеанса пользователя».

Диаграмма с объектами, созданными с использованием функции `today()` в скрипте загрузки



В приведенной выше диаграмме показаны примерные значения для каждого из созданных объектов. Например, это могут быть следующие значения:

- Последняя перезагрузка данных: 06/23/2022
- Текущее время: 06/23/2022
- Начало сеанса пользователя: 06/23/2022

Объект диаграммы «Последняя перезагрузка данных» использует значение `timer_mode` = 0. В результате этого возвращается метка времени для последней успешной перезагрузки данных.

Объект диаграммы «Текущее время» использует значение `timer_mode` = 1. В результате этого возвращается текущее время по системным часам. В случае обновления листа или объекта это значение обновляется.

Объект диаграммы «Начало сеанса пользователя» использует `timer_mode` = 2. В результате этого возвращается метка времени, когда пользователь открыл приложение и начал сеанс.

Пример 3. Сценарий

Скрипт загрузки и выражение диаграммы

Обзор

Откройте редактор загрузки данных и добавьте приведенный ниже скрипт загрузки на новую вкладку.

Скрипт загрузки содержит следующее:

- Набор данных, содержащий набор остатков по кредитам, загруженный в таблицу под именем `Loans`.
- Данные таблицы с полями для идентификаторов кредитов, остатка на начало месяца и простой процентной ставки, взимаемой по каждому кредиту за год.

Конечному пользователю требуется объект диаграммы, который будет отображать по идентификатору кредита текущий процент, начисленный по каждому кредиту за год до текущей даты. Хотя приложение перезагружается только раз в неделю, пользователю требуется обновлять результаты при каждом обновлении объекта или приложения.

Скрипт загрузки

```
Loans:
Load
*
Inline
[
loan_id,start_balance,rate
8188,$10000.00,0.024
8189,$15000.00,0.057
8190,$17500.00,0.024
8191,$21000.00,0.034
8192,$90000.00,0.084
];
```

Результаты

Выполните следующие действия.

1. Загрузите данные и откройте лист. Создайте новую таблицу.
2. Добавьте следующие поля как измерения:
   - `loan_id`
   - `start_balance`
3. Затем создайте меру, чтобы рассчитать накопленный процент:
   `=start_balance*(rate*(today(1)-monthstart(today(1)))/365)`
4. Задайте параметру меры **Формат чисел** значение **Денежный**.

Результирующая таблица

loan_id	start_balance	=start_balance*(rate*(today(1)-monthstart(today(1)))/365)
8188	$10000.00	$16.44
8189	$15000.00	$58.56
8190	$17500.00	$28.77
8191	$21000.00	$48.90
8192	$90000.00	$517.81

Функция `monthstart()`, используя функцию `today()` для получения текущей даты в качестве единственного аргумента, возвращает дату начала текущего месяца. Вычитая этот результат из текущей даты, опять же с использованием функции `today()`, выражение возвращает количество дней, прошедших до сих пор в течение этого месяца.

Затем это значение умножается на процентную ставку и делится на 365, чтобы получить эффективную процентную ставку начисленную за этот период. После этого результат умножается на начальный остаток кредита, чтобы вернуть проценты, начисленные до сих пор в течение этого месяца.

Так как используется значение 1 для аргумента `timer_mode` в функциях `today()` внутри выражения, каждый раз при обновлении объекта диаграммы (при открытии приложения, обновлении страницы, переходе между листами и т. д.) будет возвращаться текущая дата, и результаты будут обновляться соответствующим образом.

UTC

Возвращает текущее время Coordinated Universal Time.

Синтаксис:

```
UTC( )
```

Возвращаемые типы данных: dual

Пример:

```
utc( )
```

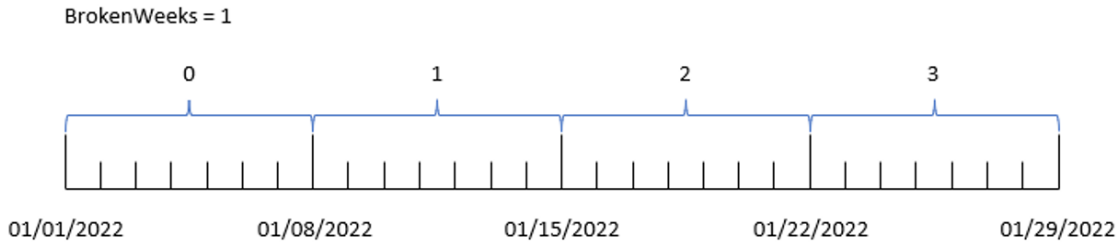
week

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:

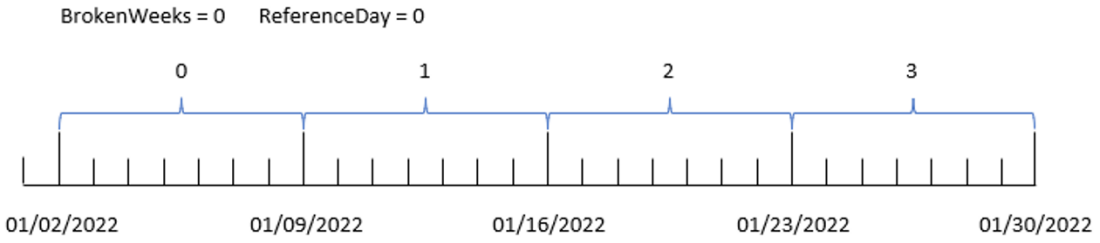
```
week(timestamp [, first_week_day [, broken_weeks [, reference_day]]])
```

Диаграмма с примером функции `week()` с неполными неделями



Отсчет номеров недель начинается с 1 января (так как по умолчанию задано Qlik Sense для использования неполных недель). Первая неделя заканчивается за день до системной переменной `FirstWeekDay` независимо от того, сколько дней прошло в этой неделе. Системная переменная `FirstWeekDay` может подавляться в функции `week()` аргументом `first_week_day`.

Диаграмма с примером функции week() с полными неделями и аргументом ReferenceDay=0



Функция `week()` также позволяет указать, следует использовать полные или неполные недели, с помощью аргумента `broken_weeks`. Если используется поддержка неполных недель, то неделя 1 должна включать определенное количество дней в январе, заданное системной переменной `ReferenceDay`. Таким образом, неделя 1 может начаться в декабре, а неделя 52 или 53 может заканчиваться в январе. В заключение аргумент `reference_day` позволяет функции переопределить системную переменную `ReferenceDay`.

В отличие от функции `weekname()`, функция `week()` не возвращает в результате значение года. Это позволяет при агрегировании сравнить недели разных годов.

В этой функции можно использовать четыре аргумента.

Аргумент №1: timestamp

Это дата для проверки в виде метки времени или выражения, определяемого по метке времени, которую требуется преобразовать, например «2012-10-12».

Аргумент №2: 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 №3: 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)

Аргумент №4: 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)

Когда это следует использовать

Функция `The week()` полезна, когда требуется сравнить агрегирования по неделям. Например, ее можно использовать, если требуется увидеть общий объем продаж продуктов по неделям. Функции `week()` отдается предпочтение перед `weekname()`, когда пользователю требуется, чтобы в вычислении не всегда использовались системные переменные приложения: `BrokenWeeks`, `FirstWeekDay` или `ReferenceDay`.

Кроме того, функция week() выбирается, когда требуется выполнить сравнение данных за несколько лет. С помощью функции week() пользователь может создать собственную комбинацию этих переменных для использования в ситуациях, когда применяется эта функция.

Эти измерения можно либо создать в скрипте загрузки, используя функцию для создания поля в таблице основного календаря, либо использовать непосредственно в диаграмме в качестве вычисляемого измерения.

Примеры функции

Пример	Результат
week('10/12/2012')	Возвращает 41.
week('35648')	Возвращает 32, так как 35 648 = 08/06/1997.
week('10/12/2012', 0, 1)	Возвращает 42.

Региональные настройки

Если не указано иное, в примерах, приведенных в данном разделе, используется следующий формат даты: ММ/ДД/ГГГГ. Формат даты указан в операторе SET DateFormat скрипта загрузки данных. В вашей системе может быть установлен другой формат даты по умолчанию в зависимости от региональных настроек и других факторов. Можно изменить формат в примерах в соответствии с потребностями. Или можно изменить форматы в скрипте загрузки в соответствии с этими примерами.

Региональные настройки по умолчанию в приложениях основаны на системных региональных настройках компьютера или сервера, где установлено ПО Qlik Sense. Если на сервере Qlik Sense, к которому обращается пользователь, выбран шведский язык, то редактор загрузки данных будет использовать шведские региональные настройки для даты, времени и валюты. Эти параметры регионального формата не связаны с языком, отображаемым в интерфейсе пользователя Qlik Sense. Qlik Sense будет отображаться на языке, который используется в браузере.

Пример 1. Системные переменные по умолчанию

Скрипт загрузки и результаты

Обзор

Откройте редактор загрузки данных и добавьте приведенный ниже скрипт загрузки на новую вкладку.

Скрипт загрузки содержит следующее:

- Набор данных, содержащий набор транзакций за последнюю неделю 2021 года, загруженный в таблицу под именем Transactions.
- Поле даты было предоставлено в формате системной переменной DateFormat (MM/DD/YYYY).

- Создание поля `week_number`, возвращающего год и номер недели, в течение которой совершены транзакции.
- Создание поля с именем `week_day`, которое отображает день недели для каждой даты транзакции.

Скрипт загрузки

```
SET DateFormat='MM/DD/YYYY';
SET FirstWeekDay=6;
SET BrokenWeeks=1;
SET ReferenceDay=0;

Transactions:
    Load
        *,
        WeekDay(date) as week_day,
        Week(date) as week_number
        ;
Load
*
Inline
[
id,date,amount
8183,12/27/2021,58.27
8184,12/28/2021,67.42
8185,12/29/2021,23.80
8186,12/30/2021,82.06
8187,12/31/2021,40.56
8188,01/01/2022,37.23
8189,01/02/2022,17.17
8190,01/03/2022,88.27
8191,01/04/2022,57.42
8192,01/05/2022,53.80
8193,01/06/2022,82.06
8194,01/07/2022,40.56
8195,01/08/2022,53.67
8196,01/09/2022,26.63
8197,01/10/2022,72.48
8198,01/11/2022,18.37
8199,01/12/2022,45.26
8200,01/13/2022,58.23
8201,01/14/2022,18.52
];
```

Результаты

Загрузите данные и откройте лист. Создайте новую таблицу и добавьте эти поля как измерения:

- `id`
- `date`

- week_day
- week_number

Результирующая таблица

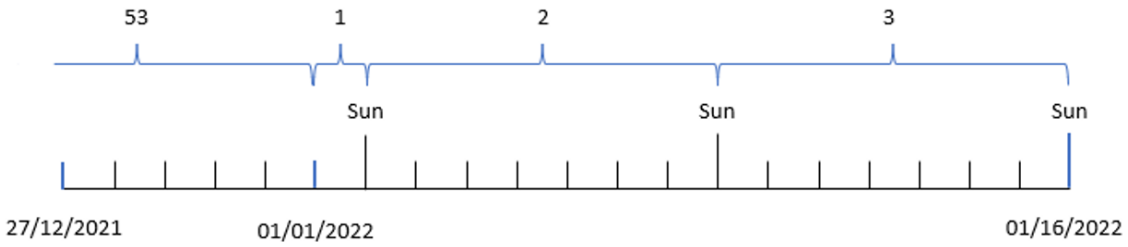
id	date	week_day	week_number
8183	12/27/2021	Mon	53
8184	12/28/2021	Tue	53
8185	12/29/2021	Wed	53
8186	12/30/2021	Thu	53
8187	12/31/2021	Fri	53
8188	01/01/2022	Sat	1
8189	01/02/2022	Sun	2
8190	01/03/2022	Mon	2
8191	01/04/2022	Tue	2
8192	01/05/2022	Wed	2
8193	01/06/2022	Thu	2
8194	01/07/2022	Fri	2
8195	01/08/2022	Sat	2
8196	01/09/2022	Sun	3
8197	01/10/2022	Mon	3
8198	01/11/2022	Tue	3
8199	01/12/2022	Wed	3
8200	01/13/2022	Thu	3
8201	01/14/2022	Fri	3

Поле «week_number» создано предшествующим оператором load с использованием функции week(), где в качестве аргумента функции передано поле date.

Другие параметры не передаются, поэтому следующие переменные по умолчанию оказывают влияние на выполнение функции week():

- BrokenWeeks: Отсчет недель начинается с 1 января
- FirstWeekDay: первым днем недели является воскресенье

Диаграмма функции week(), в которой используются системные переменные по умолчанию



Так как приложение использует системную переменную `BrokenWeeks` по умолчанию, неделя 1 начинается в субботу 1 января.

Из-за системной переменной `FirstWeekDay` по умолчанию неделя начинается в воскресенье. Первое воскресенье после 1 января выпадает на 2 января, это первый день недели 2.

Пример 2. Аргумент first_week_day

Скрипт загрузки и результаты

Обзор

Откройте редактор загрузки данных и добавьте приведенный ниже скрипт загрузки на новую вкладку.

Скрипт загрузки содержит следующее:

- Создание поля `week_number`, возвращающего год и номер недели, в течение которой совершены транзакции.
- Создание поля с именем `week_day`, которое отображает день недели для каждой даты транзакции.

В этом примере нам нужно задать вторник в качестве первого дня рабочей недели.

Скрипт загрузки

```
SET DateFormat='MM/DD/YYYY';
SET FirstWeekDay=6;
SET BrokenWeeks=1;
SET ReferenceDay=0;

Transactions:
    Load
        *,
        WeekDay(date) as week_day,
        Week(date,1) as week_number
        ;
Load
*
```

```
Inline
[
id,date,amount
8183,12/27/2022,58.27
8184,12/28/2022,67.42
8185,12/29/2022,23.80
8186,12/30/2022,82.06
8187,12/31/2021,40.56
8188,01/01/2022,37.23
8189,01/02/2022,17.17
8190,01/03/2022,88.27
8191,01/04/2022,57.42
8192,01/05/2022,53.80
8193,01/06/2022,82.06
8194,01/07/2022,40.56
8195,01/08/2022,53.67
8196,01/09/2022,26.63
8197,01/10/2022,72.48
8198,01/11/2022,18.37
8199,01/12/2022,45.26
8200,01/13/2022,58.23
8201,01/14/2022,18.52
];
```

Результаты

Загрузите данные и откройте лист. Создайте новую таблицу и добавьте эти поля как измерения:

- `id`
- `date`
- `week_day`
- `week_number`

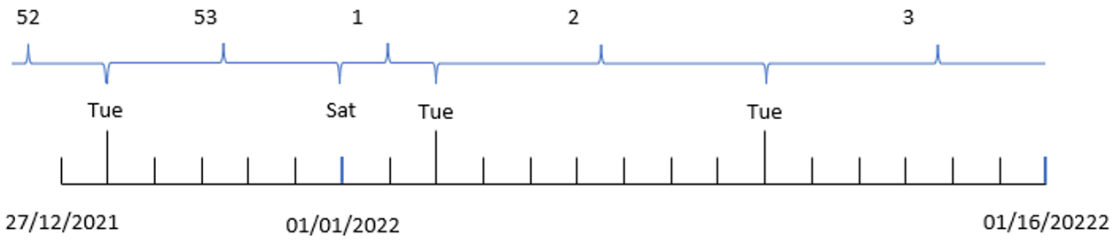
Результирующая таблица

id	date	week_day	week_number
8183	12/27/2021	Mon	52
8184	12/28/2021	Tue	53
8185	12/29/2021	Wed	53
8186	12/30/2021	Thu	53
8187	12/31/2021	Fri	53
8188	01/01/2022	Sat	1
8189	01/02/2022	Sun	1
8190	01/03/2022	Mon	1
8191	01/04/2022	Tue	2

id	date	week_day	week_number
8192	01/05/2022	Wed	2
8193	01/06/2022	Thu	2
8194	01/07/2022	Fri	2
8195	01/08/2022	Sat	2
8196	01/09/2022	Sun	2
8197	01/10/2022	Mon	2
8198	01/11/2022	Tue	3
8199	01/12/2022	Wed	3
8200	01/13/2022	Thu	3
8201	01/14/2022	Fri	3

Приложение по-прежнему использует неполные недели. Однако аргументу `first_week_day` задано значение 1 в функции `week()`. В результате этого первым днем недели становится вторник.

Диаграмма функции `week()` , пример с аргументом first_week_day



Приложение использует системную переменную `BrokenWeeks` по умолчанию, поэтому неделя 1 начинается в субботу 1 января.

Аргумент `first_week_day` функции `week()` устанавливает в качестве первого дня недели вторник. Таким образом, неделя 53 начинается 28 декабря 2021 года.

Однако поскольку функция по-прежнему использует неполные недели, неделя 1 будет включать только 2 дня, так как первый вторник после 1 января выпадает на 3 января.

Пример 3. Аргумент unbroken_weeks

Скрипт загрузки и результаты

Обзор

Откройте редактор загрузки данных и добавьте приведенный ниже скрипт загрузки на новую вкладку.

Скрипт загрузки содержит тот же набор данных и сценарий, что в первом примере.

В этом примере используются полные недели.

Скрипт загрузки

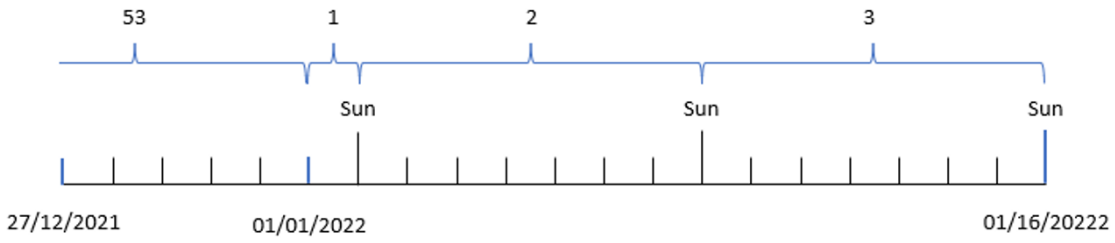
```
SET DateFormat='MM/DD/YYYY';
SET FirstWeekDay=6;
SET BrokenWeeks=1;
SET ReferenceDay=0;

Transactions:
    Load
        *,
        WeekDay(date) as week_day,
        Week(date,6,0) as week_number
        ;
Load
*
Inline
[
id,date,amount
8183,12/27/2022,58.27
8184,12/28/2022,67.42
8185,12/29/2022,23.80
8186,12/30/2022,82.06
8187,12/31/2021,40.56
8188,01/01/2022,37.23
8189,01/02/2022,17.17
8190,01/03/2022,88.27
8191,01/04/2022,57.42
8192,01/05/2022,53.80
8193,01/06/2022,82.06
8194,01/07/2022,40.56
8195,01/08/2022,53.67
8196,01/09/2022,26.63
8197,01/10/2022,72.48
8198,01/11/2022,18.37
8199,01/12/2022,45.26
8200,01/13/2022,58.23
8201,01/14/2022,18.52
];
```

Результаты

Загрузите данные и откройте лист. Создайте новую таблицу и добавьте эти поля как измерения:

- id
- date
- week_day
- week_number

Диаграмма функции week(), пример с объектом диаграммы



Результирующая таблица

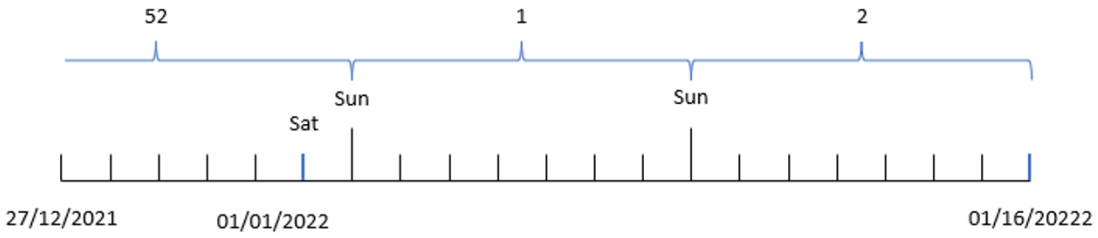
id	date	week_day	week_number
8183	12/27/2021	Mon	52
8184	12/28/2021	Tue	52
8185	12/29/2021	Wed	52
8186	12/30/2021	Thu	52
8187	12/31/2021	Fri	52
8188	01/01/2022	Sat	52
8189	01/02/2022	Sun	1
8190	01/03/2022	Mon	1
8191	01/04/2022	Tue	1
8192	01/05/2022	Wed	1
8193	01/06/2022	Thu	1
8194	01/07/2022	Fri	1
8195	01/08/2022	Sat	1
8196	01/09/2022	Sun	2
8197	01/10/2022	Mon	2
8198	01/11/2022	Tue	2

id	date	week_day	week_number
8199	01/12/2022	Wed	2
8200	01/13/2022	Thu	2
8201	01/14/2022	Fri	2

Параметру first_week_date задано значение 1, в результате чего первым днем недели становится вторник. Параметру broken_weeks задано значение 0, в результате чего функция использует полные недели. В заключение третий параметр задает reference_day = 2.

Параметру `first_week_date` задано значение 6, в результате чего первым днем недели становится воскресенье. Параметру `broken_weeks` задано значение 0, в результате чего функция использует полные недели.

Диаграмма функции week(), пример с использованием полных недель



При использовании полных недель неделя 1 необязательно начинается 1 января, но она должна содержать не меньше 4 дней января. Таким образом, в наборе данных неделя 52 заканчивается в субботу, 1 января 2022 года. В таком случае неделя 1 начинается с системой переменной `FirstWeekDay`, то есть в воскресенье, 2 января. Эта неделя заканчивается в следующую субботу, 8 января.

Пример 4. Аргумент reference_day

Скрипт загрузки и результаты

Обзор

Откройте редактор загрузки данных и добавьте приведенный ниже скрипт загрузки на новую вкладку.

Скрипт загрузки содержит следующее:

- Используется тот же набор данных и сценарий, что в третьем примере.
- Создание поля `week_number`, возвращающего год и номер недели, в течение которой совершены транзакции.

- Создание поля с именем `week_day`, которое отображает день недели для каждой даты транзакции.

Кроме того, должны соблюдаться следующие условия:

- Рабочая неделя начинается во вторник.
- Компания использует полные недели.
- Для `reference_day` задано значение 2. Другими словами, минимальное количество дней января в неделе 1 должно быть 2.

Скрипт загрузки

```
SET DateFormat='MM/DD/YYYY';
SET FirstWeekDay=6;
SET BrokenWeeks=1;
SET ReferenceDay=0;

Transactions:
    Load
        *,
        WeekDay(date) as week_day,
        Week(date,1,0,2) as week_number
        ;
Load
*
Inline
[
id,date,amount
8183,12/27/2022,58.27
8184,12/28/2022,67.42
8185,12/29/2022,23.80
8186,12/30/2022,82.06
8187,12/31/2021,40.56
8188,01/01/2022,37.23
8189,01/02/2022,17.17
8190,01/03/2022,88.27
8191,01/04/2022,57.42
8192,01/05/2022,53.80
8193,01/06/2022,82.06
8194,01/07/2022,40.56
8195,01/08/2022,53.67
8196,01/09/2022,26.63
8197,01/10/2022,72.48
8198,01/11/2022,18.37
8199,01/12/2022,45.26
8200,01/13/2022,58.23
8201,01/14/2022,18.52
];
```

Результаты

Загрузите данные и откройте лист. Создайте новую таблицу и добавьте эти поля как измерения:

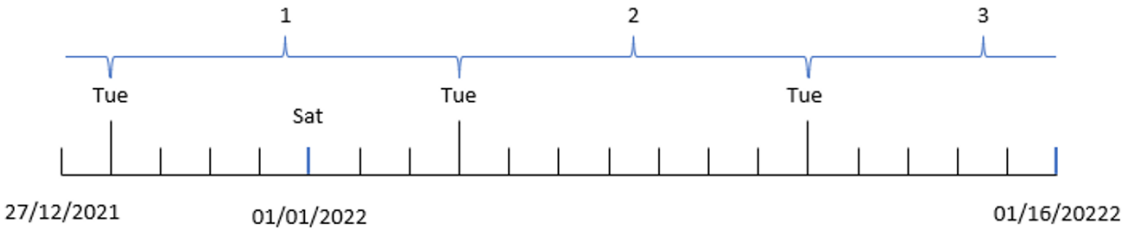
- id
- date
- week_day
- week_number

Результирующая таблица

id	date	week_day	week_number
8183	12/27/2021	Mon	52
8184	12/28/2021	Tue	1
8185	12/29/2021	Wed	1
8186	12/30/2021	Thu	1
8187	12/31/2021	Fri	1
8188	01/01/2022	Sat	1
8189	01/02/2022	Sun	1
8190	01/03/2022	Mon	1
8191	01/04/2022	Tue	2
8192	01/05/2022	Wed	2
8193	01/06/2022	Thu	2
8194	01/07/2022	Fri	2
8195	01/08/2022	Sat	2
8196	01/09/2022	Sun	2
8197	01/10/2022	Mon	2
8198	01/11/2022	Tue	3
8199	01/12/2022	Wed	3
8200	01/13/2022	Thu	3
8201	01/14/2022	Fri	3

Параметру `first_week_date` задано значение 1, в результате чего первым днем недели становится вторник. Параметру `broken_weeks` задано значение 0, в результате чего функция использует полные недели. В заключение, третий параметр задает параметр `reference_day` = 2.

Диаграмма функции week(), пример с аргументом reference_day



Так как функция использует полные недели и `reference_day` = 2 в качестве параметра, неделя 1 должна включать хотя бы два дня января. Так как первым днем недели является вторник, неделя 1 начинается 28 декабря 2021 года и заканчивается в понедельник, 3 января 2022 года.

Пример 5. Пример объекта диаграммы

Скрипт загрузки и выражение диаграммы

Обзор

Откройте редактор загрузки данных и добавьте приведенный ниже скрипт загрузки на новую вкладку.

Скрипт загрузки содержит тот же набор данных и сценарий, что в первом примере.

Однако в этом примере в приложение загружается неизмененный набор данных. Расчет, возвращающий номер недели, создается в качестве меры в объекте диаграммы.

Скрипт загрузки

```
Transactions:
Load
*
Inline
[
id,date,amount
8183,12/27/2022,58.27
8184,12/28/2022,67.42
8185,12/29/2022,23.80
8186,12/30/2022,82.06
8187,12/31/2021,40.56
8188,01/01/2022,37.23
8189,01/02/2022,17.17
8190,01/03/2022,88.27
8191,01/04/2022,57.42
8192,01/05/2022,53.80
8193,01/06/2022,82.06
8194,01/07/2022,40.56
8195,01/08/2022,53.67
8196,01/09/2022,26.63
```

```
8197,01/10/2022,72.48
8198,01/11/2022,18.37
8199,01/12/2022,45.26
8200,01/13/2022,58.23
8201,01/14/2022,18.52
];
```

Результаты

Выполните следующие действия.

1. Загрузите данные и откройте лист. Создайте новую таблицу.
2. Добавьте следующие поля как измерения:
   - `id`
   - `date`
3. Затем создайте следующую меру:
   `=week (date)`
4. Создайте меру , `week_day`, чтобы для каждой даты транзакции отображался день недели:
   `=weekday(date)`

Результирующая таблица

id	date	=week(date)	=weekday(date)
8183	12/27/2021	53	Mon
8184	12/28/2021	53	Tue
8185	12/29/2021	53	Wed
8186	12/30/2021	53	Thu
8187	12/31/2021	53	Fri
8188	01/01/2022	1	Sat
8189	01/02/2022	2	Sun
8190	01/03/2022	2	Mon
8191	01/04/2022	2	Tue
8192	01/05/2022	2	Wed
8193	01/06/2022	2	Thu
8194	01/07/2022	2	Fri
8195	01/08/2022	2	Sat
8196	01/09/2022	3	Sun
8197	01/10/2022	3	Mon
8198	01/11/2022	3	Tue

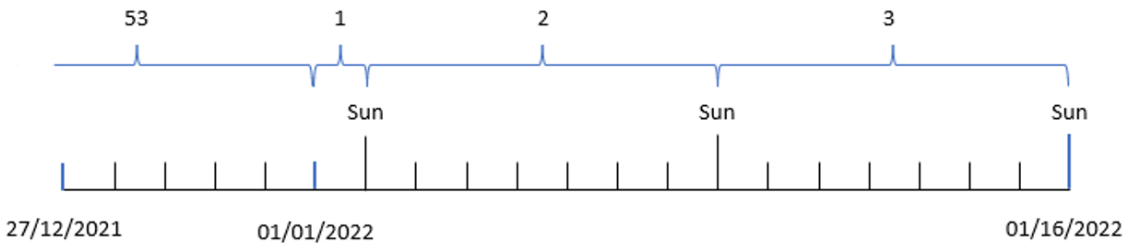
id	date	=week(date)	=weekday(date)
8199	01/12/2022	3	Wed
8200	01/13/2022	3	Thu
8201	01/14/2022	3	Fri

Поле «week_number» создано предшествующим оператором load с использованием функции `week()`, где в качестве аргумента функции передано поле `date`.

Другие параметры не передаются, поэтому следующие переменные по умолчанию оказывают влияние на выполнение функции `week()`:

- `BrokenWeeks`: Отсчет недель начинается с 1 января
- `FirstWeekDay`: Первым днем недели является воскресенье

Диаграмма функции `week()`, пример с объектом диаграммы



Так как приложение использует системную переменную `BrokenWeeks` по умолчанию, неделя 1 начинается в субботу 1 января.

Из-за системной переменной `FirstWeekDay` по умолчанию неделя начинается в воскресенье. Первое воскресенье после 1 января выпадает на 2 января, это первый день недели 2.

Пример 6. Сценарий

Скрипт загрузки и выражение диаграммы

Обзор

Откройте редактор загрузки данных и добавьте приведенный ниже скрипт загрузки на новую вкладку.

Скрипт загрузки содержит следующее:

- Набор данных, содержащий набор транзакций за последнюю неделю 2019 года и первые две недели 2020 года, загруженный в таблицу под именем `Transactions`.

- Поле даты было предоставлено в формате системной переменной `DateFormat` (MM/DD/YYYY).

Приложение в основном использует неполные недели на информационной панели. Однако конечному пользователю нужен объект диаграммы, на котором представлены общие продажи по неделям с использованием полных недель. В качестве дня ссылки следует задать 2 января, а неделя должна начинаться со вторника. Этого можно добиться, даже если это измерение недоступно в модели данных, используя в диаграмме функцию `week()` в качестве вычисляемого измерения.

Скрипт загрузки

```
SET BrokenWeeks=1;
SET ReferenceDay=0;
SET DateFormat='MM/DD/YYYY';

Transactions:
Load
*
Inline
[
id,date,amount
8183,12/27/2019,58.27
8184,12/28/2019,67.42
8185,12/29/2019,23.80
8186,12/30/2019,82.06
8187,12/31/2019,40.56
8188,01/01/2020,37.23
8189,01/02/2020,17.17
8190,01/03/2020,88.27
8191,01/04/2020,57.42
8192,01/05/2020,53.80
8193,01/06/2020,82.06
8194,01/07/2020,40.56
8195,01/08/2020,53.67
8196,01/09/2020,26.63
8197,01/10/2020,72.48
8198,01/11/2020,18.37
8199,01/12/2020,45.26
8200,01/13/2020,58.23
8201,01/14/2020,18.52
];
```

Результаты

Выполните следующие действия.

1. Загрузите данные и откройте лист. Создайте новую таблицу.
2. Создайте следующее вычисляемое измерение:
   `=week(date)`
3. Затем создайте следующую меру агрегирования:

`=sum(amount)`

4. Задайте параметру меры **Формат чисел** значение **Денежный**.
5. Выберите меню **Сортировка** и для вычисляемого изменения удалите пользовательскую сортировку.
6. Отмените выбор параметров **Сортировка по численным значениям** и **Сортировка по алфавиту**.

Результирующая таблица

week(date)	sum(amount)
52	$125.69
53	$146.42
1	$200.09
2	$347.57
3	$122.01

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).

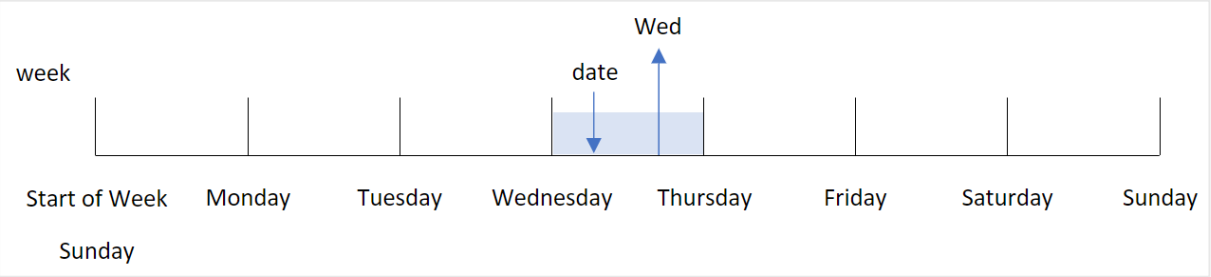
Синтаксис:

```
weekday(date [,first_week_day=0])
```

Возвращаемые типы данных: двойное значение

Функция `weekday()` определяет, на какой день недели выпадает дата. Затем она возвращает строковое значение, представляющее этот день.

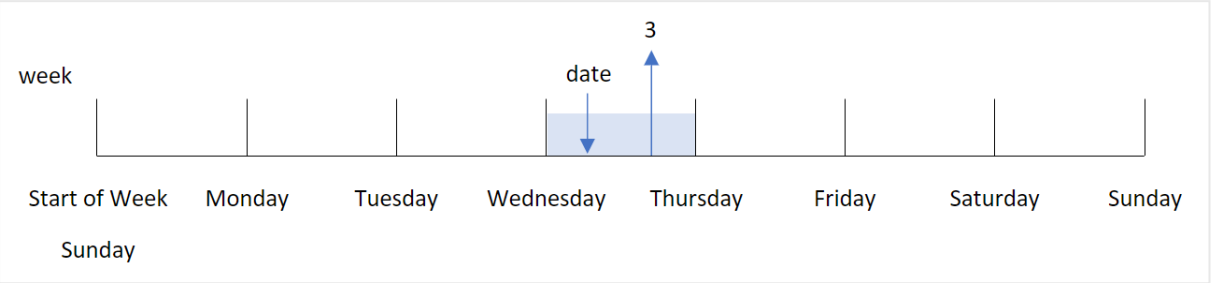
Диаграмма функции `weekday()`, возвращающей название дня недели, на который выпадает дата



Результат возвращает значение, соответствующее дню недели (0-6), в зависимости от первого дня недели. Например, если в качестве первого дня недели задано воскресенье, то для среды будет возвращаться значение `3`. Этот день начала недели определяется системной переменной `FirstWeekDay` или параметром функции `first_week_day`.

Это числовое значение можно использовать в составе арифметического выражения. Например, его можно умножить на 1, чтобы вернуть само значение.

Диаграмма функции weekday() с числовым значением дня, которое отображается вместо названия дня



Когда это следует использовать

Функция weekday() полезна, когда требуется сравнить агрегирования по дням недели, например, если требуется сравнить средние продажи продуктов по дням недели.

Эти измерения можно создать в скрипте загрузки, используя функцию для создания поля в таблице **основного календаря**, или непосредственно в диаграмме в качестве вычисляемого измерения.

Связанные темы

Темы	Взаимодействие
FirstWeekDay (page 225)	Определяет день начала каждой недели.

Аргументы

Аргумент	Описание
date	Дата или метка времени для вычисления.
first_week_day	Указывает день начала недели. Если не указано, используется значение переменной **FirstWeekDay**. *FirstWeekDay (page 225)*

Можно использовать следующие значения, чтобы задать первый день недели в аргументе first_week_day:

значения first_week_day

День	Значение
Понедельник	0
Вторник	1
Среда	2
Четверг	3

День	Значение
Пятница	4
Суббота	5
Воскресенье	6

Региональные настройки

Если не указано иное, в примерах, приведенных в данном разделе, используется следующий формат даты: ММ/ДД/ГГГГ. Формат даты указан в операторе `SET DateFormat` скрипта загрузки данных. В вашей системе может быть установлен другой формат даты по умолчанию в зависимости от региональных настроек и других факторов. Можно изменить формат в примерах в соответствии с потребностями. Или можно изменить форматы в скрипте загрузки в соответствии с этими примерами.

Региональные настройки по умолчанию в приложениях основаны на системных региональных настройках компьютера или сервера, где установлено ПО Qlik Sense. Если на сервере Qlik Sense, к которому обращается пользователь, выбран шведский язык, то редактор загрузки данных будет использовать шведские региональные настройки для даты, времени и валюты. Эти параметры регионального формата не связаны с языком, отображаемым в интерфейсе пользователя Qlik Sense. Qlik Sense будет отображаться на языке, который используется в браузере.

Если не указано иначе, в этих примерах для элемента `FirstWeekDay` установлено значение `0`.

Примеры функции

Пример	Результат
`weekday('10/12/1971')`	Возвращает Tue (вторник) и 1.
`weekday('10/12/1971' , 6)`	Возвращает Tue (вторник) и 2. В этом примере мы используем воскресенье (`6`) в качестве первого дня недели.

`SET FirstWeekDay=6;`
`...`
 	
`weekday('10/12/1971')`
	Возвращает Tue (вторник) и 2.

Пример 1. Строка с названием дня недели

Скрипт загрузки и результаты

Обзор

Откройте редактор загрузки данных и добавьте приведенный ниже скрипт загрузки на новую вкладку.

Скрипт загрузки содержит следующее:

- Набор данных, содержащий набор транзакций за 2022 год, загружаемый в таблицу под именем `Transactions`.
- Системная переменная `FirstWeekDay`, для которой задано значение 6 (воскресенье).
- Переменная `DayNames`, настроенная для использования названий дней недели по умолчанию.
- Предшествующая загрузка, содержащая функцию `weekday()`, которая настроена как поле `week_day` и возвращает день недели, когда совершены транзакции.

Скрипт загрузки

```
SET DateFormat='MM/DD/YYYY';
SET DayNames='Mon;Tue;Wed;Thu;Fri;Sat;Sun';
SET FirstWeekDay=6;

Transactions:
    Load
       *,
       WeekDay(date) as week_day
       ;
Load
*
Inline
[
id,date,amount
8188,01/01/2022,37.23
8189,01/02/2022,17.17
8190,01/03/2022,88.27
8191,01/04/2022,57.42
8192,01/05/2022,53.80
8193,01/06/2022,82.06
8194,01/07/2022,40.39
];
```

Результаты

Загрузите данные и откройте лист. Создайте новую таблицу и добавьте эти поля как измерения:

- `id`
- `date`
- `week_day`

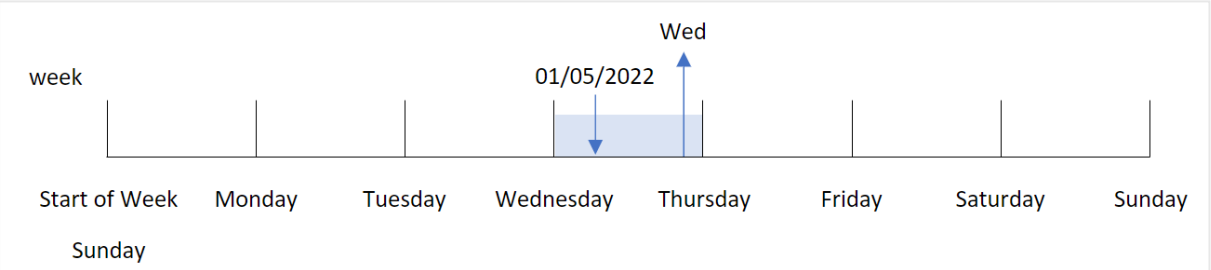
Результирующая таблица

id	date	week_day
8188	01/01/2022	Sat
8189	01/02/2022	Sun
8190	01/03/2022	Mon
8191	01/04/2022	Tue
8192	01/05/2022	Wed
8193	01/06/2022	Thu
8194	01/07/2022	Fri

Поле `week_day` создано предшествующим оператором load с использованием функции `weekday()`, где в качестве аргумента функции передано поле даты.

Функция `weekday()` возвращает строковое значение дня недели; другими словами, она возвращает название дня недели, заданное системной переменной `DayNames`.

Диаграмма функции `weekday()`, которая возвращает среду в качестве дня недели для транзакции 8192



Транзакция 8192 совершена 5 января. Системная переменная `FirstWeekDay` задает в качестве первого дня недели воскресенье. Функция `weekday()` определяет, что транзакция совершена в среду, и возвращает это значение как сокращенную форму системной переменной `DayNames` в поле `week_day`.

Значения в поле `week_day` выравниваются по правому краю столбца, так как это двойной текстово-числовой результат для поля (Wednesday, 3). Чтобы преобразовать значение поля в числовой эквивалент, можно поместить поле в функцию `num()`. Например, в транзакции 8192 значение Wednesday (среда) можно преобразовать в число 3.

Пример 2. Аргумент first_week_day

Скрипт загрузки и результаты

Обзор

Откройте редактор загрузки данных и добавьте приведенный ниже скрипт загрузки на новую вкладку.

Скрипт загрузки содержит следующее:

- Набор данных, содержащий набор транзакций за 2022 год, загружаемый в таблицу под именем `Transactions`.
- Системная переменная `FirstWeekDay`, для которой задано значение 6 (воскресенье).
- Переменная `DayNames`, настроенная для использования названий дней недели по умолчанию.
- Предшествующая загрузка, содержащая функцию `weekday()`, которая настроена как поле `week_day` и возвращает день недели, когда совершены транзакции.

Скрипт загрузки

```
SET DateFormat='MM/DD/YYYY';
SET DayNames='Mon;Tue;Wed;Thu;Fri;Sat;Sun';
SET FirstWeekDay=6;

Transactions:
    Load
       *,
       WeekDay(date,1) as week_day
       ;
Load
*
Inline
[
id,date,amount
8188,01/01/2022,37.23
8189,01/02/2022,17.17
8190,01/03/2022,88.27
8191,01/04/2022,57.42
8192,01/05/2022,53.80
8193,01/06/2022,82.06
8194,01/07/2022,40.39
];
```

Результаты

Загрузите данные и откройте лист. Создайте новую таблицу и добавьте эти поля как измерения:

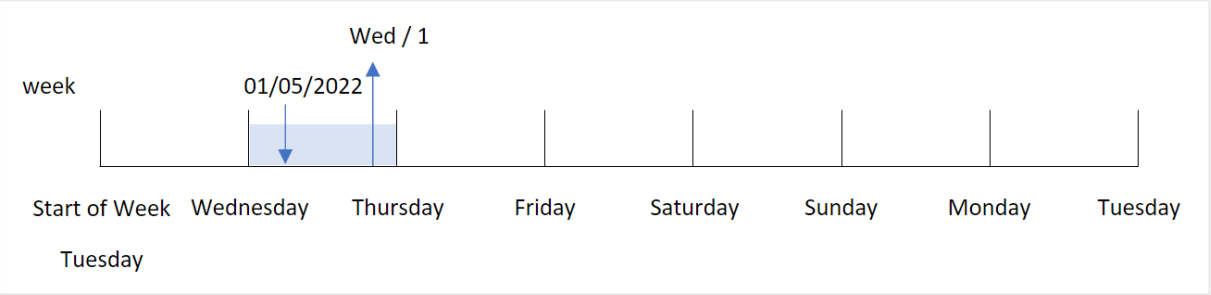
- `id`
- `date`
- `week_day`

Результирующая таблица

id	date	week_day
8188	01/01/2022	Sat
8189	01/02/2022	Sun
8190	01/03/2022	Mon

id	date	week_day
8191	01/04/2022	Tue
8192	01/05/2022	Wed
8193	01/06/2022	Thu
8194	01/07/2022	Fri

Диаграмма функции weekday(), которая показывает, что для среды предусмотрено двойное числовое значение 1



Поскольку используется аргумент `first_week_day = 1` в функции `weekday()`, первым днем недели является вторник. Таким образом, все транзакции, совершенные во вторник, имеют двойное числовое значение `0`.

Транзакция 8192 совершена 5 января. Функция `weekday()` определяет, что эта дата выпадает на среду, поэтому выражение возвращает двойное числовое значение `1`.

Пример 3. Пример объекта диаграммы

Скрипт загрузки и выражение диаграммы

Обзор

Откройте редактор загрузки данных и добавьте приведенный ниже скрипт загрузки на новую вкладку.

Скрипт загрузки содержит следующее:

- Набор данных, содержащий набор транзакций за 2022 год, загружаемый в таблицу под именем `Transactions`.
- Системная переменная `FirstWeekDay`, для которой задано значение `6` (воскресенье).
- Переменная `DayNames`, настроенная для использования названий дней недели по умолчанию.

Однако в этом примере в приложение загружается неизмененный набор данных. Расчет, который определяет день недели, создается как мера в диаграмме приложения.

Скрипт загрузки

```
SET DateFormat='MM/DD/YYYY';
SET DayNames='Mon;Tue;Wed;Thu;Fri;Sat;Sun';
SET FirstWeekDay=6;

Transactions:
Load
*
Inline
[
id,date,amount
8188,01/01/2022,37.23
8189,01/02/2022,17.17
8190,01/03/2022,88.27
8191,01/04/2022,57.42
8192,01/05/2022,53.80
8193,01/06/2022,82.06
8194,01/07/2022,40.39
];
```

Результаты

Загрузите данные и откройте лист. Создайте новую таблицу и добавьте эти поля как измерения:

- `id`
- `date`

Чтобы рассчитать значение дня недели, создайте следующую меру:

- `=weekday(date)`

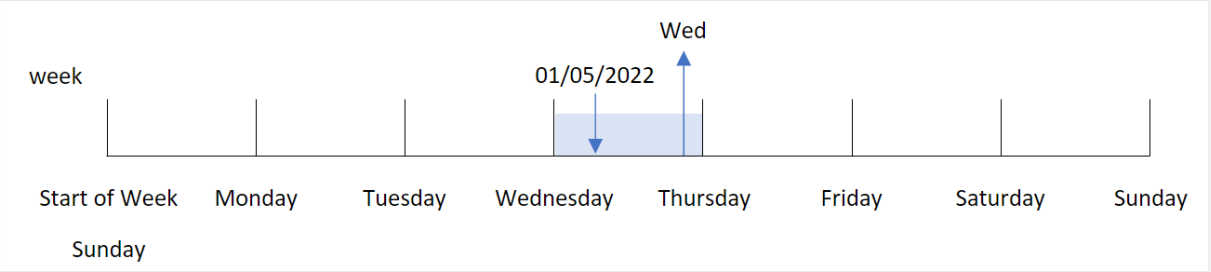
Результирующая таблица

id	date	=weekday(date)
8188	01/01/2022	Sat
8189	01/02/2022	Sun
8190	01/03/2022	Mon
8191	01/04/2022	Tue
8192	01/05/2022	Wed
8193	01/06/2022	Thu
8194	01/07/2022	Fri

Поле `=weekday(date)` создано в диаграмме с использованием функции `weekday()`, где в качестве аргумента функции передано поле даты.

Функция `weekday()` возвращает строковое значение дня недели; другими словами, она возвращает название дня недели, заданное системной переменной `DayNames`.

Диаграмма функции weekday(), которая возвращает среду в качестве дня недели для транзакции 8192



Транзакция 8192 совершена 5 января. Системная переменная `FirstWeekDay` задает в качестве первого дня недели воскресенье. Функция `weekday()` определяет, что транзакция совершена в среду, и возвращает это значение как сокращенную форму системной переменной `DayNames` в поле `=weekday(date)`.

Пример 4. Сценарий

Скрипт загрузки и результаты

Обзор

Откройте редактор загрузки данных и добавьте приведенный ниже скрипт загрузки на новую вкладку.

Скрипт загрузки содержит следующее:

- Набор данных, содержащий набор транзакций за 2022 год, загружаемый в таблицу под именем `Transactions`.
- Системная переменная `FirstWeekDay`, для которой задано значение `6` (воскресенье).
- Переменная `DayNames`, настроенная для использования названий дней недели по умолчанию.

Конечному пользователю нужна диаграмма, на которой представлены средние продажи для транзакций по дням недели.

Скрипт загрузки

```
SET DateFormat='MM/DD/YYYY';
SET DayNames='Mon;Tue;Wed;Thu;Fri;Sat;Sun';
SET FirstWeekDay=6;

Transactions:
LOAD
  RecNo() AS id,
  MakeDate(2022, 1, Ceil(Rand() * 31)) as date,
  Rand() * 1000 AS amount

Autogenerate(1000);
```

Результаты

Загрузите данные и откройте лист. Создайте новую таблицу и добавьте эти поля как измерения:

- `=weekday(date)`
- `=avg(amount)`

Задайте параметру **Формат чисел** меры значение **Денежный**.

Результирующая таблица

weekday(date)	Avg(amount)
Sun	$536.96
Mon	$500.80
Tue	$515.63
Wed	$509.21
Thu	$482.70
Fri	$441.33
Sat	$505.22

weekend

Эта функция возвращает значение, которое соответствует метке времени, равной последней миллисекунде последнего дня (воскресенье) календарной недели, и содержит **date**. По умолчанию для вывода используется формат **DateFormat**, заданный в скрипте.

Синтаксис:

```
WeekEnd(date [, period_no[, first_week_day]])
```

Возвращаемые типы данных: двойное значение

Диаграмма с примером функции weekend()

Функция `weekend()` определяет, на какой день недели приходится дата. Затем она возвращает метку времени в формате даты для последней миллисекунды этой недели. Первый день недели определяется переменной среды `FirstWeekDay`. Однако ее можно переопределить с помощью аргумента `first_week_day` в функции `weekend()`.

Аргументы

Аргумент	Описание
date	Дата или метка времени 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 для **first_week_day**: 0 – понедельник, 1 – вторник, 2 – среда, 3 – четверг, 4 – пятница, 5 – суббота и 6 – воскресенье. Для получения дополнительной информации о системной переменной см. *FirstWeekDay (page 225)*
broken_weeks	Если параметр **broken_weeks** не указан, значение переменной **BrokenWeeks** будет использовано для определения, какими должны быть недели: целыми или разбитыми.

Когда это следует использовать

Функция `weekend()` широко используется в составе выражения, когда пользователю требуется учитывать в расчетах оставшиеся дни недели, к которой относится указанная дата. Например, с ее помощью можно рассчитать общую сумму процентов, еще не начисленных в течение недели.

Пример	Результат
`weekend('01/10/2013')`	Возвращает 01/12/2013 23:59:59.
`weekend('01/10/2013', -1)`	Возвращает 01/05/2013 23:59:59..
`weekend('01/10/2013', 0, 1)`	Возвращает 01/14/2013 23:59:59.

Региональные настройки

Если не указано иное, в примерах, приведенных в данном разделе, используется следующий формат даты: ММ/ДД/ГГГГ. Формат даты указан в операторе `SET DateFormat` скрипта загрузки данных. В вашей системе может быть установлен другой формат даты по умолчанию в зависимости от региональных настроек и других факторов. Можно изменить формат в примерах в соответствии с потребностями. Или можно изменить форматы в скрипте загрузки в соответствии с этими примерами.

Региональные настройки по умолчанию в приложениях основаны на системных региональных настройках компьютера или сервера, где установлено ПО Qlik Sense. Если на сервере Qlik Sense, к которому обращается пользователь, выбран шведский язык, то редактор загрузки данных будет

использовать шведские региональные настройки для даты, времени и валюты. Эти параметры регионального формата не связаны с языком, отображаемым в интерфейсе пользователя Qlik Sense. Qlik Sense будет отображаться на языке, который используется в браузере.

Пример 1. Базовый пример

Скрипт загрузки и результаты

Обзор

Откройте редактор загрузки данных и добавьте приведенный ниже скрипт загрузки на новую вкладку.

Скрипт загрузки содержит следующее:

- Набор данных, содержащий набор транзакций за 2022 год, загружается в таблицу под именем `Transactions`.
- Поле даты было предоставлено в формате системной переменной `DateFormat` (MM/DD/YYYY).
- Создание поля `end_of_week`, возвращающего метку времени окончания недели, в течение которой совершены транзакции.

Скрипт загрузки

```
SET FirstWeekDay=6;

Transactions:
    Load
        *,
        weekend(date) as end_of_week,
        timestamp(weekend(date)) as end_of_week_timestamp
        ;
Load
*
Inline
[
id,date,amount
8188,1/7/2022,17.17
8189,1/19/2022,37.23
8190,2/28/2022,88.27
8191,2/5/2022,57.42
8192,3/16/2022,53.80
8193,4/1/2022,82.06
8194,5/7/2022,40.39
8195,5/16/2022,87.21
8196,6/15/2022,95.93
8197,6/26/2022,45.89
8198,7/9/2022,36.23
8199,7/22/2022,25.66
8200,7/23/2022,82.77
8201,7/27/2022,69.98
8202,8/2/2022,76.11
```

```
8203,8/8/2022,25.12
8204,8/19/2022,46.23
8205,9/26/2022,84.21
8206,10/14/2022,96.24
8207,10/29/2022,67.67
];
```

Результаты

Загрузите данные и откройте лист. Создайте новую таблицу и добавьте эти поля как измерения:

- `date`
- `end_of_week`
- `end_of_week_timestamp`

Результирующая таблица

date	end_of_week	end_of_week_timestamp
1/7/2022	01/08/2022	1/8/2022 11:59:59 PM
1/19/2022	01/22/2022	1/22/2022 11:59:59 PM
2/5/2022	02/05/2022	2/5/2022 11:59:59 PM
2/28/2022	03/05/2022	3/5/2022 11:59:59 PM
3/16/2022	03/19/2022	3/19/2022 11:59:59 PM
4/1/2022	04/02/2022	4/2/2022 11:59:59 PM
5/7/2022	05/07/2022	5/7/2022 11:59:59 PM
5/16/2022	05/21/2022	5/21/2022 11:59:59 PM
6/15/2022	06/18/2022	6/18/2022 11:59:59 PM
6/26/2022	07/02/2022	7/2/2022 11:59:59 PM
7/9/2022	07/09/2022	7/9/2022 11:59:59 PM
7/22/2022	07/23/2022	7/23/2022 11:59:59 PM
7/23/2022	07/23/2022	7/23/2022 11:59:59 PM
7/27/2022	07/30/2022	7/30/2022 11:59:59 PM
8/2/2022	08/06/2022	8/6/2022 11:59:59 PM
8/8/2022	08/13/2022	8/13/2022 11:59:59 PM
8/19/2022	08/20/2022	8/20/2022 11:59:59 PM
9/26/2022	10/01/2022	10/1/2022 11:59:59 PM
10/14/2022	10/15/2022	10/15/2022 11:59:59 PM
10/29/2022	10/29/2022	10/29/2022 11:59:59 PM

Поле `end_of_week` создано предшествующим оператором load с использованием функции `weekend ()`, где в качестве аргумента функции передано поле даты.

Функция `weekend()` определяет, к какой неделе относится значение даты, и возвращает метку времени для последней миллисекунды этой недели.

Диаграмма функции `weekend()`, базовый пример

Транзакция 8191 совершена 5 февраля. Системная переменная `FirstWeekDay` задает в качестве первого дня недели воскресенье. Функция `weekend()` определяет, что первой субботой после 5 февраля и, следовательно, последним днем недели, является 5 февраля. Таким образом, значение `end_of_week` для этой транзакции возвращает последнюю миллисекунду этого дня, то есть 23:59:59 (11:59:59 PM) 5 февраля.

Пример 2. Скрипт period_no

Скрипт загрузки и результаты

Обзор

Откройте редактор загрузки данных и добавьте приведенный ниже скрипт загрузки на новую вкладку.

Скрипт загрузки содержит следующее:

- Используется тот же набор данных и сценарий, что в первом примере.
- Создание поля `previous_week_end`, возвращающего метку времени начала недели, которая предшествует совершению транзакции.

Скрипт загрузки

```
SET DateFormat='MM/DD/YYYY';

Transactions:
    Load
        *,
        weekend(date,-1) as previous_week_end,
        timestamp(weekend(date,-1)) as previous_week_end_timestamp
        ;
Load
*
Inline
[
id,date,amount
8188,1/7/2022,17.17
```

```
8189,1/19/2022,37.23
8190,2/28/2022,88.27
8191,2/5/2022,57.42
8192,3/16/2022,53.80
8193,4/1/2022,82.06
8194,5/7/2022,40.39
8195,5/16/2022,87.21
8196,6/15/2022,95.93
8197,6/26/2022,45.89
8198,7/9/2022,36.23
8199,7/22/2022,25.66
8200,7/23/2022,82.77
8201,7/27/2022,69.98
8202,8/2/2022,76.11
8203,8/8/2022,25.12
8204,8/19/2022,46.23
8205,9/26/2022,84.21
8206,10/14/2022,96.24
8207,10/29/2022,67.67
];
```

Результаты

Загрузите данные и откройте лист. Создайте новую таблицу и добавьте эти поля как измерения:

- `date`
- `previous_week_end`
- `previous_week_end_timestamp`

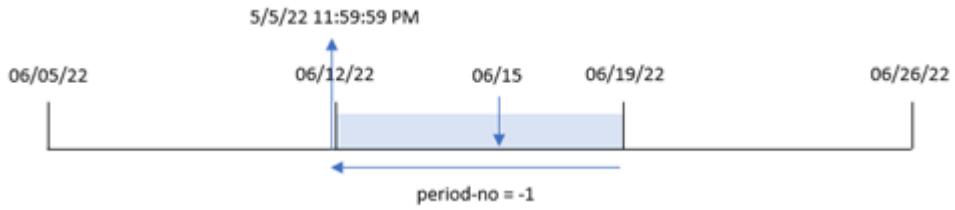
Результирующая таблица

date	end_of_week	end_of_week_timestamp
1/7/2022	01/01/2022	1/1/2022 11:59:59 PM
1/19/2022	01/15/2022	1/15/2022 11:59:59 PM
2/5/2022	01/29/2022	1/29/2022 11:59:59 PM
2/28/2022	02/26/2022	2/26/2022 11:59:59 PM
3/16/2022	03/12/2022	3/12/2022 11:59:59 PM
4/1/2022	03/26/2022	3/26/2022 11:59:59 PM
5/7/2022	04/30/2022	4/30/2022 11:59:59 PM
5/16/2022	05/14/2022	5/14/2022 11:59:59 PM
6/15/2022	06/11/2022	6/11/2022 11:59:59 PM
6/26/2022	06/25/2022	6/25/2022 11:59:59 PM
7/9/2022	07/02/2022	7/2/2022 11:59:59 PM
7/22/2022	07/16/2022	7/16/2022 11:59:59 PM

date	end_of_week	end_of_week_timestamp
7/23/2022	07/16/2022	7/16/2022 11:59:59 PM
7/27/2022	07/23/2022	7/23/2022 11:59:59 PM
8/2/2022	07/30/2022	7/30/2022 11:59:59 PM
8/8/2022	08/06/2022	8/6/2022 11:59:59 PM
8/19/2022	08/13/2022	8/13/2022 11:59:59 PM
9/26/2022	09/24/2022	9/24/2022 11:59:59 PM
10/14/2022	10/08/2022	10/8/2022 11:59:59 PM
10/29/2022	10/22/2022	10/22/2022 11:59:59 PM

В этом случае, так как в качестве аргумента смещения в функции `weekend()` использовалось `period_no` = -1, функция сначала определяет неделю, в течение которой совершены транзакции. Затем она возвращается на неделю назад и определяет последнюю миллисекунду предыдущей недели.

Диаграмма функции `weekend()`, пример с аргументом period_no



Транзакция 8196 совершена 15 июня. Функция `weekend()` определяет, что неделя начинается 12 июня. Таким образом, предыдущая неделя заканчивается 11 июня в 23:59:59 (11:59:59 PM). Именно это значение возвращается для поля `previous_week_end`.

Пример 3. Аргумент first_week_day

Скрипт загрузки и результаты

Обзор

Откройте редактор загрузки данных и добавьте приведенный ниже скрипт загрузки на новую вкладку.

Скрипт загрузки содержит тот же набор данных и сценарий, что в первом примере. Однако в этом примере требуется задать вторник в качестве первого дня недели.

Скрипт загрузки

```
SET DateFormat='MM/DD/YYYY';

Transactions:
```

```
    Load
        *,
        weekend(date,0,1) as end_of_week,
        timestamp(weekend(date,0,1)) as end_of_week_timestamp,
        ;
Load
*
Inline
[
id,date,amount
8188,1/7/2022,17.17
8189,1/19/2022,37.23
8190,2/28/2022,88.27
8191,2/5/2022,57.42
8192,3/16/2022,53.80
8193,4/1/2022,82.06
8194,5/7/2022,40.39
8195,5/16/2022,87.21
8196,6/15/2022,95.93
8197,6/26/2022,45.89
8198,7/9/2022,36.23
8199,7/22/2022,25.66
8200,7/23/2022,82.77
8201,7/27/2022,69.98
8202,8/2/2022,76.11
8203,8/8/2022,25.12
8204,8/19/2022,46.23
8205,9/26/2022,84.21
8206,10/14/2022,96.24
8207,10/29/2022,67.67
];
```

Результаты

Загрузите данные и откройте лист. Создайте новую таблицу и добавьте эти поля как измерения:

- `date`
- `end_of_week`
- `end_of_week_timestamp`

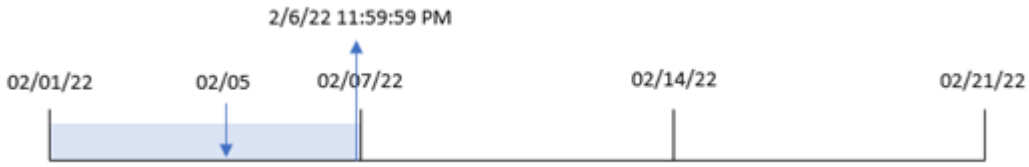
Результирующая таблица

date	end_of_week	end_of_week_timestamp
1/7/2022	01/10/2022	1/10/2022 11:59:59 PM
1/19/2022	01/24/2022	1/24/2022 11:59:59 PM
2/5/2022	02/07/2022	2/7/2022 11:59:59 PM
2/28/2022	02/28/2022	2/28/2022 11:59:59 PM
3/16/2022	03/21/2022	3/21/2022 11:59:59 PM

date	end_of_week	end_of_week_timestamp
4/1/2022	04/04/2022	4/4/2022 11:59:59 PM
5/7/2022	05/09/2022	7/9/2022 11:59:59 PM
5/16/2022	05/16/2022	5/16/2022 11:59:59 PM
6/15/2022	06/20/2022	6/20/2022 11:59:59 PM
6/26/2022	06/27/2022	6/27/2022 11:59:59 PM
7/9/2022	07/11/2022	7/11/2022 11:59:59 PM
7/22/2022	07/25/2022	7/25/2022 11:59:59 PM
7/23/2022	07/25/2022	7/25/2022 11:59:59 PM
7/27/2022	08/01/2022	8/1/2022 11:59:59 PM
8/2/2022	08/08/2022	8/8/2022 11:59:59 PM
8/8/2022	08/08/2022	8/8/2022 11:59:59 PM
8/19/2022	08/22/2022	8/22/2022 11:59:59 PM
9/26/2022	09/26/2022	9/26/2022 11:59:59 PM
10/14/2022	10/17/2022	10/17/2022 11:59:59 PM
10/29/2022	10/31/2022	10/31/2022 11:59:59 PM

В этом случае, поскольку аргумент `first_week_date` =1 используется в функции `weekend()`, в качестве первого дня недели задан вторник.

Диаграмма функции `weekend()` , пример с аргументом first_week_day



Транзакция 8191 совершена 5 февраля. Функция `weekend()` определяет, что первым понедельником после этой даты и, следовательно, последним днем недели является 6 февраля, и возвращается значение 23:59:59 (11:59:59 PM) 6 февраля.

Пример 4. Пример объекта диаграммы

Скрипт загрузки и выражение диаграммы

Обзор

Откройте редактор загрузки данных и добавьте приведенный ниже скрипт загрузки на новую вкладку.

Скрипт загрузки содержит тот же набор данных и сценарий, что в первом примере. Однако в этом примере в приложение загружается неизмененный набор данных. Расчет, возвращающий метку времени окончания недели, в котором совершены транзакции, создается как мера в объекте диаграммы в приложении.

Скрипт загрузки

```
Transactions:
Load
*
Inline
[
id,date,amount
8188,1/7/2022,17.17
8189,1/19/2022,37.23
8190,2/28/2022,88.27
8191,2/5/2022,57.42
8192,3/16/2022,53.80
8193,4/1/2022,82.06
8194,5/7/2022,40.39
8195,5/16/2022,87.21
8196,6/15/2022,95.93
8197,6/26/2022,45.89
8198,7/9/2022,36.23
8199,7/22/2022,25.66
8200,7/23/2022,82.77
8201,7/27/2022,69.98
8202,8/2/2022,76.11
8203,8/8/2022,25.12
8204,8/19/2022,46.23
8205,9/26/2022,84.21
8206,10/14/2022,96.24
8207,10/29/2022,67.67
];
```

Результаты

Загрузите данные и откройте лист. Создайте новую таблицу и добавьте это поле как измерение: `date`.

Чтобы рассчитать начало недели, когда была совершена транзакция, добавьте следующие меры:

- `=weekend(date)`
- `=timestamp(weekend(date))`

Результирующая таблица

date	=weekend(date)	=timestamp(weekend(date))
1/7/2022	01/08/2022	1/8/2022 11:59:59 PM
1/19/2022	01/22/2022	1/22/2022 11:59:59 PM

date	=weekend(date)	=timestamp(weekend(date))
2/5/2022	02/05/2022	2/5/2022 11:59:59 PM
2/28/2022	03/05/2022	3/5/2022 11:59:59 PM
3/16/2022	03/19/2022	3/19/2022 11:59:59 PM
4/1/2022	04/02/2022	4/2/2022 11:59:59 PM
5/7/2022	05/07/2022	5/7/2022 11:59:59 PM
5/16/2022	05/21/2022	5/21/2022 11:59:59 PM
6/15/2022	06/18/2022	6/18/2022 11:59:59 PM
6/26/2022	07/02/2022	7/2/2022 11:59:59 PM
7/9/2022	07/09/2022	7/9/2022 11:59:59 PM
7/22/2022	07/23/2022	7/23/2022 11:59:59 PM
7/23/2022	07/23/2022	7/23/2022 11:59:59 PM
7/27/2022	07/30/2022	7/30/2022 11:59:59 PM
8/2/2022	08/06/2022	8/6/2022 11:59:59 PM
8/8/2022	08/13/2022	8/13/2022 11:59:59 PM
8/19/2022	08/20/2022	8/20/2022 11:59:59 PM
9/26/2022	10/01/2022	10/1/2022 11:59:59 PM
10/14/2022	10/15/2022	10/15/2022 11:59:59 PM
10/29/2022	10/29/2022	10/29/2022 11:59:59 PM

Мера `end_of_week` создана в объекте диаграммы с использованием функции `weekend()`, где в качестве аргумента функции передано поле даты. Функция `weekend()` определяет, к какой неделе относится значение даты, и возвращает метку времени для последней миллисекунды этой недели.

Диаграмма функции `weekend()`, пример с объектом диаграммы

Транзакция 8191 совершена 5 февраля. Системная переменная `FirstWeekDay` задает в качестве первого дня недели воскресенье. Функция `weekend()` определяет, что первой субботой после 5 февраля и, следовательно, последним днем недели, является 5 февраля. Таким образом, значение `end_of_week` для этой транзакции возвращает последнюю миллисекунду этого дня, то есть 23:59:59 (11:59:59 PM) 5 февраля.

Пример 5. Сценарий

Скрипт загрузки и выражение диаграммы

Обзор

Откройте редактор загрузки данных и добавьте приведенный ниже скрипт загрузки на новую вкладку.

Скрипт загрузки содержит следующее:

- Набор данных, загруженный в таблицу, называется `Employee_Expenses`
- Данные, включающие идентификаторы сотрудников, имена сотрудников и средние ежедневные заявки на возмещение расходов каждого сотрудника.

Конечному пользователю требуется получить объект диаграммы, отображающий по идентификатору и имени сотрудника расчетные расходы, которые еще предстоит понести в течение оставшейся части недели.

Скрипт загрузки

```
Employee_Expenses:
Load
*
Inline
[
employee_id,employee_name,avg_daily_claim
182,Mark, $15
183,Deryck, $12.5
184,Dexter, $12.5
185,Sydney,$27
186,Agatha,$18
];
```

Результаты

Выполните следующие действия.

1. Загрузите данные и откройте лист. Создайте новую таблицу и добавьте эти поля как измерения:
   - `employee_id`
   - `employee_name`
2. Затем создайте меру, чтобы рассчитать накопленный процент:
   `=(weekend(today(1))-today(1))*avg_daily_claim`
3. Задайте параметру меры **Формат чисел** значение **Денежный**.

Результирующая таблица

employee_id	employee_name	=(weekend(today(1))-today(1))*avg_daily_claim
182	Mark	$90.00
183	Deryck	$75.00
184	Dexter	$75.00
185	Sydney	$162.00
186	Agatha	$108.00

Используя сегодняшнюю дату в качестве единственного аргумента, функция `weekend()` возвращает дату окончания текущей недели. Затем, вычитая сегодняшнюю дату из даты окончания недели, выражение возвращает количество дней, оставшихся в этой неделе.

Затем это значение умножается на среднюю ежедневную заявку на возмещение расходов каждого сотрудника для расчета оценочной суммы заявок, которые каждый сотрудник должен подать за оставшиеся дни недели.

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**.

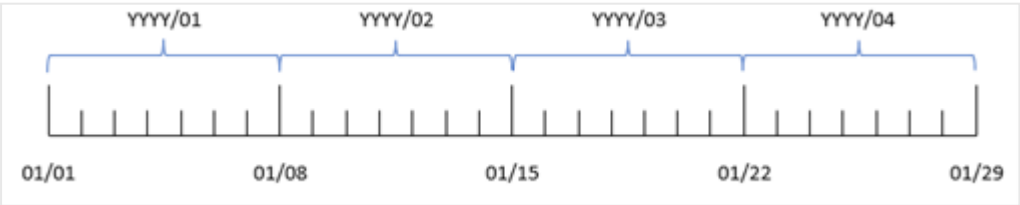
Синтаксис:

```
WeekName(date[, period_no[,first_week_day]])
```

Функция `weekname()` определяет, на какую неделю приходится дата, и возвращает номер недели и год этой недели. Первый день недели определяется системной переменной `FirstWeekDay`. Однако также можно изменить первый день недели, используя аргумент `first_week_day` в функции `weekname()`.

По умолчанию приложения Qlik Sense используют неполные недели (определяемые системной переменной `BrokenWeeks`), поэтому отсчет номера недели начинается с 1 января и заканчивается за день до системной переменной `FirstWeekDay` независимо от того, сколько дней прошло.

Схема функции weekname.



Однако если в приложении используются непрерывные недели, неделя 1 может начинаться в предыдущем году или в первые несколько дней января. Это зависит от того, как используются системные переменные `ReferenceDay` и `FirstWeekDay`.

Когда это следует использовать

Функция `weekname()` полезна, когда требуется сравнить агрегации по неделям.

Например, если требуется увидеть общий объем продаж продуктов по неделям. Чтобы не допустить противоречия с перемененной среды `BrokenWeeks` в приложении, используйте `weekname()` вместо `lunarweekname()`. Если приложение использует полные недели, неделя 1 может содержать даты декабря предыдущего года или исключать даты января текущего года. Если приложение использует неполные недели, неделя 1 может содержать менее семи дней.

Возвращаемые типы данных: двойное значение

Аргументы

Аргумент	Описание
date	Дата или метка времени 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 – понедельник, 1 – вторник, 2 – среда, 3 – четверг, 4 – пятница, 5 – суббота и 6 – воскресенье. Для получения дополнительной информации о системной переменной см. *FirstWeekDay (page 225)*.

Можно использовать следующие значения, чтобы задать первый день недели в аргументе `first_week_day`:

значения first_week_day

День	Значение
Понедельник	0
Вторник	1
Среда	2
Четверг	3
Пятница	4
Суббота	5
Воскресенье	6

Региональные настройки

Если не указано иное, в примерах, приведенных в данном разделе, используется следующий формат даты: MM/ДД/ГГГГ. Формат даты указан в операторе `SET DateFormat` скрипта загрузки данных. В вашей системе может быть установлен другой формат даты по умолчанию в зависимости от региональных настроек и других факторов. Можно изменить формат в примерах в соответствии с потребностями. Или можно изменить форматы в скрипте загрузки в соответствии с этими примерами.

Региональные настройки по умолчанию в приложениях основаны на системных региональных настройках компьютера или сервера, где установлено ПО Qlik Sense. Если на сервере Qlik Sense, к которому обращается пользователь, выбран шведский язык, то редактор загрузки данных будет использовать шведские региональные настройки для даты, времени и валюты. Эти параметры регионального формата не связаны с языком, отображаемым в интерфейсе пользователя Qlik Sense. Qlik Sense будет отображаться на языке, который используется в браузере.

Примеры функции

Пример	Результат
weekname('01/12/2013')	Возвращает 2013/02.
weekname('01/12/2013', -1)	Returns 2013/01.
weekname('01/12/2013', 0, 1)	Возвращает 2013/02.

Пример 1. Дата без дополнительных аргументов

Скрипт загрузки и результаты

Обзор

Откройте редактор загрузки данных и добавьте приведенный ниже скрипт загрузки на новую вкладку.

Скрипт загрузки содержит следующее:

- Набор данных, содержащий набор транзакций за последнюю неделю 2021 года, загружается в таблицу под именем `Transactions`.
- Системная переменная `DateFormat`, для которой задан формат `MM/DD/YYYY`.
- Системная переменная `BrokenWeeks`, для которой задан формат `1`.
- Системная переменная `FirstWeekDay`, для которой задан формат `6`.
- Предыдущая загрузка, которая содержит следующее:
    - Функция `weekday()`, заданная как поле `week_number`, возвращающая год и номер недели, когда были совершены транзакции.
    - Функция `weekname()`, заданная как поле с именем `week_day`, для отображения значения дня недели для каждой даты транзакции.

Скрипт загрузки

```
SET BrokenWeeks=1;
SET DateFormat='MM/DD/YYYY';
SET FirstWeekDay=6;

Transactions:
   Load
     *,
     WeekDay(date) as week_day,
     Weekname(date) as week_number
     ;
Load
*
Inline
[
id,date,amount
8183,12/27/2021,58.27
8184,12/28/2021,67.42
8185,12/29/2021,23.80
8186,12/30/2021,82.06
8187,12/31/2021,40.56
8188,01/01/2022,37.23
8189,01/02/2022,17.17
8190,01/03/2022,88.27
8191,01/04/2022,57.42
8192,01/05/2022,53.80
8193,01/06/2022,82.06
8194,01/07/2022,40.56
8195,01/08/2022,53.67
8196,01/09/2022,26.63
8197,01/10/2022,72.48
8198,01/11/2022,18.37
8199,01/12/2022,45.26
8200,01/13/2022,58.23
8201,01/14/2022,18.52
];
```

Результаты

Загрузите данные и откройте лист. Создайте новую таблицу и добавьте эти поля как измерения:

- id
- date
- week_day
- week_number

Результирующая таблица

id	date	week_day	week_number
8183	12/27/2021	Mon	2021/53

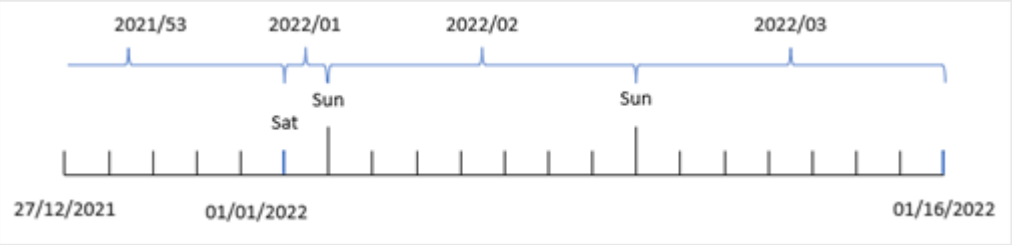
id	date	week_day	week_number
8184	12/28/2021	Tue	2021/53
8185	12/29/2021	Wed	2021/53
8186	12/30/2021	Thu	2021/53
8187	12/31/2021	Fri	2021/53
8188	01/01/2022	Sat	2022/01
8189	01/02/2022	Sun	2022/02
8190	01/03/2022	Mon	2022/02
8191	01/04/2022	Tue	2022/02
8192	01/05/2022	Wed	2022/02
8193	01/06/2022	Thu	2022/02
8194	01/07/2022	Fri	2022/02
8195	01/08/2022	Sat	2022/02
8196	01/09/2022	Sun	2022/03
8197	01/10/2022	Mon	2022/03
8198	01/11/2022	Tue	2022/03
8199	01/12/2022	Wed	2022/03
8200	01/13/2022	Thu	2022/03
8201	01/14/2022	Fri	2022/03

Поле `week_number` создано предшествующим оператором load с использованием функции `weekname()`, где в качестве аргумента функции передано поле даты.

Функция `weekname()` изначально определяет, на какую неделю приходится значение даты, и возвращает номер недели и год, когда была совершена транзакция.

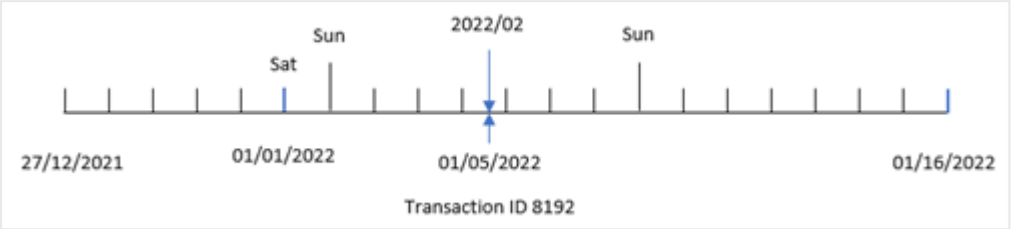
Системная переменная `FirstWeekDay` задает воскресенье как первый день недели. Системная переменная `BrokenWeeks` задает для приложения использование неполных недель, что означает, что неделя 1 начнется 1 января.

Диаграмма функции `weekname()` с переменными по умолчанию.



Неделя 1 начинается 1 января, то есть в субботу, поэтому транзакции, совершенные в этот день, возвращают значение `2022/01` (год и номер недели).

Диаграмма функции `weekname()`, определяющей номер недели для транзакции 8192.



Поскольку приложение использует неполные недели, а первый день недели – воскресенье, транзакции, совершенные со 2 по 8 января, возвращают значение `2022/02` (номер недели 2 в 2022 году). Примером этого может быть транзакция 8192, которая совершена 5 января и возвращает значение `2022/02` для поля `week_number`.

Пример 2. Скрипт period_no

Скрипт загрузки и результаты

Обзор

Используется тот же набор данных и сценарий, что в первом примере.

Однако в этом примере задача состоит в том, чтобы создать поле `previous_week_number`, которое возвращает год и номер недели, предшествующей той, на которой были совершены транзакции.

Откройте Редактор загрузки данных и добавьте следующий скрипт загрузки на новую вкладку.

Скрипт загрузки

```
SET BrokenWeeks=1;
SET FirstWeekDay=6;

Transactions:
   Load
     *,
     weekname(date,-1) as previous_week_number
     ;
Load
*
Inline
[
id,date,amount
8183,12/27/2021,58.27
8184,12/28/2021,67.42
8185,12/29/2021,23.80
8186,12/30/2021,82.06
8187,12/31/2021,40.56
8188,01/01/2022,37.23
8189,01/02/2022,17.17
```

```
8190,01/03/2022,88.27
8191,01/04/2022,57.42
8192,01/05/2022,53.80
8193,01/06/2022,82.06
8194,01/07/2022,40.56
8195,01/08/2022,53.67
8196,01/09/2022,26.63
8197,01/10/2022,72.48
8198,01/11/2022,18.37
8199,01/12/2022,45.26
8200,01/13/2022,58.23
8201,01/14/2022,18.52
];
```

Результаты

Загрузите данные и откройте лист. Создайте новую таблицу и добавьте эти поля как измерения:

- id
- date
- week_day
- week_number

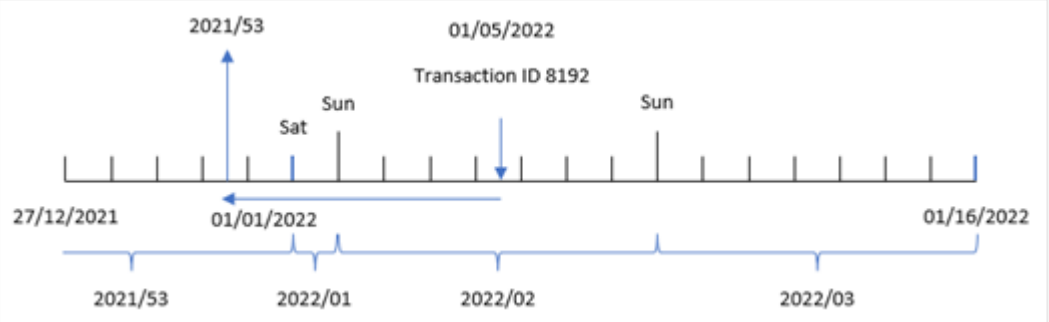
Результирующая таблица

id	date	week_day	week_number
8183	12/27/2021	Mon	2021/52
8184	12/28/2021	Tue	2021/52
8185	12/29/2021	Wed	2021/52
8186	12/30/2021	Thu	2021/52
8187	12/31/2021	Fri	2021/52
8188	01/01/2022	Sat	2021/52
8189	01/02/2022	Sun	2021/53
8190	01/03/2022	Mon	2021/53
8191	01/04/2022	Tue	2021/53
8192	01/05/2022	Wed	2021/53
8193	01/06/2022	Thu	2021/53
8194	01/07/2022	Fri	2021/53
8195	01/08/2022	Sat	2022/01
8196	01/09/2022	Sun	2022/02
8197	01/10/2022	Mon	2022/02

id	date	week_day	week_number
8198	01/11/2022	Tue	2022/02
8199	01/12/2022	Wed	2022/02
8200	01/13/2022	Thu	2022/02
8201	01/14/2022	Fri	2022/02

Так как в качестве аргумента смещения в функции `weekname()` использовалось `period_no = -1`, функция сначала определяет неделю, в течение которой совершены транзакции. Затем она возврващается на неделю назад и определяет первую миллисекунду этой недели.

Диаграмма функции `weekname()` со смещением `period_no = -1`.



Транзакция 8192 совершена 5 января 2022 года. Функция `weekname()` возвращается на неделю назад к 30 декабря 2021 года и возвращает номер недели и год для этой даты — `2021/53`.

Пример 3. Аргумент first_week_day

Скрипт загрузки и результаты

Обзор

Используется тот же набор данных и сценарий, что в первом примере.

Однако в этом примере политика компании такова, что рабочая неделя начинается во вторник.

Откройте Редактор загрузки данных и добавьте следующий скрипт загрузки на новую вкладку.

Скрипт загрузки

```
SET BrokenWeeks=1;
SET DateFormat='MM/DD/YYYY';

Transactions:
   Load
     *,
     weekday(date) as week_day,
     weekname(date,0,1) as week_number
     ;
Load
```

```
*
Inline
[
id,date,amount
8183,12/27/2021,58.27
8184,12/28/2021,67.42
8185,12/29/2021,23.80
8186,12/30/2021,82.06
8187,12/31/2021,40.56
8188,01/01/2022,37.23
8189,01/02/2022,17.17
8190,01/03/2022,88.27
8191,01/04/2022,57.42
8192,01/05/2022,53.80
8193,01/06/2022,82.06
8194,01/07/2022,40.56
8195,01/08/2022,53.67
8196,01/09/2022,26.63
8197,01/10/2022,72.48
8198,01/11/2022,18.37
8199,01/12/2022,45.26
8200,01/13/2022,58.23
8201,01/14/2022,18.52
];
```

Результаты

Загрузите данные и откройте лист. Создайте новую таблицу и добавьте эти поля как измерения:

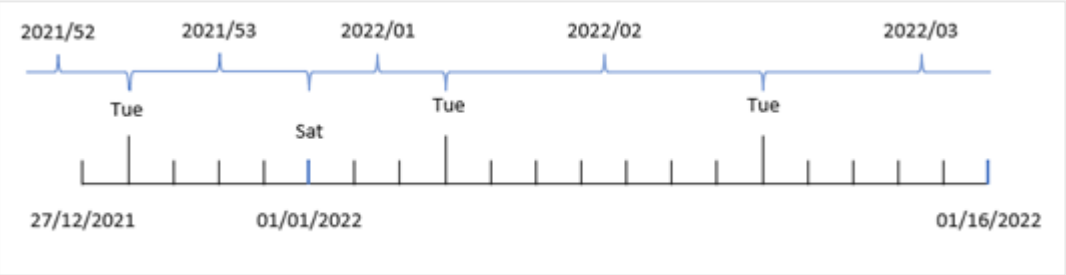
- id
- date
- week_day
- week_number

Результирующая таблица

id	date	week_day	week_number
8183	12/27/2021	Mon	2021/52
8184	12/28/2021	Tue	2021/53
8185	12/29/2021	Wed	2021/53
8186	12/30/2021	Thu	2021/53
8187	12/31/2021	Fri	2021/53
8188	01/01/2022	Sat	2022/01
8189	01/02/2022	Sun	2022/01
8190	01/03/2022	Mon	2022/01
8191	01/04/2022	Tue	2022/02

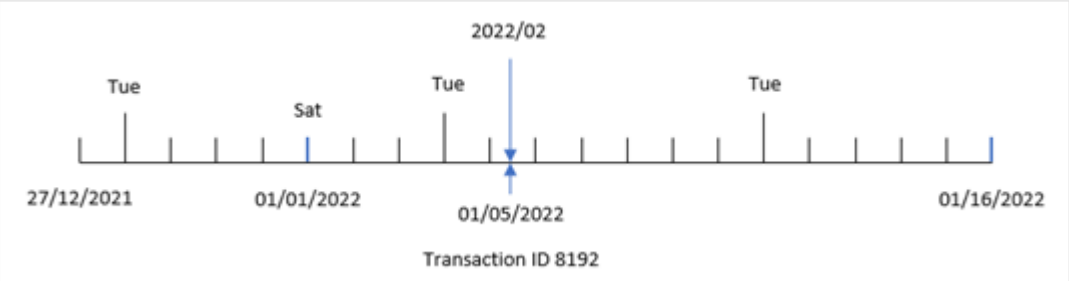
id	date	week_day	week_number
8192	01/05/2022	Wed	2022/02
8193	01/06/2022	Thu	2022/02
8194	01/07/2022	Fri	2022/02
8195	01/08/2022	Sat	2022/02
8196	01/09/2022	Sun	2022/02
8197	01/10/2022	Mon	2022/02
8198	01/11/2022	Tue	2022/03
8199	01/12/2022	Wed	2022/03
8200	01/13/2022	Thu	2022/03
8201	01/14/2022	Fri	2022/03

Диаграмма функции `weekname()`, где первым днем недели является вторник.



Поскольку аргумент `first_week_date` функции `1` используется в функции `weekname()`, в качестве первого дня недели используется вторник. Таким образом, функция определяет, что 53-я неделя 2021 года начинается во вторник, 28 декабря; и, поскольку в приложении используются неполные недели, неделя 1 начинается 1 января 2022 г. и заканчивается в последнюю миллисекунду понедельника, 3 января 2022 г.

Диаграмма, которая показывает номер недели для транзакции 8192, где первым днем недели является вторник.



Транзакция 8192 совершена 5 января 2022 года. Таким образом, используя для параметра `first_week_day` значение «вторник», функция `weekname()` возвращает значение `2022/02` для поля `week_number`.

Пример 4. Пример объекта диаграммы

Скрипт загрузки и выражение диаграммы

Обзор

Используется тот же набор данных и сценарий, что в первом примере.

Однако в этом примере в приложение загружается неизмененный набор данных. Расчет, который возвращает номер года для недели, когда была совершена транзакция, создается как мера в объекте диаграммы в приложении.

Скрипт загрузки

```
SET BrokenWeeks=1;
Transactions:
Load
*
Inline
[
id,date,amount
8183,12/27/2021,58.27
8184,12/28/2021,67.42
8185,12/29/2021,23.80
8186,12/30/2021,82.06
8187,12/31/2021,40.56
8188,01/01/2022,37.23
8189,01/02/2022,17.17
8190,01/03/2022,88.27
8191,01/04/2022,57.42
8192,01/05/2022,53.80
8193,01/06/2022,82.06
8194,01/07/2022,40.56
8195,01/08/2022,53.67
8196,01/09/2022,26.63
8197,01/10/2022,72.48
8198,01/11/2022,18.37
8199,01/12/2022,45.26
8200,01/13/2022,58.23
8201,01/14/2022,18.52
];
```

Результаты

Загрузите данные и откройте лист. Создайте новую таблицу и добавьте эти поля как измерения:

- `id`
- `date`
- `=week_day (date)`

Чтобы рассчитать начало недели, когда была совершена транзакция, создайте следующую меру:

```
=weekname(date)
```

Результирующая таблица

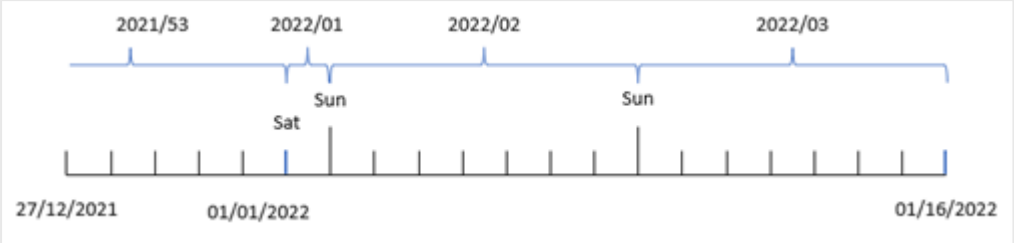
id	date	=weekday(date)	=weekname(date)
8183	12/27/2021	Mon	2021/53
8184	12/28/2021	Tue	2021/53
8185	12/29/2021	Wed	2021/53
8186	12/30/2021	Thu	2021/53
8187	12/31/2021	Fri	2021/53
8188	01/01/2022	Sat	2022/01
8189	01/02/2022	Sun	2022/02
8190	01/03/2022	Mon	2022/02
8191	01/04/2022	Tue	2022/02
8192	01/05/2022	Wed	2022/02
8193	01/06/2022	Thu	2022/02
8194	01/07/2022	Fri	2022/02
8195	01/08/2022	Sat	2022/02
8196	01/09/2022	Sun	2022/03
8197	01/10/2022	Mon	2022/03
8198	01/11/2022	Tue	2022/03
8199	01/12/2022	Wed	2022/03
8200	01/13/2022	Thu	2022/03
8201	01/14/2022	Fri	2022/03

Поле `week_number` создано как мера в объекте диаграммы с использованием функции `weekname()`, где в качестве аргумента функции передано поле даты.

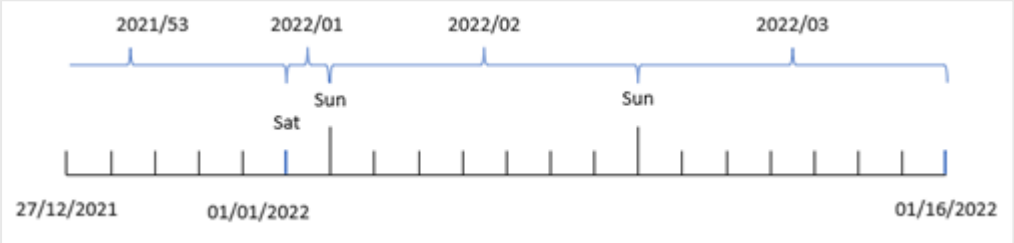
Функция `weekname()` изначально определяет, на какую неделю приходится значение даты, и возвращает номер недели и год, когда была совершена транзакция.

Системная переменная `FirstWeekDay` задает воскресенье как первый день недели. Системная переменная `BrokenWeeks` задает для приложения использование неполных недель, что означает, что неделя 1 начинается 1 января.

Диаграмма, которая показывает номер недели, где первым днем недели является воскресенье.



Диаграмма, которая показывает, что транзакция 8192 совершена на неделе 2.



Поскольку приложение использует неполные недели, а первый день недели – воскресенье, транзакции, совершенные со 2 по 8 января, возвращают значение 2022/02, то есть неделя 2 в 2022 году. Обратите внимание, что транзакция 8192 совершена 5 января и возвращает значение 2022/02 для поля `week_number`.

Пример 5. Сценарий

Скрипт загрузки и результаты

Обзор

Откройте редактор загрузки данных и добавьте приведенный ниже скрипт загрузки на новую вкладку.

Скрипт загрузки содержит следующее:

- Набор данных, содержащий набор транзакций за последнюю неделю 2019 года, загружается в таблицу под именем `Transactions`.
- Системная переменная `BrokenWeeks`, для которой задан формат `0`.
- Системная переменная `ReferenceDay`, для которой задан формат `2`.
- Системная переменная `DateFormat`, для которой задан формат `MM/DD/YYYY`.

Скрипт загрузки

```
SET BrokenWeeks=0;
SET ReferenceDay=2;
SET DateFormat='MM/DD/YYYY';

Transactions:
Load
```

```
*
Inline
[
id,date,amount
8183,12/27/2019,58.27
8184,12/28/2019,67.42
8185,12/29/2019,23.80
8186,12/30/2019,82.06
8187,12/31/2019,40.56
8188,01/01/2020,37.23
8189,01/02/2020,17.17
8190,01/03/2020,88.27
8191,01/04/2020,57.42
8192,01/05/2020,53.80
8193,01/06/2020,82.06
8194,01/07/2020,40.56
8195,01/08/2020,53.67
8196,01/09/2020,26.63
8197,01/10/2020,72.48
8198,01/11/2020,18.37
8199,01/12/2020,45.26
8200,01/13/2020,58.23
8201,01/14/2020,18.52
];
```

Результаты

Загрузите данные и откройте лист. Создайте новую таблицу.

Создайте вычисляемое измерение, используя следующее выражение:

```
=weekname(date)
```

Чтобы рассчитать общий объем продаж, создайте следующую меру агрегирования:

```
=sum(amount)
```

Задайте параметру **Формат чисел** меры значение **Денежный**.

Результирующая таблица

weekname(date)	=sum(amount)
2019/52	$125.69
2020/01	$346.51
2020/02	$347.57
2020/03	$122.01

Чтобы продемонстрировать результаты использования функции weekname() в этом сценарии, добавьте следующее поле в качестве измерения:

```
date
```

Таблица результатов с полем даты

weekname(date)	date	=sum(amount)
2019/52	12/27/2019	$58.27
2019/52	12/28/2019	$67.42
2020/01	12/29/2019	$23.80
2020/01	12/30/2019	$82.06
2020/01	12/31/2019	$40.56
2020/01	01/01/2020	$37.23
2020/01	01/02/2020	$17.17
2020/01	01/03/2020	$88.27
2020/01	01/04/2020	$57.42
2020/02	01/05/2020	$53.80
2020/02	01/06/2020	$82.06
2020/02	01/07/2020	$40.56
2020/02	01/08/2020	$53.67
2020/02	01/09/2020	$26.63
2020/02	01/10/2020	$72.48
2020/02	01/11/2020	$18.37
2020/03	01/12/2020	$45.26
2020/03	01/13/2020	$58.23
2020/03	01/14/2020	$18.52

Поскольку в приложении используются непрерывные недели, а для первой недели требуется как минимум два дня в январе из-за системной переменной `ReferenceDay`, неделя 1 2020 года включает транзакции с 29 декабря 2019 года.

weekstart

Эта функция возвращает значение, которое соответствует метке времени, равной первой миллисекунде первого дня календарной недели, и содержит **date**. По умолчанию для вывода используется формат **DateFormat**, заданный в скрипте.

Синтаксис:

```
WeekStart(date [, period_no[, first_week_day]])
```

Возвращаемые типы данных: двойное значение

Схема функции weekstart()

Функция `weekstart()` определяет, на какой день недели приходится дата. Затем она возвращает метку времени в формате даты для первой миллисекунды этой недели. Первый день недели определяется переменной среды `FirstWeekDay`. Однако ее можно переопределить с помощью аргумента `first_week_day` в функции `weekstart()`.

Аргументы

Аргумент	Описание
date	Дата или метка времени 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 – понедельник, 1 – вторник, 2 – среда, 3 – четверг, 4 – пятница, 5 – суббота и 6 – воскресенье. Для получения дополнительной информации о системной переменной см. *FirstWeekDay (page 225)*.

Когда это следует использовать

Функция `weekstart()` широко используется в составе выражения, когда пользователю требуется учитывать в расчетах часть недели, которая уже прошла. Например, с ее помощью можно вычислить совокупную заработную плату, заработанную сотрудниками за эту неделю по состоянию на текущий момент.

Примеры функции

Пример	Результат
`weekstart('01/12/2013')`	Возвращает 01/07/2013.
`weekstart('01/12/2013', -1 )`	Возвращает 11/31/2012.
`weekstart('01/12/2013', 0, 1)`	Возвращает 01/08/2013.

Региональные настройки

Если не указано иное, в примерах, приведенных в данном разделе, используется следующий формат даты: ММ/ДД/ГГГГ. Формат даты указан в операторе `SET DateFormat` скрипта загрузки данных. В вашей системе может быть установлен другой формат даты по умолчанию в зависимости от региональных настроек и других факторов. Можно изменить формат в примерах в соответствии с потребностями. Или можно изменить форматы в скрипте загрузки в соответствии с этими примерами.

Региональные настройки по умолчанию в приложениях основаны на системных региональных настройках компьютера или сервера, где установлено ПО Qlik Sense. Если на сервере Qlik Sense, к которому обращается пользователь, выбран шведский язык, то редактор загрузки данных будет использовать шведские региональные настройки для даты, времени и валюты. Эти параметры регионального формата не связаны с языком, отображаемым в интерфейсе пользователя Qlik Sense. Qlik Sense будет отображаться на языке, который используется в браузере.

Пример 1. Без дополнительных аргументов

Скрипт загрузки и результаты

Обзор

Откройте редактор загрузки данных и добавьте приведенный ниже скрипт загрузки на новую вкладку.

Скрипт загрузки содержит следующее:

- Набор данных, содержащий набор транзакций за 2022 год, загружается в таблицу под именем `Transactions`.
- Поле даты было предоставлено в формате системной переменной `DateFormat` (MM/DD/YYYY).
- Создание поля `start_of_week`, возвращающего метку времени начала недели, в течение которой совершены транзакции.

Скрипт загрузки

```
SET FirstWeekDay=6;

Transactions:
    Load
        *,
        weekstart(date) as start_of_week,
        timestamp(weekstart(date)) as start_of_week_timestamp
        ;
Load
*
Inline
[
id,date,amount
8188,1/7/2022,17.17
```

```
8189,1/19/2022,37.23
8190,2/28/2022,88.27
8191,2/5/2022,57.42
8192,3/16/2022,53.80
8193,4/1/2022,82.06
8194,5/7/2022,40.39
8195,5/16/2022,87.21
8196,6/15/2022,95.93
8197,6/26/2022,45.89
8198,7/9/2022,36.23
8199,7/22/2022,25.66
8200,7/23/2022,82.77
8201,7/27/2022,69.98
8202,8/2/2022,76.11
8203,8/8/2022,25.12
8204,8/19/2022,46.23
8205,9/26/2022,84.21
8206,10/14/2022,96.24
8207,10/29/2022,67.67
];
```

Результаты

Загрузите данные и откройте лист. Создайте новую таблицу и добавьте эти поля как измерения:

- `date`
- `start_of_week`
- `start_of_week_timestamp`

Результирующая таблица

date	start_of_week	start_of_week_timestamp
1/7/2022	01/02/2022	1/2/2022 12:00:00 AM
1/19/2022	01/16/2022	1/16/2022 12:00:00 AM
2/5/2022	01/30/2022	1/30/2022 12:00:00 AM
2/28/2022	02/27/2022	2/27/2022 12:00:00 AM
3/16/2022	03/13/2022	3/13/2022 12:00:00 AM
4/1/2022	03/27/2022	3/27/2022 12:00:00 AM
5/7/2022	05/01/2022	5/1/2022 12:00:00 AM
5/16/2022	05/15/2022	5/15/2022 12:00:00 AM
6/15/2022	06/12/2022	6/12/2022 12:00:00 AM
6/26/2022	06/26/2022	6/26/2022 12:00:00 AM
7/9/2022	07/03/2022	7/3/2022 12:00:00 AM
7/22/2022	07/17/2022	7/17/2022 12:00:00 AM

date	start_of_week	start_of_week_timestamp
7/23/2022	07/17/2022	7/17/2022 12:00:00 AM
7/27/2022	07/24/2022	7/24/2022 12:00:00 AM
8/2/2022	07/31/2022	7/31/2022 12:00:00 AM
8/8/2022	08/07/2022	8/7/2022 12:00:00 AM
8/19/2022	08/14/2022	8/14/2022 12:00:00 AM
9/26/2022	09/25/2022	9/25/2022 12:00:00 AM
10/14/2022	10/09/2022	10/9/2022 12:00:00 AM
10/29/2022	10/23/2022	10/23/2022 12:00:00 AM

Поле `start_of_week` создано предшествующим оператором load с использованием функции `weekstart()`, где в качестве аргумента функции передано поле даты.

Функция `weekstart()` определяет, к какой неделе относится значение даты, и возвращает метку времени для первой миллисекунды этой недели.

Диаграмма функции `weekstart()`, пример без дополнительных аргументов

Транзакция 8191 совершена 5 февраля. Системная переменная `FirstWeekDay` задает в качестве первого дня недели воскресенье. Функция `weekstart()` определяет, что первой субботой перед 5 февраля и, следовательно, первым днем недели, является 30 января. Таким образом, значение `start_of_week` для этой транзакции возвращает первую миллисекунду этого дня, то есть 00:00:00 (12:00:00 AM) 30 января.

Пример 2. Скрипт period_no

Скрипт загрузки и результаты

Обзор

Откройте редактор загрузки данных и добавьте приведенный ниже скрипт загрузки на новую вкладку.

Скрипт загрузки содержит следующее:

- Используется тот же набор данных и сценарий, что в первом примере.
- Создание поля `previous_week_start`, возвращающего метку времени начала квартала, который предшествует совершению транзакции.

Скрипт загрузки

```
SET DateFormat='MM/DD/YYYY';

Transactions:
    Load
        *,
        weekstart(date,-1) as previous_week_start,
        timestamp(weekstart(date,-1)) as previous_week_start_timestamp
        ;
Load
*
Inline
[
id,date,amount
8188,1/7/2022,17.17
8189,1/19/2022,37.23
8190,2/28/2022,88.27
8191,2/5/2022,57.42
8192,3/16/2022,53.80
8193,4/1/2022,82.06
8194,5/7/2022,40.39
8195,5/16/2022,87.21
8196,6/15/2022,95.93
8197,6/26/2022,45.89
8198,7/9/2022,36.23
8199,7/22/2022,25.66
8200,7/23/2022,82.77
8201,7/27/2022,69.98
8202,8/2/2022,76.11
8203,8/8/2022,25.12
8204,8/19/2022,46.23
8205,9/26/2022,84.21
8206,10/14/2022,96.24
8207,10/29/2022,67.67
];
```

Результаты

Загрузите данные и откройте лист. Создайте новую таблицу и добавьте эти поля как измерения:

- `date`
- `previous_week_start`
- `previous_week_start_timestamp`

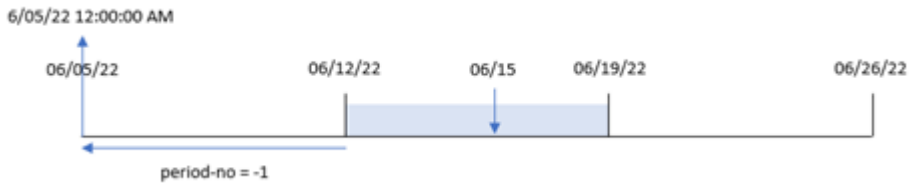
Результирующая таблица

date	previous_week_start	previous_week_start_timestamp
1/7/2022	12/26/2021	12/26/2021 12:00:00 AM
1/19/2022	01/09/2022	1/9/2022 12:00:00 AM

date	previous_week_start	previous_week_start_timestamp
2/5/2022	01/23/2022	1/23/2022 12:00:00 AM
2/28/2022	02/20/2022	2/20/2022 12:00:00 AM
3/16/2022	03/06/2022	3/6/2022 12:00:00 AM
4/1/2022	03/20/2022	3/20/2022 12:00:00 AM
5/7/2022	04/24/2022	4/24/2022 12:00:00 AM
5/16/2022	05/08/2022	5/8/2022 12:00:00 AM
6/15/2022	06/05/2022	6/5/2022 12:00:00 AM
6/26/2022	06/19/2022	6/19/2022 12:00:00 AM
7/9/2022	06/26/2022	6/26/2022 12:00:00 AM
7/22/2022	07/10/2022	7/10/2022 12:00:00 AM
7/23/2022	07/10/2022	7/10/2022 12:00:00 AM
7/27/2022	07/17/2022	7/17/2022 12:00:00 AM
8/2/2022	07/24/2022	7/24/2022 12:00:00 AM
8/8/2022	07/31/2022	7/31/2022 12:00:00 AM
8/19/2022	08/07/2022	8/7/2022 12:00:00 AM
9/26/2022	09/18/2022	9/18/2022 12:00:00 AM
10/14/2022	10/02/2022	10/2/2022 12:00:00 AM
10/29/2022	10/16/2022	10/16/2022 12:00:00 AM

В этом случае, так как в качестве аргумента смещения в функции `weekstart()` использовалось `period_no` = -1, функция сначала определяет неделю, в течение которой совершаются транзакции. Затем она возврващается на неделю назад и определяет первую миллисекунду этой недели.

Диаграмма функции `weekstart()`, пример с аргументом period_no



Транзакция 8196 совершена 15 июня. Функция `weekstart()` определяет, что неделя начинается 12 июня. Таким образом, предыдущая неделя началась 5 июня в 00:00:00 (12:00:00 AM). Именно это значение возвращается для поля `previous_week_start`.

Пример 3. Аргумент first_week_day

Скрипт загрузки и результаты

Обзор

Откройте редактор загрузки данных и добавьте приведенный ниже скрипт загрузки на новую вкладку.

Скрипт загрузки содержит тот же набор данных и сценарий, что в первом примере. Однако в этом примере требуется задать вторник в качестве первого дня недели.

Скрипт загрузки

```
SET DateFormat='MM/DD/YYYY';

Transactions:
    Load
        *,
        weekstart(date,0,1) as start_of_week,
        timestamp(weekstart(date,0,1)) as start_of_week_timestamp
        ;
Load
*
Inline
[
id,date,amount
8188,1/7/2022,17.17
8189,1/19/2022,37.23
8190,2/28/2022,88.27
8191,2/5/2022,57.42
8192,3/16/2022,53.80
8193,4/1/2022,82.06
8194,5/7/2022,40.39
8195,5/16/2022,87.21
8196,6/15/2022,95.93
8197,6/26/2022,45.89
8198,7/9/2022,36.23
8199,7/22/2022,25.66
8200,7/23/2022,82.77
8201,7/27/2022,69.98
8202,8/2/2022,76.11
8203,8/8/2022,25.12
8204,8/19/2022,46.23
8205,9/26/2022,84.21
8206,10/14/2022,96.24
8207,10/29/2022,67.67
];
```

Результаты

Загрузите данные и откройте лист. Создайте новую таблицу и добавьте эти поля как измерения:

- date
- start_of_week
- start_of_week_timestamp

Результирующая таблица

date	start_of_week	start_of_week_timestamp
1/7/2022	01/04/2022	1/4/2022 12:00:00 AM
1/19/2022	01/18/2022	1/18/2022 12:00:00 AM
2/5/2022	02/01/2022	2/1/2022 12:00:00 AM
2/28/2022	02/22/2022	2/22/2022 12:00:00 AM
3/16/2022	03/15/2022	3/15/2022 12:00:00 AM
4/1/2022	03/29/2022	3/29/2022 12:00:00 AM
5/7/2022	05/03/2022	5/3/2022 12:00:00 AM
5/16/2022	05/10/2022	5/10/2022 12:00:00 AM
6/15/2022	06/14/2022	6/14/2022 12:00:00 AM
6/26/2022	06/21/2022	6/21/2022 12:00:00 AM
7/9/2022	07/05/2022	7/5/2022 12:00:00 AM
7/22/2022	07/19/2022	7/19/2022 12:00:00 AM
7/23/2022	07/19/2022	7/19/2022 12:00:00 AM
7/27/2022	07/26/2022	7/26/2022 12:00:00 AM
8/2/2022	08/02/2022	8/2/2022 12:00:00 AM
8/8/2022	08/02/2022	8/2/2022 12:00:00 AM
8/19/2022	08/16/2022	8/16/2022 12:00:00 AM
9/26/2022	09/20/2022	9/20/2022 12:00:00 AM
10/14/2022	10/11/2022	10/11/2022 12:00:00 AM
10/29/2022	10/25/2022	10/25/2022 12:00:00 AM

В этом случае, поскольку аргумент `first_week_date` =1 используется в функции `weekstart()`, в качестве первого дня недели задан вторник.

Диаграмма функции `weekstart()` , пример с аргументом first_week_day

Транзакция 8191 совершена 5 февраля. Функция `weekstart()` определяет, что первым вторником перед этой датой и, следовательно, первым днем недели является 1 февраля, и возвращается значение 00:00:00 (12:00:00 AM) 6 февраля.

Пример 4. Пример объекта диаграммы

Скрипт загрузки и выражение диаграммы

Обзор

Откройте редактор загрузки данных и добавьте приведенный ниже скрипт загрузки на новую вкладку.

Скрипт загрузки содержит тот же набор данных и сценарий, что в первом примере.

Однако в этом примере в приложение загружается неизмененный набор данных. Расчет, возвращающий метку времени начала недели, в течение которой совершены транзакции, создается как мера в объекте диаграммы в приложении.

Скрипт загрузки

```
Transactions:

Load
*
Inline
[
id,date,amount
8188,1/7/2022,17.17
8189,1/19/2022,37.23
8190,2/28/2022,88.27
8191,2/5/2022,57.42
8192,3/16/2022,53.80
8193,4/1/2022,82.06
8194,5/7/2022,40.39
8195,5/16/2022,87.21
8196,6/15/2022,95.93
8197,6/26/2022,45.89
8198,7/9/2022,36.23
8199,7/22/2022,25.66
8200,7/23/2022,82.77
8201,7/27/2022,69.98
8202,8/2/2022,76.11
8203,8/8/2022,25.12
8204,8/19/2022,46.23
8205,9/26/2022,84.21
8206,10/14/2022,96.24
8207,10/29/2022,67.67
];
```

Результаты

Загрузите данные и откройте лист. Создайте новую таблицу и добавьте это поле как измерение: `date`.

Чтобы рассчитать начало недели, когда совершена транзакция, добавьте следующие меры:

- `=weekstart(date)`
- `=timestamp(weekstart(date))`

Результирующая таблица

date	start_of_week	start_of_week_timestamp
1/7/2022	01/02/2022	1/2/2022 12:00:00 AM
1/19/2022	01/16/2022	1/16/2022 12:00:00 AM
2/5/2022	01/30/2022	1/30/2022 12:00:00 AM
2/28/2022	02/27/2022	2/27/2022 12:00:00 AM
3/16/2022	03/13/2022	3/13/2022 12:00:00 AM
4/1/2022	03/27/2022	3/27/2022 12:00:00 AM
5/7/2022	05/01/2022	5/1/2022 12:00:00 AM
5/16/2022	05/15/2022	5/15/2022 12:00:00 AM
6/15/2022	06/12/2022	6/12/2022 12:00:00 AM
6/26/2022	06/26/2022	6/26/2022 12:00:00 AM
7/9/2022	07/03/2022	7/3/2022 12:00:00 AM
7/22/2022	07/17/2022	7/17/2022 12:00:00 AM
7/23/2022	07/17/2022	7/17/2022 12:00:00 AM
7/27/2022	07/24/2022	7/24/2022 12:00:00 AM
8/2/2022	07/31/2022	7/31/2022 12:00:00 AM
8/8/2022	08/07/2022	8/7/2022 12:00:00 AM
8/19/2022	08/14/2022	8/14/2022 12:00:00 AM
9/26/2022	09/25/2022	9/25/2022 12:00:00 AM
10/14/2022	10/09/2022	10/9/2022 12:00:00 AM
10/29/2022	10/23/2022	10/23/2022 12:00:00 AM

Мера `start_of_week` создана в объекте диаграммы с использованием функции `weekstart()`, где в качестве аргумента функции передано поле `date`.

Функция `weekstart()` определяет, к какой неделе относится значение даты, и возвращает метку времени для первой миллисекунды этой недели.

Диаграмма функции weekstart(), пример с объектом диаграммы

Транзакция 8191 совершена 5 февраля. Системная переменная `FirstWeekDay` задает в качестве первого дня недели воскресенье. Функция `weekstart()` определяет, что первым воскресеньем перед 5 февраля и, следовательно, первым днем недели является 30 января. Поэтому значение `start_of_week` для этой транзакции возвращает первую миллисекунду этого дня, то есть 00:00:00 (12:00:00 AM) 30 января.

Пример 5. Сценарий

Скрипт загрузки и выражение диаграммы

Обзор

Откройте редактор загрузки данных и добавьте приведенный ниже скрипт загрузки на новую вкладку.

Скрипт загрузки содержит следующее:

- Набор данных, загруженный в таблицу, называется `Payroll`
- Данные, включающие идентификаторы сотрудников, имена сотрудников и среднюю дневную заработную плату каждого сотрудника.

Сотрудники приступают к работе в понедельник и работают 6 дней в неделю. Системную переменную `FirstWeekDay` изменять нельзя.

Конечному пользователю требуется объект диаграммы, который по идентификатору и имени сотрудника отображает заработную плату, заработанную в течение недели до текущей даты.

Скрипт загрузки

```
Payroll:
Load
*
Inline
[
employee_id,employee_name,day_rate
182,Mark, $150
183,Deryck, $125
184,Dexter, $125
185,Sydney,$270
186,Agatha,$128
];
```

Результаты

Выполните следующие действия.

1. Загрузите данные и откройте лист. Создайте новую таблицу и добавьте эти поля как измерения:
   - `employee_id`
   - `employee_name`
2. Затем создайте меру для расчета заработной платы, заработанной в течение недели до текущей даты:
   `=if(today(1)-weekstart(today(1),0,0)<7,(today(1)-weekstart(today(1),0,0))*day_rate,day_rate*6)`
3. Задайте параметру меры **Формат чисел** значение **Денежный**.

Результирующая таблица

employee_id	employee_name	=if(today(1)-weekstart(today(1),0,0)<7,(today(1)-weekstart(today(1),0,0))*day_rate,day_rate*6)
182	Mark	$600.00
183	Deryck	$500.00
184	Dexter	$500.00
185	Sydney	$1080.00
186	Agatha	$512.00

Функция `weekstart()`, используя сегодняшнюю дату в качестве первого аргумента и 0 в качестве третьего аргумента, задает понедельник в качестве первого дня недели и возвращает дату начала текущей недели. Вычитая этот результат из текущей даты, выражение затем возвращает количество дней, прошедших до сих пор в течение этой недели.

Затем условие снова проверяет, прошло ли больше 6 дней на этой неделе. Если да, то `day_rate` сотрудника умножается на 6 дней. В противном случае `day_rate` умножается на количество дней этой недели, которые уже прошли.

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:

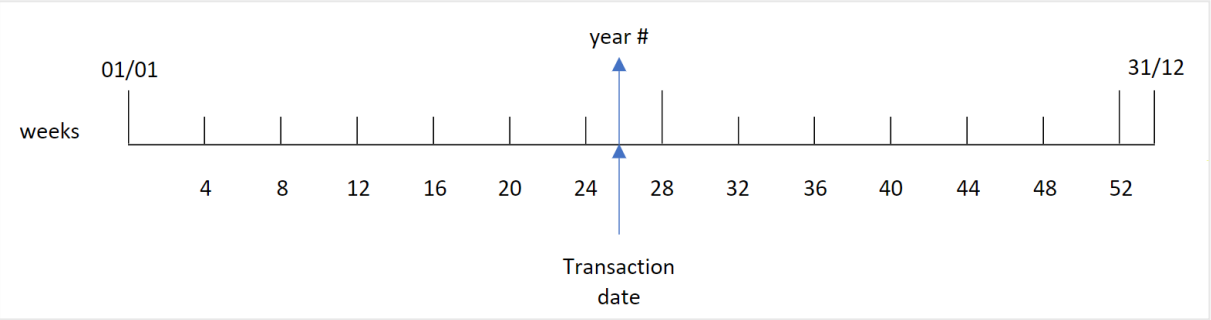
```
weekyear(expression)
```

Возвращаемые типы данных: целое

Функция `weekyear()` определяет, к какой неделе года относится дата. Затем она возвращает год, соответствующий этому номеру недели.

В приложениях Qlik по умолчанию используются неполные недели (это определяет системная переменная `BrokenWeeks`), и неделя 1 начинается 1 января, а год заканчивается после недели 52. Таким образом, функция `weekyear()` всегда возвращает то же значение, что функция `week()`, когда приложение использует неполные недели.

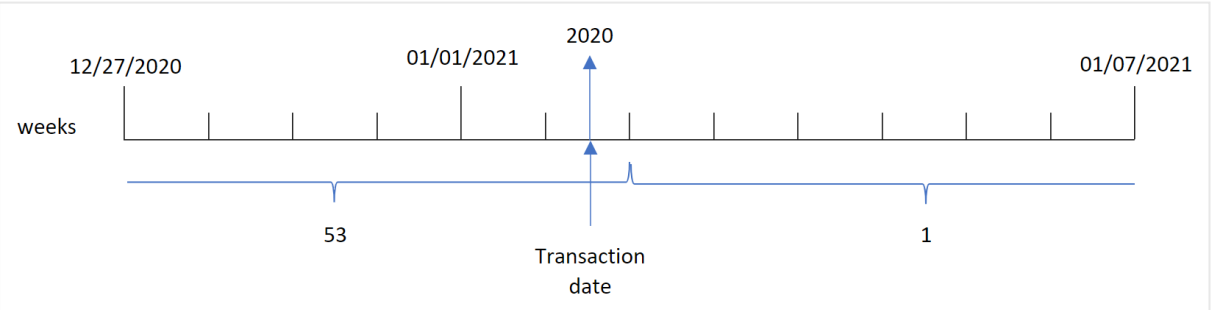
Диаграмма диапазона функции `weekyear()`



Однако если системная переменная `BrokenWeeks` задает использование полных недель, неделя 1 должна содержать не меньше определенного количества дней января на основе значения, заданного системной переменной `ReferenceDay`.

Например, если используется `ReferenceDay = 4`, неделя 1 должна включать не меньше четырех дней января. Неделя 1 может включать дни декабря предыдущего года, а последняя неделя года может включать дни января следующего года. В подобных ситуациях функция `weekyear()` возвращает значение, не совпадающее с результатом функции `year()`.

Диаграмма диапазона функции `weekyear()` при использовании полных недель



Когда это следует использовать

Функция `weekyear()` полезна, когда требуется сравнить агрегирования по годам, например, если требуется увидеть общий объем продаж продуктов по годам. Функции `weekyear()` отдается предпочтение перед `year()`, когда пользователю требуется обеспечить согласованность с переменной `BrokenWeeks` в приложении.

Региональные настройки

Если не указано иное, в примерах, приведенных в данном разделе, используется следующий формат даты: ММ/ДД/ГГГГ. Формат даты указан в операторе `SET DateFormat` скрипта загрузки данных. В вашей системе может быть установлен другой формат даты по умолчанию в зависимости от региональных настроек и других факторов. Можно изменить формат в примерах в соответствии с потребностями. Или можно изменить форматы в скрипте загрузки в соответствии с этими примерами.

Региональные настройки по умолчанию в приложениях основаны на системных региональных настройках компьютера или сервера, где установлено ПО Qlik Sense. Если на сервере Qlik Sense, к которому обращается пользователь, выбран шведский язык, то редактор загрузки данных будет использовать шведские региональные настройки для даты, времени и валюты. Эти параметры регионального формата не связаны с языком, отображаемым в интерфейсе пользователя Qlik Sense. Qlik Sense будет отображаться на языке, который используется в браузере.

Примеры функции

Пример	**Результат**
`weekyear ('12/30/1996')`	Возвращает 1997, поскольку неделя 1 1997 года начинается 12/30/1996
`weekyear ('01/02/1997')`	Возвращает 1997
`weekyear ('12/28/1997')`	Возвращает 1997
`weekyear ('12/30/1997')`	Возвращает 1998, поскольку неделя 1 1998 года начинается 12/29/1997
`weekyear ('01/02/1999')`	Возвращает 1998, поскольку неделя 53 1998 года оканчивается 01/03/1999

Связанные темы

Тема	**Взаимодействие**
week (page 1070)	Возвращает номер недели в виде целого числа согласно стандарту ISO 8601
year (page 1145)	Возвращает год в виде целого числа, а выражение интерпретируется как дата согласно стандартной интерпретации чисел.

Пример 1. Неполные недели

Скрипт загрузки и результаты

Обзор

Откройте редактор загрузки данных и добавьте приведенный ниже скрипт загрузки на новую вкладку.

Скрипт загрузки содержит следующее:

- Набор данных, содержащий набор транзакций за последнюю неделю 2020 года и первую неделю 2021, загружается в таблицу под именем `Transactions`.
- Переменная `BrokenWeeks`, для которой задано значение 1.
- Предыдущая загрузка, которая содержит следующее:
  - Функция `weekyear()`, заданная как поле `week_year` и возвращающая год, в течение которого совершены транзакции.
  - Функция `week()`, заданная как поле `week`, которая показывает день недели для каждой даты транзакции.

Скрипт загрузки

```
SET BrokenWeeks=1;

Transactions:
       Load
       *,
       week(date) as week,
       weekyear(date) as week_year
       ;
Load
*
Inline
[
id,date,amount
8176,12/28/2020,19.42
8177,12/29/2020,23.80
8178,12/30/2020,82.06
8179,12/31/2020,40.56
8180,01/01/2021,37.23
8181,01/02/2021,17.17
8182,01/03/2021,88.27
8183,01/04/2021,57.42
8184,01/05/2021,67.42
8185,01/06/2021,23.80
8186,01/07/2021,82.06
];
```

Результаты

Загрузите данные и откройте лист. Создайте новую таблицу и добавьте эти поля как измерения:

- id
- date
- week
- week_year

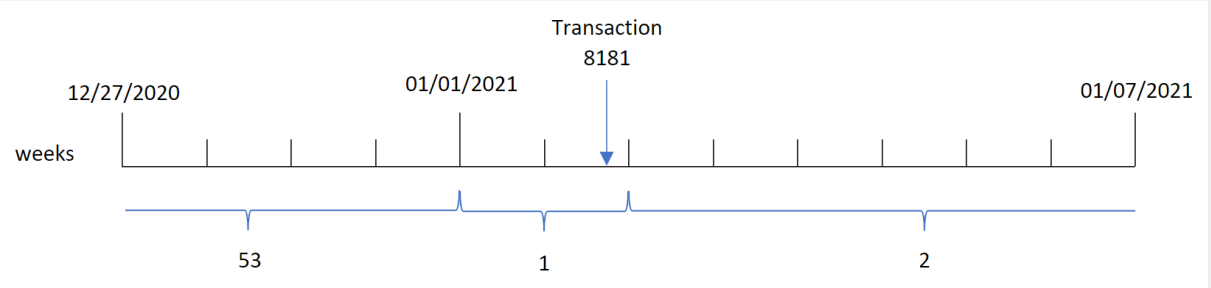
Результирующая таблица

id	date	week	week_year
8176	12/28/2020	53	2020
8177	12/29/2020	53	2020
8178	12/30/2020	53	2020
8179	12/31/2020	53	2020
8180	01/01/2021	1	2021
8181	01/02/2021	1	2021
8182	01/03/2021	2	2021
8183	01/04/2021	2	2021
8184	01/05/2021	2	2021
8185	01/06/2021	2	2021
8186	01/07/2021	2	2021

Поле week_year создано предшествующим оператором load с использованием функции weekyear(), где в качестве аргумента функции передано поле даты.

Системной переменной BrokenWeeks задано значение 1, то есть приложение использует неполные недели. Неделя 1 начинается 1 января.

Диаграмма диапазона функции weekyear() при использовании неполных недель



Транзакция 8181 совершена 2 января, то есть в течение недели 1. Поэтому функция возвращает значение 2021 для поля week_year.

Пример 2. Полные недели

Скрипт загрузки и результаты

Обзор

Откройте редактор загрузки данных и добавьте приведенный ниже скрипт загрузки на новую вкладку.

Скрипт загрузки содержит следующее:

- Набор данных, содержащий набор транзакций за последнюю неделю 2020 года и первую неделю 2021, загружается в таблицу под именем `Transactions`.
- Переменная `BrokenWeeks`, для которой задано значение `0`.
- Предыдущая загрузка, которая содержит следующее:
  - Функция `weekyear()`, заданная как поле `week_year` и возвращающая год, в течение которого совершены транзакции.
  - Функция `week()`, заданная как поле `week` и отображающая день недели для каждой даты транзакции.

Однако в этом примере политика компании требует использования полных недель.

Скрипт загрузки

```
SET BrokenWeeks=0;

Transactions:
       Load
       *,
       week(date) as week,
       weekyear(date) as week_year
       ;
Load
*
Inline
[
id,date,amount
8176,12/28/2020,19.42
8177,12/29/2020,23.80
8178,12/30/2020,82.06
8179,12/31/2020,40.56
8180,01/01/2021,37.23
8181,01/02/2021,17.17
8182,01/03/2021,88.27
8183,01/04/2021,57.42
8184,01/05/2021,67.42
8185,01/06/2021,23.80
8186,01/07/2021,82.06
];
```

Результаты

Загрузите данные и откройте лист. Создайте новую таблицу и добавьте эти поля как измерения:

- `id`
- `date`
- `week`
- `week_year`

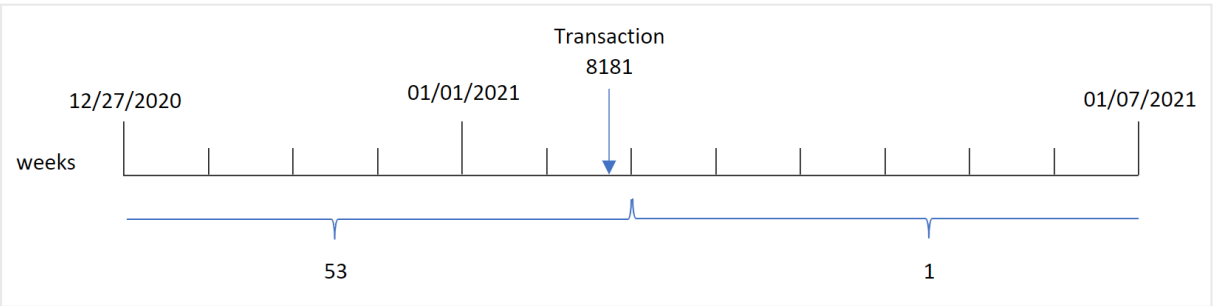
Результирующая таблица

id	date	week	week_year
8176	12/28/2020	53	2020
8177	12/29/2020	53	2020
8178	12/30/2020	53	2020
8179	12/31/2020	53	2020
8180	01/01/2021	53	2020
8181	01/02/2021	53	2020
8182	01/03/2021	1	2021
8183	01/04/2021	1	2021
8184	01/05/2021	1	2021
8185	01/06/2021	1	2021
8186	01/07/2021	1	2021

Системной переменной `BrokenWeeks` задано значение `0`, то есть приложение использует полные недели. Таким образом, неделя 1 не должна начинаться 1 января.

Неделя 53 2020 года продолжается до конца 2 января 2021, а неделя 1 2021 года начинается в воскресение, 3 января 2021 года.

Диаграмма диапазона функции `weekyear()` при использовании полных недель



Транзакция 8181 совершена 2 января, то есть в течение недели 1. Поэтому функция возвращает значение 2021 для поля `week_year`.

Пример 3. Пример объекта диаграммы

Скрипт загрузки и выражение диаграммы

Обзор

Используется тот же набор данных и сценарий, что в первом примере.

Однако в этом примере в приложение загружается неизмененный набор данных. Расчет, возвращающий номер недели года, в течение которой совершена транзакция, создается как мера в объекте диаграммы в приложении.

Скрипт загрузки

```
SET BrokenWeeks=1;

Transactions:
Load
*
Inline
[
id,date,amount
8176,12/28/2020,19.42
8177,12/29/2020,23.80
8178,12/30/2020,82.06
8179,12/31/2020,40.56
8180,01/01/2021,37.23
8181,01/02/2021,17.17
8182,01/03/2021,88.27
8183,01/04/2021,57.42
8184,01/05/2021,67.42
8185,01/06/2021,23.80
8186,01/07/2021,82.06
];
```

Результаты

Загрузите данные и откройте лист. Создайте новую таблицу и добавьте эти поля как измерения:

- `id`
- `date`

Чтобы рассчитать неделю, когда была совершена транзакция, создайте следующую меру:

- `=week(date)`

Чтобы по номеру недели рассчитать год, когда была совершена транзакция, создайте следующую меру:

- `=weekyear(date)`

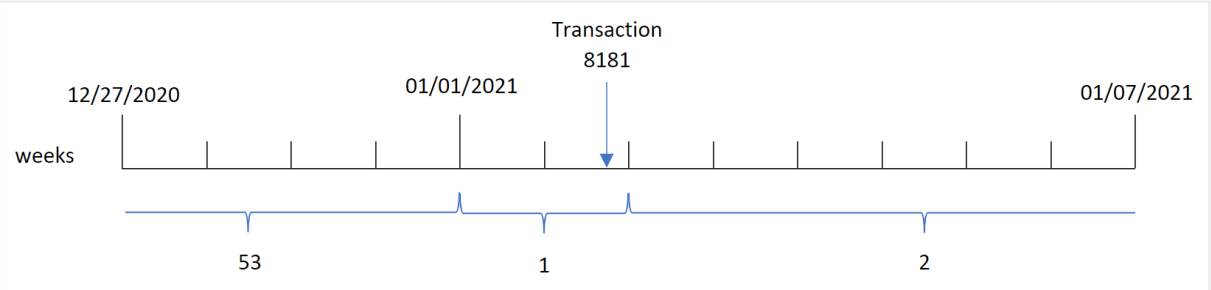
Результирующая таблица

id	date	week	week_year
8176	12/28/2020	53	2020
8177	12/29/2020	53	2020
8178	12/30/2020	53	2020
8179	12/31/2020	53	2020
8180	01/01/2021	1	2021
8181	01/02/2021	1	2021
8182	01/03/2021	2	2021
8183	01/04/2021	2	2021
8184	01/05/2021	2	2021
8185	01/06/2021	2	2021
8186	01/07/2021	2	2021

Поле `week_year` создано предшествующим оператором load с использованием функции `weekyear()`, где в качестве аргумента функции передано поле даты.

Системной переменной `BrokenWeeks` задано значение `1`, то есть приложение использует неполные недели. Неделя 1 начинается 1 января.

Диаграмма диапазона функции `weekyear()` при использовании неполных недель



Транзакция 8181 совершена 2 января, то есть в течение недели 1. Поэтому функция возвращает значение 2021 для поля `week_year`.

Пример 4. Сценарий

Скрипт загрузки и результаты

Обзор

Откройте редактор загрузки данных и добавьте приведенный ниже скрипт загрузки на новую вкладку.

Скрипт загрузки содержит следующее:

- Набор данных, содержащий набор транзакций за последнюю неделю 2020 года и первую неделю 2021, загружается в таблицу под именем `Transactions`.
- Переменная `BrokenWeeks`, для которой задано значение `0`. Это означает, что приложение будет использовать полные недели.
- Переменная `ReferenceDay`, для которой задано значение `2`. Это означает, что год будет начинаться 2 января и первая неделя будет содержать не меньше двух дней января.
- Переменная `FirstWeekDay`, для которой задано значение `1`. Это значит, что первым днем недели будет вторник.

Политика компании требует использования неполных недель. Конечному пользователю нужна диаграмма, на которой представлены общие продажи по годам. В приложении используются полные недели, и неделя 1 содержит не меньше двух дней января.

Скрипт загрузки

```
SET BrokenWeeks=0;
SET ReferenceDay=2;
SET FirstWeekDay=1;

Transactions:
Load
*
Inline
[
id,date,amount
8176,12/28/2020,19.42
8177,12/29/2020,23.80
8178,12/30/2020,82.06
8179,12/31/2020,40.56
8180,01/01/2021,37.23
8181,01/02/2021,17.17
8182,01/03/2021,88.27
8183,01/04/2021,57.42
8184,01/05/2021,67.42
8185,01/06/2021,23.80
8186,01/07/2021,82.06
];
```

Результаты

Загрузите данные и откройте лист. Создайте новую таблицу.

Чтобы по номеру недели рассчитать год, когда была совершена транзакция, создайте следующую меру:

- `=weekyear(date)`

Чтобы рассчитать общий объем продаж, создайте следующую меру:

- `sum(amount)`

Задайте параметру **Формат чисел** меры значение **Денежный**.

Результирующая таблица

weekyear(date)	=sum(amount)
2020	19.42
2021	373.37

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

Функция `year()` доступна как функция скрипта и как функция диаграммы. Функция возвращает год для определенной даты. Она широко используется с целью создания поля года в качестве измерения в основном календаре.

Когда это следует использовать

Функция `year()` полезна, когда требуется сравнить агрегирования по годам. Например, ее можно использовать, если требуется увидеть общий объем продаж продуктов по годам.

Эти измерения можно создать в скрипте загрузки с помощью функции создания поля в таблице основного календаря. Ее также можно использовать непосредственно в диаграмме в качестве вычисляемого измерения.

Примеры функции

Пример	Результат
`year( '2012-10-12' )`	возвращает 2012
`year( '35648' )`	возвращает 1997, так как 35648 = 1997-08-06

Региональные настройки

Если не указано иное, в примерах, приведенных в данном разделе, используется следующий формат даты: ММ/ДД/ГГГГ. Формат даты указан в операторе `SET DateFormat` скрипта загрузки данных. В вашей системе может быть установлен другой формат даты по умолчанию в зависимости от региональных настроек и других факторов. Можно изменить формат в примерах в соответствии с потребностями. Или можно изменить форматы в скрипте загрузки в соответствии с этими примерами.

Региональные настройки по умолчанию в приложениях основаны на системных региональных настройках компьютера или сервера, где установлено ПО Qlik Sense. Если на сервере Qlik Sense, к которому обращается пользователь, выбран шведский язык, то редактор загрузки данных будет

использовать шведские региональные настройки для даты, времени и валюты. Эти параметры регионального формата не связаны с языком, отображаемым в интерфейсе пользователя Qlik Sense. Qlik Sense будет отображаться на языке, который используется в браузере.

Пример 1. Набор данных DateFormat (скрипт)

Скрипт загрузки и результаты

Обзор

Откройте редактор загрузки данных и добавьте приведенный ниже скрипт загрузки на новую вкладку.

Скрипт загрузки содержит следующее:

- Набор данных с датами, загруженный в таблицу под именем `Master Calendar`.
- Используется системная переменная `DateFormat` со значением по умолчанию MM/DD/YYYY.
- Предшествующая загрузка, используемая для создания дополнительного поля под именем `year` с помощью функции `year()`.

Скрипт загрузки

```
SET DateFormat='MM/DD/YYYY';

Master_Calendar:

    Load
        date,
        year(date) as year
        ;
Load
date
Inline
[
date
12/28/2020
12/29/2020
12/30/2020
12/31/2020
01/01/2021
01/02/2021
01/03/2021
01/04/2021
01/05/2021
01/06/2021
01/07/2021
];
```

Результаты

Загрузите данные и откройте лист. Создайте новую таблицу и добавьте эти поля как измерения:

- date
- year

Результирующая таблица

date	год
12/28/2020	2020
12/29/2020	2020
12/30/2020	2020
12/31/2020	2020
01/01/2021	2021
01/02/2021	2021
01/03/2021	2021
01/04/2021	2021
01/05/2021	2021
01/06/2021	2021
01/07/2021	2021

Пример 2. Даты ANSI

Скрипт загрузки и результаты

Обзор

Откройте редактор загрузки данных и добавьте приведенный ниже скрипт загрузки на новую вкладку.

Скрипт загрузки содержит следующее:

- Набор данных с датами, загруженный в таблицу под именем `Master Calendar`.
- Используется системная переменная `DateFormat` со значением по умолчанию MM/DD/YYYY. Однако даты, включенные в набор данных, имеют формат стандарта ANSI.
- Предшествующая загрузка, используемая для создания дополнительного поля под именем `year` с помощью функции `year()`.

Скрипт загрузки

```
SET DateFormat='MM/DD/YYYY';

Master_Calendar:
    Load
        date,
        year(date) as year
```

```
        ;
Load
date
Inline
[
date
2020-12-28
2020-12-29
2020-12-30
2020-12-31
2021-01-01
2021-01-02
2021-01-03
2021-01-04
2021-01-05
2021-01-06
2021-01-07
];
```

Результаты

Загрузите данные и откройте лист. Создайте новую таблицу и добавьте эти поля как измерения:

- date
- year

Результирующая таблица

date	год
2020-12-28	2020
2020-12-29	2020
2020-12-30	2020
2020-12-31	2020
2021-01-01	2021
2021-01-02	2021
2021-01-03	2021
2021-01-04	2021
2021-01-05	2021
2021-01-06	2021
2021-01-07	2021

Пример 3. Неформатированные даты

Скрипт загрузки и результаты

Обзор

Откройте редактор загрузки данных и добавьте приведенный ниже скрипт загрузки на новую вкладку.

Скрипт загрузки содержит следующее:

- Набор данных с датами в числовом формате, загруженный в таблицу под именем `Master Calendar`.
- Используется системная переменная `DateFormat` со значением по умолчанию MM/DD/YYYY.
- Предшествующая загрузка, используемая для создания дополнительного поля под именем `year` с помощью функции `year()`.

Загружается первоначальная неотформатированная дата, `unformatted_date`, и для уточнения используется дополнительное поле, `long_date`, с целью преобразования числовой даты в поле форматированной даты с помощью функции `date()`.

Скрипт загрузки

```
SET DateFormat='MM/DD/YYYY';

Master_Calendar:
    Load
        unformatted_date,
        date(unformatted_date) as long_date,
        year(unformatted_date) as year
        ;
Load
unformatted_date
Inline
[
unformatted_date
44868
44898
44928
44958
44988
45018
45048
45078
45008
45038
45068
];
```

Результаты

Загрузите данные и откройте лист. Создайте новую таблицу и добавьте эти поля как измерения:

- `unformatted_date`
- `long_date`
- `year`

Результирующая таблица

unformatted_date	long_date	год
44868	11/03/2022	2022
44898	12/03/2022	2022
44928	01/02/2023	2023
44958	02/01/2023	2023
44988	03/03/2023	2023
45008	03/23/2023	2023
45018	04/02/2023	2023
45038	04/22/2023	2023
45048	05/02/2023	2023
45068	05/22/2023	2023
45078	06/01/2023	2023

Пример 4. Пример объекта диаграммы

Скрипт загрузки и выражение диаграммы

Обзор

Откройте редактор загрузки данных и добавьте приведенный ниже скрипт загрузки на новую вкладку.

В этом примере набор данных с размещенными заказами загружается в таблицу под именем Sales. Данная таблица содержит слишком много полей.

- `id`
- `sales_date`
- `amount`

Гарантии при продаже продукта действуют в течение двух лет с даты продажи. Задача – создать меру в диаграмме для определения года, когда истекает срок действия гарантии.

Скрипт загрузки

```
Sales:
Load
id,
sales_date,
amount
Inline
[
id,sales_date,amount
1,12/28/2020,231.24,
2,12/29/2020,567.28,
3,12/30/2020,364.28,
4,12/31/2020,575.76,
5,01/01/2021,638.68,
6,01/02/2021,785.38,
7,01/03/2021,967.46,
8,01/04/2021,287.67
9,01/05/2021,764.45,
10,01/06/2021,875.43,
11,01/07/2021,957.35
];
```

Результаты

Загрузите данные и откройте лист. Создайте новую таблицу и добавьте это поле как измерение: `sales_date`.

Создайте следующую меру:

```
=year(sales_date+365*2)
```

Результирующая таблица

sales_date	=year(sales_date+365*2)
12/28/2020	2022
12/29/2020	2022
12/30/2020	2022
12/31/2020	2022
01/01/2021	2023
01/02/2021	2023
01/03/2021	2023
01/04/2021	2023
01/05/2021	2023
01/06/2021	2023
01/07/2021	2023

Результаты этой меры приводятся в таблице выше. Чтобы прибавить два года к дате, необходимо умножить 365 на 2 и прибавить результат к дате продажи. Таким образом, для продаж, совершенных в 2020 году, срок действия гарантии истекает в 2022 году.

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]])
```

Другими словами, функция `yearend()` определяет, на какой год приходится дата. Затем она возвращает метку времени в формате даты для последней миллисекунды этого года. По умолчанию первым месяцем года является январь. Однако также можно изменить первый месяц, используя аргумент `first_month_of_year` в функции `yearend()`.

> *Функция `yearend()` не учитывает системную переменную `FirstMonthOfYear`. Год начинается 1 января, если для его изменения не используется аргумент `first_month_of_year`.*

Схема функции `yearend()`.

Когда это следует использовать

Функция `yearend()` используется в составе выражения, когда требуется использовать в расчетах часть года, которая еще не прошла, например, если требуется рассчитать общую сумму процентов, еще не начисленных в течение года.

Возвращаемые типы данных: двойное значение

Аргументы

Аргумент	Описание
date	Дата или метка времени для вычисления.

Аргумент	Описание
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**.

Можно использовать следующие значения, чтобы задать первый месяц года в аргументе `first_month_of_year`:

значения first_month_of_year

Месяц	Значение
Февраль	2
Март	3
Апрель	4
Мау	5
Июнь	6
Июль	7
Август	8
Сентябрь	9
Октябрь	10
Ноябрь	11
Декабрь	12

Региональные настройки

Если не указано иное, в примерах, приведенных в данном разделе, используется следующий формат даты: ММ/ДД/ГГГГ. Формат даты указан в операторе `SET DateFormat` скрипта загрузки данных. В вашей системе может быть установлен другой формат даты по умолчанию в зависимости от региональных настроек и других факторов. Можно изменить формат в примерах в соответствии с потребностями. Или можно изменить форматы в скрипте загрузки в соответствии с этими примерами.

Региональные настройки по умолчанию в приложениях основаны на системных региональных настройках компьютера или сервера, где установлено ПО Qlik Sense. Если на сервере Qlik Sense, к которому обращается пользователь, выбран шведский язык, то редактор загрузки данных будет использовать шведские региональные настройки для даты, времени и валюты. Эти параметры регионального формата не связаны с языком, отображаемым в интерфейсе пользователя Qlik Sense. Qlik Sense будет отображаться на языке, который используется в браузере.

Примеры функции

Пример	Результат
`yearend('10/19/2001')`	Возвращает 12/31/2001 23:59:59.
`yearend('10/19/2001', -1)`	Возвращает 12/31/2000 23:59:59.
`yearend('10/19/2001', 0, 4)`	Возвращает 03/31/2002 23:59:59.

Пример 1. Без дополнительных аргументов

Скрипт загрузки и результаты

Обзор

Откройте редактор загрузки данных и добавьте приведенный ниже скрипт загрузки на новую вкладку.

Скрипт загрузки содержит следующее:

- Набор данных, содержащий набор транзакций с 2020 по 2022 год, загружается в таблицу под именем «`Transactions`».
- Поле даты было предоставлено в формате `(MM/DD/YYYY)`системной переменной `DateFormat`.
- Предыдущий оператор load, который содержит следующее:
  - Функция `yearend()`, заданная как поле `year_end`.
  - Функция `Timestamp()`, заданная как поле `year_end_timestamp`.

Скрипт загрузки

```
SET DateFormat='MM/DD/YYYY';

Transactions:
    Load
       *,
       yearend(date) as year_end,
       timestamp(yearend(date)) as year_end_timestamp
       ;
Load
*
Inline
[
id,date,amount
8188,01/13/2020,37.23
8189,02/26/2020,17.17
8190,03/27/2020,88.27
8191,04/16/2020,57.42
8192,05/21/2020,53.80
8193,08/14/2020,82.06
8194,10/07/2020,40.39
8195,12/05/2020,87.21
```

```
8196,01/22/2021,95.93
8197,02/03/2021,45.89
8198,03/17/2021,36.23
8199,04/23/2021,25.66
8200,05/04/2021,82.77
8201,06/30/2021,69.98
8202,07/26/2021,76.11
8203,12/27/2021,25.12
8204,06/06/2022,46.23
8205,07/18/2022,84.21
8206,11/14/2022,96.24
8207,12/12/2022,67.67
];
```

Результаты

Загрузите данные и откройте лист. Создайте новую таблицу и добавьте эти поля как измерения:

- id
- date
- year_end
- year_end_timestamp

Результирующая таблица

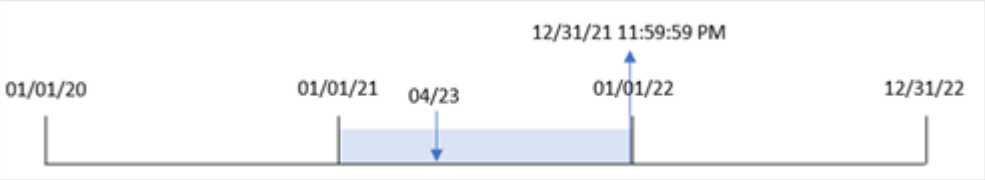
id	date	year_end	year_end_timestamp
8188	01/13/2020	12/31/2020	12/31/2020 11:59:59 PM
8189	02/26/2020	12/31/2020	12/31/2020 11:59:59 PM
8190	03/27/2020	12/31/2020	12/31/2020 11:59:59 PM
8191	04/16/2020	12/31/2020	12/31/2020 11:59:59 PM
8192	05/21/2020	12/31/2020	12/31/2020 11:59:59 PM
8193	08/14/2020	12/31/2020	12/31/2020 11:59:59 PM
8194	10/07/2020	12/31/2020	12/31/2020 11:59:59 PM
8195	12/05/2020	12/31/2020	12/31/2020 11:59:59 PM
8196	01/22/2021	12/31/2021	12/31/2021 11:59:59 PM
8197	02/03/2021	12/31/2021	12/31/2021 11:59:59 PM
8198	03/17/2021	12/31/2021	12/31/2021 11:59:59 PM
8199	04/23/2021	12/31/2021	12/31/2021 11:59:59 PM
8200	05/04/2021	12/31/2021	12/31/2021 11:59:59 PM
8201	06/30/2021	12/31/2021	12/31/2021 11:59:59 PM
8202	07/26/2021	12/31/2021	12/31/2021 11:59:59 PM

id	date	year_end	year_end_timestamp
8203	12/27/2021	12/31/2021	12/31/2021 11:59:59 PM
8204	06/06/2022	12/31/2022	12/31/2022 11:59:59 PM
8205	07/18/2022	12/31/2022	12/31/2022 11:59:59 PM
8206	11/14/2022	12/31/2022	12/31/2022 11:59:59 PM
8207	12/12/2022	12/31/2022	12/31/2022 11:59:59 PM

Поле `year_end` создано предшествующим оператором load с использованием функции `yearend()`, где в качестве аргумента функции передано поле даты.

Функция `yearend()` первоначально определяет, к какому году относится значение даты, и возвращает метку времени для последней миллисекунды этого года.

Диаграмма функции `yearend()` с выбранной транзакцией 8199.



Транзакция 8199 совершена 23 апреля 2021 года. Функция `yearend()` возвращает последнюю миллисекунду этого года, то есть 31 декабря в 23:59:59.

Пример 2. Скрипт period_no

Скрипт загрузки и результаты

Обзор

Используется тот же набор данных и сценарий, что в первом примере.

Однако в этом примере задача состоит в том, чтобы создать поле `previous_year_end`, которое возвращает метку времени даты окончания года, предшествующего тому, в котором совершена транзакция.

Скрипт загрузки

```
SET DateFormat='MM/DD/YYYY';

Transactions:
    Load
       *,
       yearend(date,-1) as previous_year_end,
       timestamp(yearend(date,-1)) as previous_year_end_timestamp
       ;
Load
*
```

```
Inline
[
id,date,amount
8188,01/13/2020,37.23
8189,02/26/2020,17.17
8190,03/27/2020,88.27
8191,04/16/2020,57.42
8192,05/21/2020,53.80
8193,08/14/2020,82.06
8194,10/07/2020,40.39
8195,12/05/2020,87.21
8196,01/22/2021,95.93
8197,02/03/2021,45.89
8198,03/17/2021,36.23
8199,04/23/2021,25.66
8200,05/04/2021,82.77
8201,06/30/2021,69.98
8202,07/26/2021,76.11
8203,12/27/2021,25.12
8204,06/06/2022,46.23
8205,07/18/2022,84.21
8206,11/14/2022,96.24
8207,12/12/2022,67.67
];
```

Результаты

Загрузите данные и откройте лист. Создайте новую таблицу и добавьте эти поля как измерения:

- id
- date
- previous_year_end
- previous_ year_end_timestamp

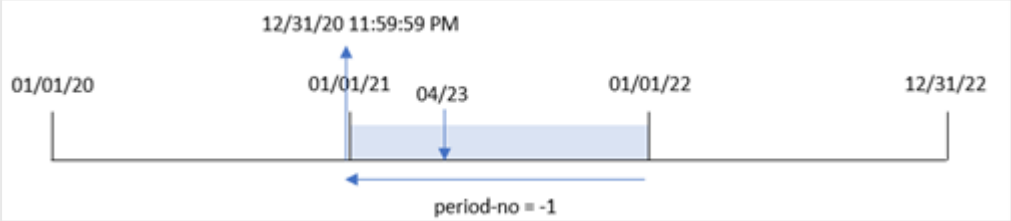
Результирующая таблица

id	date	previous_year_end	previous_year_end_timestamp
8188	01/13/2020	12/31/2019	12/31/2019 11:59:59 PM
8189	02/26/2020	12/31/2019	12/31/2019 11:59:59 PM
8190	03/27/2020	12/31/2019	12/31/2019 11:59:59 PM
8191	04/16/2020	12/31/2019	12/31/2019 11:59:59 PM
8192	05/21/2020	12/31/2019	12/31/2019 11:59:59 PM
8193	08/14/2020	12/31/2019	12/31/2019 11:59:59 PM
8194	10/07/2020	12/31/2019	12/31/2019 11:59:59 PM
8195	12/05/2020	12/31/2019	12/31/2019 11:59:59 PM
8196	01/22/2021	12/31/2020	12/31/2020 11:59:59 PM

id	date	previous_year_end	previous_year_end_timestamp
8197	02/03/2021	12/31/2020	12/31/2020 11:59:59 PM
8198	03/17/2021	12/31/2020	12/31/2020 11:59:59 PM
8199	04/23/2021	12/31/2020	12/31/2020 11:59:59 PM
8200	05/04/2021	12/31/2020	12/31/2020 11:59:59 PM
8201	06/30/2021	12/31/2020	12/31/2020 11:59:59 PM
8202	07/26/2021	12/31/2020	12/31/2020 11:59:59 PM
8203	12/27/2021	12/31/2020	12/31/2020 11:59:59 PM
8204	06/06/2022	12/31/2021	12/31/2021 11:59:59 PM
8205	07/18/2022	12/31/2021	12/31/2021 11:59:59 PM
8206	11/14/2022	12/31/2021	12/31/2021 11:59:59 PM
8207	12/12/2022	12/31/2021	12/31/2021 11:59:59 PM

Так как в качестве аргумента смещения в функции `yearend()` использовалось `period_no = -1`, функция сначала определяет год, в течение которого совершены транзакции. Затем она возвращается на год назад и определяет последнюю миллисекунду этого года.

Диаграмма функции `yearend()` с period_no = -1.



Транзакция 8199 совершена 23 апреля 2021 года. Функция `yearend()` возвращает последнюю миллисекунду предыдущего года, то есть 31 декабря 2020 года в 23:59:59, для поля `previous_year_end`.

Пример 3. Аргумент first_month_of_year

Скрипт загрузки и результаты

Обзор

Используется тот же набор данных и сценарий, что в первом примере.

Однако в этом примере политика компании такова, что год начинается 1 апреля.

Скрипт загрузки

```
SET DateFormat='MM/DD/YYYY';
```

```
Transactions:
    Load
       *,
       yearend(date,0,4) as year_end,
       timestamp(yearend(date,0,4)) as year_end_timestamp
       ;
Load
*
Inline
[
id,date,amount
8188,01/13/2020,37.23
8189,02/26/2020,17.17
8190,03/27/2020,88.27
8191,04/16/2020,57.42
8192,05/21/2020,53.80
8193,08/14/2020,82.06
8194,10/07/2020,40.39
8195,12/05/2020,87.21
8196,01/22/2021,95.93
8197,02/03/2021,45.89
8198,03/17/2021,36.23
8199,04/23/2021,25.66
8200,05/04/2021,82.77
8201,06/30/2021,69.98
8202,07/26/2021,76.11
8203,12/27/2021,25.12
8204,06/06/2022,46.23
8205,07/18/2022,84.21
8206,11/14/2022,96.24
8207,12/12/2022,67.67
];
```

Результаты

Загрузите данные и откройте лист. Создайте новую таблицу и добавьте эти поля как измерения:

- id
- date
- year_end
- year_end_timestamp

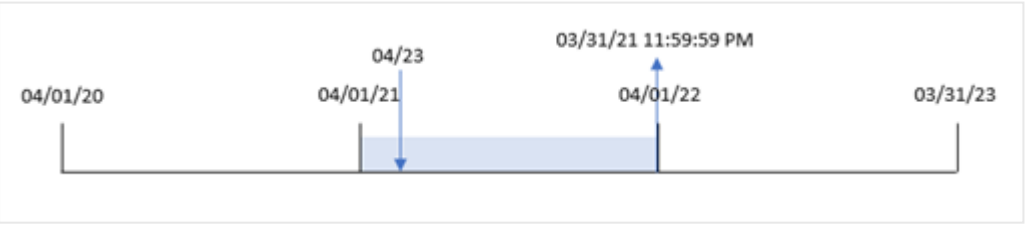
Результирующая таблица

id	date	year_end	year_end_timestamp
8188	01/13/2020	03/31/2020	3/31/2020 11:59:59 PM
8189	02/26/2020	03/31/2020	3/31/2020 11:59:59 PM
8190	03/27/2020	03/31/2020	3/31/2020 11:59:59 PM
8191	04/16/2020	03/31/2021	3/31/2021 11:59:59 PM

id	date	year_end	year_end_timestamp
8192	05/21/2020	03/31/2021	3/31/2021 11:59:59 PM
8193	08/14/2020	03/31/2021	3/31/2021 11:59:59 PM
8194	10/07/2020	03/31/2021	3/31/2021 11:59:59 PM
8195	12/05/2020	03/31/2021	3/31/2021 11:59:59 PM
8196	01/22/2021	03/31/2021	3/31/2021 11:59:59 PM
8197	02/03/2021	03/31/2021	3/31/2021 11:59:59 PM
8198	03/17/2021	03/31/2021	3/31/2021 11:59:59 PM
8199	04/23/2021	03/31/2022	3/31/2022 11:59:59 PM
8200	05/04/2021	03/31/2022	3/31/2022 11:59:59 PM
8201	06/30/2021	03/31/2022	3/31/2022 11:59:59 PM
8202	07/26/2021	03/31/2022	3/31/2022 11:59:59 PM
8203	12/27/2021	03/31/2022	3/31/2022 11:59:59 PM
8204	06/06/2022	03/31/2023	3/31/2023 11:59:59 PM
8205	07/18/2022	03/31/2023	3/31/2023 11:59:59 PM
8206	11/14/2022	03/31/2023	3/31/2023 11:59:59 PM
8207	12/12/2022	03/31/2023	3/31/2023 11:59:59 PM

Поскольку аргумент `first_month_of_year = 4` используется в функции `yearend()`, в качестве первого дня года задается 1 апреля, а в качестве последнего дня года – 31 марта.

Диаграмма функции `yearend()` с апрелем в качестве первого месяца года.



Транзакция 8199 совершена 23 апреля 2021 года. Поскольку функция `yearend()` задает в качестве начала года 1 апреля, она возвращает 31 марта 2022 года в качестве значения `year_end` для транзакции.

Пример 4. Пример объекта диаграммы

Скрипт загрузки и выражение диаграммы

Обзор

Используется тот же набор данных и сценарий, что в первом примере.

Однако в этом примере в приложение загружается неизмененный набор данных. Расчет, который возвращает метку даты окончания года, в котором совершены транзакции, создается как мера в объекте диаграммы в приложении.

Скрипт загрузки

```
Transactions:
Load
*
Inline
[
id,date,amount
8188,01/13/2020,37.23
8189,02/26/2020,17.17
8190,03/27/2020,88.27
8191,04/16/2020,57.42
8192,05/21/2020,53.80
8193,08/14/2020,82.06
8194,10/07/2020,40.39
8195,12/05/2020,87.21
8196,01/22/2021,95.93
8197,02/03/2021,45.89
8198,03/17/2021,36.23
8199,04/23/2021,25.66
8200,05/04/2021,82.77
8201,06/30/2021,69.98
8202,07/26/2021,76.11
8203,12/27/2021,25.12
8204,06/06/2022,46.23
8205,07/18/2022,84.21
8206,11/14/2022,96.24
8207,12/12/2022,67.67
];
```

Результаты

Загрузите данные и откройте лист. Создайте новую таблицу и добавьте эти поля как измерения:

- `id`
- `date`

Чтобы рассчитать год, в котором совершена транзакция, создайте следующие меры:

- `=yearend(date)`
- `=timestamp(yearend(date))`

Результирующая таблица

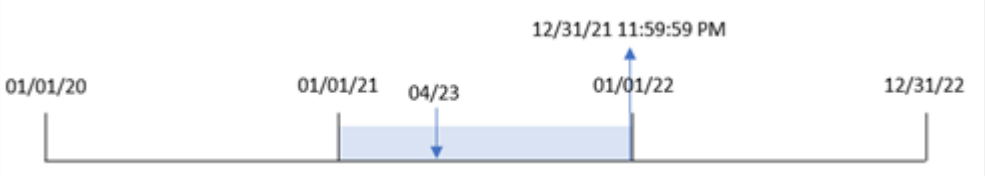
id	date	=yearend(date)	=timestamp(yearend(date))
8188	01/13/2020	12/31/2020	12/31/2020 11:59:59 PM

id	date	=yearend(date)	=timestamp(yearend(date))
8189	02/26/2020	12/31/2020	12/31/2020 11:59:59 PM
8190	03/27/2020	12/31/2020	12/31/2020 11:59:59 PM
8191	04/16/2020	12/31/2020	12/31/2020 11:59:59 PM
8192	05/21/2020	12/31/2020	12/31/2020 11:59:59 PM
8193	08/14/2020	12/31/2020	12/31/2020 11:59:59 PM
8194	10/07/2020	12/31/2020	12/31/2020 11:59:59 PM
8195	12/05/2020	12/31/2020	12/31/2020 11:59:59 PM
8196	01/22/2021	12/31/2021	12/31/2021 11:59:59 PM
8197	02/03/2021	12/31/2021	12/31/2021 11:59:59 PM
8198	03/17/2021	12/31/2021	12/31/2021 11:59:59 PM
8199	04/23/2021	12/31/2021	12/31/2021 11:59:59 PM
8200	05/04/2021	12/31/2021	12/31/2021 11:59:59 PM
8201	06/30/2021	12/31/2021	12/31/2021 11:59:59 PM
8202	07/26/2021	12/31/2021	12/31/2021 11:59:59 PM
8203	12/27/2021	12/31/2021	12/31/2021 11:59:59 PM
8204	06/06/2022	12/31/2022	12/31/2022 11:59:59 PM
8205	07/18/2022	12/31/2022	12/31/2022 11:59:59 PM
8206	11/14/2022	12/31/2022	12/31/2022 11:59:59 PM
8207	12/12/2022	12/31/2022	12/31/2022 11:59:59 PM

Мера `end_of_year` создана в объекте диаграммы с использованием функции `yearend()`, где в качестве аргумента функции передано поле даты.

Функция `yearend()` первоначально определяет, к какому году относится значение даты, и возвращает метку времени для последней миллисекунды этого года.

Диаграмма функции `yearend()`, которая показывает, что транзакция 8199 совершена в апреле.



Транзакция 8199 совершена 23 апреля 2021 года. Функция `yearend()` возвращает последнюю миллисекунду этого года, то есть 31 декабря в 23:59:59.

Пример 5. Сценарий

Скрипт загрузки и выражение диаграммы

Обзор

Откройте редактор загрузки данных и добавьте приведенный ниже скрипт загрузки на новую вкладку.

Скрипт загрузки содержит следующее:

- Набор данных загружается в таблицу под именем «`Employee_Expenses`». Данная таблица содержит следующие поля:
    - employee IDs (ИД работника)
    - employee name (имя работника)
    - average daily expense claims of each employee (средние ежедневные заявки на возмещение расходов каждого работника)

Конечному пользователю требуется получить объект диаграммы, отображающий по идентификатору и имени сотрудника расчетные расходы, которые еще предстоит понести в течение оставшейся части года. Финансовый год начинается в январе.

Скрипт загрузки

```
Employee_Expenses:
Load
*
Inline
[
employee_id,employee_name,avg_daily_claim
182,Mark, $15
183,Deryck, $12.5
184,Dexter, $12.5
185,Sydney,$27
186,Agatha,$18
];
```

Результаты

Загрузите данные и откройте лист. Создайте новую таблицу и добавьте эти поля как измерения:

- `employee_id`
- `employee_name`

Чтобы рассчитать планируемые заявки на возмещение расходов, создайте следующую меру:

```
=(yearend(today(1))-today(1))*avg_daily_claim
```

Задайте параметру **Формат чисел** меры значение **Денежный**.

Результирующая таблица

employee_id	employee_name	=(yearend(today(1))-today(1))*avg_daily_claim
182	Mark	$3240.00
183	Deryck	$2700.00
184	Dexter	$2700.00
185	Sydney	$5832.00
186	Agatha	$3888.00

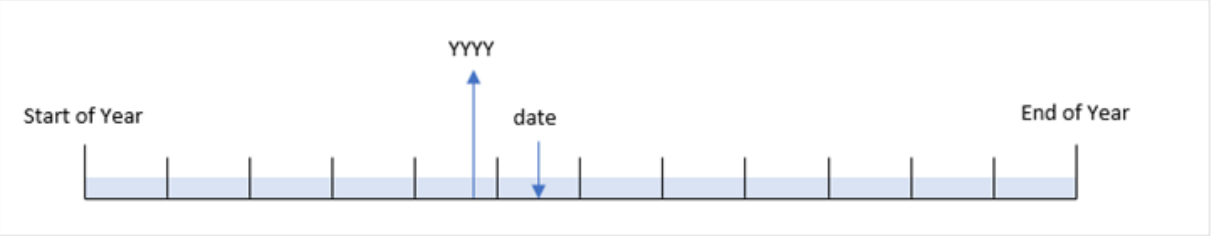
Используя сегодняшнюю дату в качестве единственного аргумента, функция `yearend()` возвращает дату окончания текущего года. Затем, вычитая сегодняшнюю дату из даты окончания года, выражение возвращает количество дней, оставшихся в этом году.

Затем это значение умножается на среднюю ежедневную заявку на возмещение расходов каждого сотрудника для расчета оценочной суммы заявок, которые каждый сотрудник должен подать до конца года.

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**.

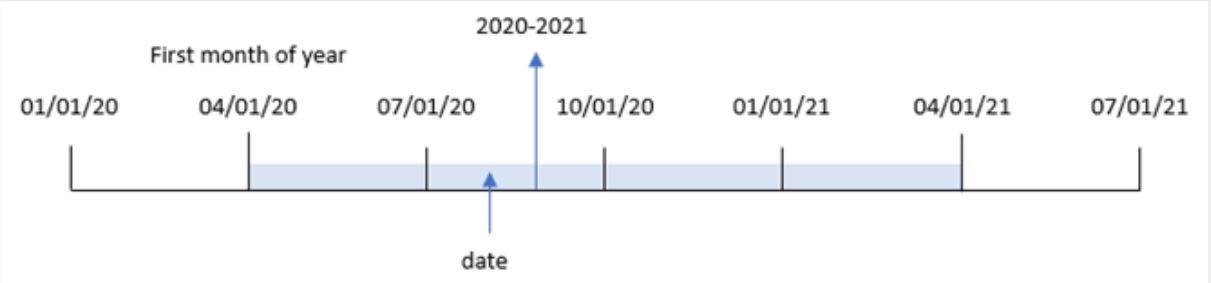
Диаграмма диапазона времени функции `yearname()`.



Функция `yearname()` отличается от функции `year()` тем, что позволяет сместить дату, которую требуется оценить, и позволяет задать первый месяц года.

Если первый месяц года не январь, функция вернет два четырехзначных значения года в течение двенадцатимесячного периода, содержащего дату. Например, если начало года – апрель, а оцениваемая дата – 06/30/2020, будет возвращен результат будет `2020-2021`.

Диаграмма функции `yearname()`, где в качестве первого месяца года задан апрель.



Синтаксис:

```
YearName(date[, period_no[, first_month_of_year]] )
```

Возвращаемые типы данных: двойное значение

Аргумент	Описание
date	Дата или метка времени 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.

Можно использовать следующие значения, чтобы задать первый месяц года в аргументе `first_month_of_year`:

значения first_month_of_year

Месяц	Значение
Февраль	2
Март	3
Апрель	4
Мау	5
Июнь	6
Июль	7
Август	8
Сентябрь	9
Октябрь	10
Ноябрь	11
Декабрь	12

Когда это следует использовать

Функция `yearname()` полезна для сравнения агрегирования по годам. Например, если требуется увидеть общий объем продаж продуктов по году.

Эти измерения можно создать в скрипте загрузки с помощью функции создания поля в таблице основного календаря. Их также можно создать в диаграмме как вычисляемые измерения.

Региональные настройки

Если не указано иное, в примерах, приведенных в данном разделе, используется следующий формат даты: ММ/ДД/ГГГГ. Формат даты указан в операторе `SET DateFormat` скрипта загрузки данных. В вашей системе может быть установлен другой формат даты по умолчанию в зависимости от региональных настроек и других факторов. Можно изменить формат в примерах в соответствии с потребностями. Или можно изменить форматы в скрипте загрузки в соответствии с этими примерами.

Региональные настройки по умолчанию в приложениях основаны на системных региональных настройках компьютера или сервера, где установлено ПО Qlik Sense. Если на сервере Qlik Sense, к которому обращается пользователь, выбран шведский язык, то редактор загрузки данных будет использовать шведские региональные настройки для даты, времени и валюты. Эти параметры регионального формата не связаны с языком, отображаемым в интерфейсе пользователя Qlik Sense. Qlik Sense будет отображаться на языке, который используется в браузере.

Примеры функции

Пример	Результат
`yearname('10/19/2001')`	Возвращает «2001».
`yearname('10/19/2001',-1)`	Возвращает «2000».
`yearname('10/19/2001',0,4)`	Возвращает «2001-2002».

Связанные темы

Тема	Описание
year (page 1145)	Эта функция возвращает год в виде целого числа, а выражение интерпретируется как дата согласно стандартной интерпретации чисел.

Пример 1. Без дополнительных аргументов

Скрипт загрузки и результаты

Обзор

Откройте редактор загрузки данных и добавьте приведенный ниже скрипт загрузки на новую вкладку.

Скрипт загрузки содержит следующее:

- Набор данных, содержащий набор транзакций с 2020 по 2022 год, загружается в таблицу под именем «Transactions».
- Системная переменная DateFormat, для которой задан формат MM/DD/YYYY.
- Предшествующая загрузка, которая использует функцию yearname(), заданную как поле year_name.

Скрипт загрузки

```
SET DateFormat='MM/DD/YYYY';

Transactions:
    Load
      *,
      yearname(date) as year_name
      ;
Load
*
Inline
[
id,date,amount
8188,'01/13/2020',37.23
8189,'02/26/2020',17.17
8190,'03/27/2020',88.27
8191,'04/16/2020',57.42
8192,'05/21/2020',53.80
8193,'08/14/2020',82.06
8194,'10/07/2020',40.39
8195,'12/05/2020',87.21
8196,'01/22/2021',95.93
8197,'02/03/2021',45.89
8198,'03/17/2021',36.23
8199,'04/23/2021',25.66
8200,'05/04/2021',82.77
8201,'06/30/2021',69.98
8202,'07/26/2021',76.11
8203,'12/27/2021',25.12
8204,'06/06/2022',46.23
8205,'07/18/2022',84.21
8206,'11/14/2022',96.24
8207,'12/12/2022',67.67
];
```

Результаты

Загрузите данные и откройте лист. Создайте новую таблицу и добавьте эти поля как измерения:

- date
- year_name

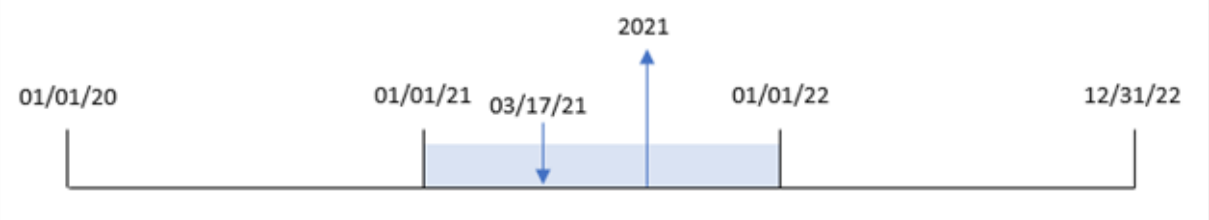
Результирующая таблица

date	year_name
01/13/2020	2020
02/26/2020	2020
03/27/2020	2020
04/16/2020	2020
05/21/2020	2020
08/14/2020	2020
10/07/2020	2020
12/05/2020	2020
01/22/2021	2021
02/03/2021	2021
03/17/2021	2021
04/23/2021	2021
05/04/2021	2021
06/30/2021	2021
07/26/2021	2021
12/27/2021	2021
06/06/2022	2022
07/18/2022	2022
11/14/2022	2022
12/12/2022	2022

Поле `year_name` создано предшествующим оператором load с использованием функции `yearname()`, где в качестве аргумента функции передано поле даты.

Функция `yearname()` определяет, к какому году относится значение даты, и возвращает его в виде четырехзначного значения года.

Диаграмма функции `yearname()`, которая показывает 2021 как значение года.



Пример 2. Скрипт period_no

Скрипт загрузки и результаты

Обзор

Откройте редактор загрузки данных и добавьте приведенный ниже скрипт загрузки на новую вкладку.

Скрипт загрузки содержит следующее:

- Набор данных, содержащий набор транзакций с 2020 по 2022 год, загружается в таблицу под именем Transactions.
- Системная переменная DateFormat, для которой задан формат `MM/DD/YYYY`.
- Предшествующая загрузка, которая использует функцию `yearname()`, заданную как поле `year_name`.

Скрипт загрузки

```
SET DateFormat='MM/DD/YYYY';

Transactions:
    Load
      *,
      yearname(date,-1) as prior_year_name
      ;
Load
*
Inline
[
id,date,amount
8188,'01/13/2020',37.23
8189,'02/26/2020',17.17
8190,'03/27/2020',88.27
8191,'04/16/2020',57.42
8192,'05/21/2020',53.80
8193,'08/14/2020',82.06
8194,'10/07/2020',40.39
8195,'12/05/2020',87.21
8196,'01/22/2021',95.93
8197,'02/03/2021',45.89
8198,'03/17/2021',36.23
8199,'04/23/2021',25.66
8200,'05/04/2021',82.77
8201,'06/30/2021',69.98
8202,'07/26/2021',76.11
8203,'12/27/2021',25.12
8204,'06/06/2022',46.23
8205,'07/18/2022',84.21
8206,'11/14/2022',96.24
8207,'12/12/2022',67.67
];
```

Результаты

Загрузите данные и откройте лист. Создайте новую таблицу и добавьте эти поля как измерения:

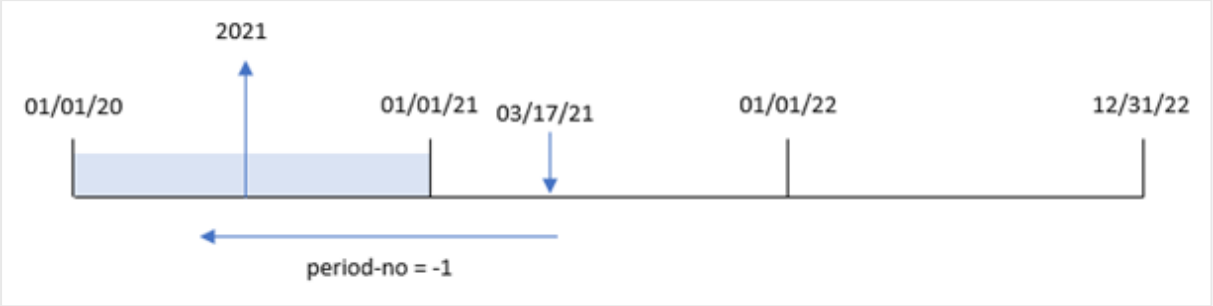
- `date`
- `prior_year_name`

Результирующая таблица

date	prior_year_name
01/13/2020	2019
02/26/2020	2019
03/27/2020	2019
04/16/2020	2019
05/21/2020	2019
08/14/2020	2019
10/07/2020	2019
12/05/2020	2019
01/22/2021	2020
02/03/2021	2020
03/17/2021	2020
04/23/2021	2020
05/04/2021	2020
06/30/2021	2020
07/26/2021	2020
12/27/2021	2020
06/06/2022	2021
07/18/2022	2021
11/14/2022	2021
12/12/2022	2021

Так как в качестве аргумента смещения в функции `yearname()` используется `period_no = -1`, функция сначала определяет год, в течение которого совершены транзакции. Затем функция сдвигается на один год назад и возвращает результирующий год.

Диаграмма функции `yearname()` со смещением `period_no = -1`.



Пример 3. Аргумент first_month_of_year

Скрипт загрузки и результаты

Обзор

Откройте редактор загрузки данных и добавьте приведенный ниже скрипт загрузки на новую вкладку.

Скрипт загрузки содержит следующее:

- Используется тот же набор данных, что и в первом примере.
- Системная переменная `DateFormat`, для которой задан формат `MM/DD/YYYY`.
- Предшествующая загрузка, которая использует функцию `yearname()`, заданную как поле `year_name`.

Скрипт загрузки

```
SET DateFormat='MM/DD/YYYY';

Transactions:
    Load
      *,
      yearname(date,0,4) as year_name
      ;
Load
*
Inline
[
id,date,amount
8188,'01/13/2020',37.23
8189,'02/26/2020',17.17
8190,'03/27/2020',88.27
8191,'04/16/2020',57.42
8192,'05/21/2020',53.80
8193,'08/14/2020',82.06
8194,'10/07/2020',40.39
8195,'12/05/2020',87.21
8196,'01/22/2021',95.93
8197,'02/03/2021',45.89
```

```
8198,'03/17/2021',36.23
8199,'04/23/2021',25.66
8200,'05/04/2021',82.77
8201,'06/30/2021',69.98
8202,'07/26/2021',76.11
8203,'12/27/2021',25.12
8204,'06/06/2022',46.23
8205,'07/18/2022',84.21
8206,'11/14/2022',96.24
8207,'12/12/2022',67.67
];
```

Результаты

Загрузите данные и откройте лист. Создайте новую таблицу и добавьте эти поля как измерения:

- `id`
- `date`
- `year_name`

Результирующая таблица

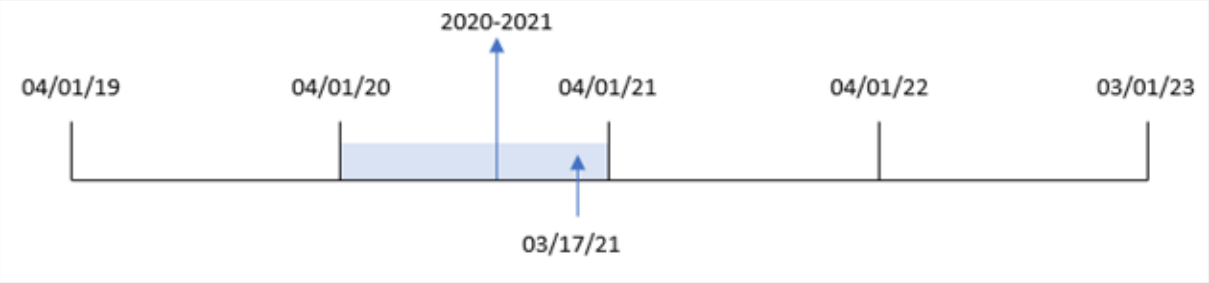
date	year_name
01/13/2020	2019-2020
02/26/2020	2019-2020
03/27/2020	2019-2020
04/16/2020	2020-2021
05/21/2020	2020-2021
08/14/2020	2020-2021
10/07/2020	2020-2021
12/05/2020	2020-2021
01/22/2021	2020-2021
02/03/2021	2020-2021
03/17/2021	2020-2021
04/23/2021	2021-2022
05/04/2021	2021-2022
06/30/2021	2021-2022
07/26/2021	2021-2022
12/27/2021	2021-2022
06/06/2022	2022-2023

date	year_name
07/18/2022	2022-2023
11/14/2022	2022-2023
12/12/2022	2022-2023

Поскольку аргумент `first_month_of_year` функции `4` используется в функции `yearname()`, начало года перемещается с 1 января на 1 апреля. Таким образом, каждый двенадцатимесячный период пересекает два календарных года, а функция `yearname()` возвращает два года в виде четырехзначных чисел для оцениваемых дат.

Транзакция 8199 совершена 17 марта 2021 года. Функция `yearname()` задает в качестве начала года 1 апреля и в качестве окончания 30 марта. Таким образом, транзакция 8198 произошла в периоде с 1 апреля 2020 г. по 30 марта 2021 г. В результате функция `yearname()` возвращает значение 2020-2021.

Диаграмма функции `yearname()`, где в качестве первого месяца года задан март.



Пример 4. Пример объекта диаграммы

Скрипт загрузки и выражение диаграммы

Обзор

Откройте редактор загрузки данных и добавьте приведенный ниже скрипт загрузки на новую вкладку.

Скрипт загрузки содержит следующее:

- Используется тот же набор данных, что и в первом примере.
- Системная переменная `DateFormat`, для которой задан формат `MM/DD/YYYY`.

Однако поле, возвращающее год, в котором совершена транзакция, создается как мера в объекте диаграммы.

Скрипт загрузки

```
SET DateFormat='MM/DD/YYYY';

Transactions:
```

```
Load
*
Inline
[
id,date,amount
8188,'01/13/2020',37.23
8189,'02/26/2020',17.17
8190,'03/27/2020',88.27
8191,'04/16/2020',57.42
8192,'05/21/2020',53.80
8193,'08/14/2020',82.06
8194,'10/07/2020',40.39
8195,'12/05/2020',87.21
8196,'01/22/2021',95.93
8197,'02/03/2021',45.89
8198,'03/17/2021',36.23
8199,'04/23/2021',25.66
8200,'05/04/2021',82.77
8201,'06/30/2021',69.98
8202,'07/26/2021',76.11
8203,'12/27/2021',25.12
8204,'06/06/2022',46.23
8205,'07/18/2022',84.21
8206,'11/14/2022',96.24
8207,'12/12/2022',67.67
];
```

Результаты

Загрузите данные и откройте лист. Создайте новую таблицу и добавьте это поле как измерение:

```
date
```

Чтобы рассчитать поле `year_name`, создайте эту меру:

```
=yearname(date)
```

Результирующая таблица

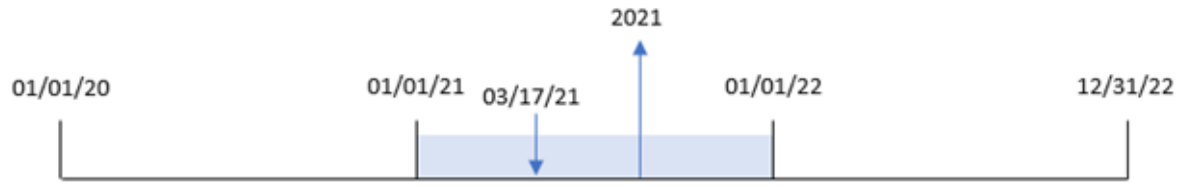
date	=yearname(date)
01/13/2020	2020
02/26/2020	2020
03/27/2020	2020
04/16/2020	2020
05/21/2020	2020
08/14/2020	2020
10/07/2020	2020
12/05/2020	2020
01/22/2021	2021

date	=yearname(date)
02/03/2021	2021
03/17/2021	2021
04/23/2021	2021
05/04/2021	2021
06/30/2021	2021
07/26/2021	2021
12/27/2021	2021
06/06/2022	2022
07/18/2022	2022
11/14/2022	2022
12/12/2022	2022

Мера `year_name` создана в объекте диаграммы с использованием функции `yearname()`, где в качестве аргумента функции передано поле даты.

Функция `yearname()` определяет, к какому году относится значение даты, и возвращает его в виде четырехзначного значения года.

Диаграмма функции `yearname()`, которая показывает 2021 как значение года.



Пример 5. Сценарий

Скрипт загрузки и выражение диаграммы

Обзор

Откройте редактор загрузки данных и добавьте приведенный ниже скрипт загрузки на новую вкладку.

Скрипт загрузки содержит следующее:

- Используется тот же набор данных, что и в первом примере.
- Системная переменная `DateFormat`, для которой задан формат `MM/DD/YYYY`.

Конечный пользователь хотел бы получить диаграмму, на которой представлены общие продажи по кварталам для транзакций. Используйте функцию `yearname()` в качестве вычисляемого измерения для создания этой диаграммы, когда измерение `yearname()` недоступно в модели данных.

Скрипт загрузки

```
SET DateFormat='MM/DD/YYYY';

Transactions:
Load
*
Inline
[
id,date,amount
8188,'01/13/2020',37.23
8189,'02/26/2020',17.17
8190,'03/27/2020',88.27
8191,'04/16/2020',57.42
8192,'05/21/2020',53.80
8193,'08/14/2020',82.06
8194,'10/07/2020',40.39
8195,'12/05/2020',87.21
8196,'01/22/2021',95.93
8197,'02/03/2021',45.89
8198,'03/17/2021',36.23
8199,'04/23/2021',25.66
8200,'05/04/2021',82.77
8201,'06/30/2021',69.98
8202,'07/26/2021',76.11
8203,'12/27/2021',25.12
8204,'06/06/2022',46.23
8205,'07/18/2022',84.21
8206,'11/14/2022',96.24
8207,'12/12/2022',67.67
];
```

Результаты

Загрузите данные и откройте лист. Создайте новую таблицу.

Чтобы сравнить сводки по годам, создайте это вычисляемое измерение:

```
=yearname(date)
```

Создайте это измерение:

```
=sum(amount)
```

Задайте параметру **Формат чисел** меры значение **Денежный**.

Результирующая таблица

yearname(date)	=sum(amount)
2020	$463.55
2021	$457.69
2022	$294.35

yearstart

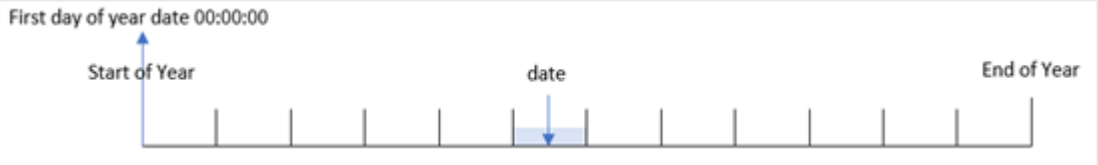
Эта функция возвращает метку времени, соответствующую началу первого дня года, содержащего значение **date**. По умолчанию для вывода используется формат **DateFormat**, установленный в скрипте.

Синтаксис:

```
YearStart(date[, period_no[, first_month_of_year]])
```

Другими словами, функция `yearstart()` определяет, на какой год приходится дата. Затем она возвращает метку времени в формате даты для первой миллисекунды этого года. По умолчанию первым месяцем года является январь. Однако первый месяц можно изменить, используя аргумент `first_month_of_year` в функции `yearstart()`.

Диаграмма функции `yearstart()`, показывающая диапазон времени, который может охватывать функция.



Когда это следует использовать

Функция `yearstart()` используется в составе выражения, когда требуется использовать в расчетах часть года, которая уже прошла, например, если требуется рассчитать проценты, накопленные за год до настоящего времени.

Возвращаемые типы данных: двойное значение

Аргументы

Аргумент	Описание
date	Дата или метка времени 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**.

Следующие месяцы можно использовать в `first_month_of_year argument`:

значения first_month_of_year

Месяц	Значение
February	2
Март	3
Апрель	4
May	5
Июнь	6
Июль	7
Август	8
Сентябрь	9
Октябрь	10
November	11
Декабрь	12

Региональные настройки

Если не указано иное, в примерах, приведенных в данном разделе, используется следующий формат даты: ММ/ДД/ГГГГ. Формат даты указан в операторе `SET DateFormat` скрипта загрузки данных. В вашей системе может быть установлен другой формат даты по умолчанию в зависимости от региональных настроек и других факторов. Можно изменить формат в примерах в соответствии с потребностями. Или можно изменить форматы в скрипте загрузки в соответствии с этими примерами.

Региональные настройки по умолчанию в приложениях основаны на системных региональных настройках компьютера или сервера, где установлено ПО Qlik Sense. Если на сервере Qlik Sense, к которому обращается пользователь, выбран шведский язык, то редактор загрузки данных будет использовать шведские региональные настройки для даты, времени и валюты. Эти параметры регионального формата не связаны с языком, отображаемым в интерфейсе пользователя Qlik Sense. Qlik Sense будет отображаться на языке, который используется в браузере.

Примеры функции

Пример	Результат
`yearstart('10/19/2001')`	Возвращает 01/01/2001 00:00:00.
`yearstart('10/19/2001',-1)`	Возвращает 01/01/2000 00:00:00.
`yearstart('10/19/2001',0,4)`	Возвращает 04/01/2001 00:00:00.

Пример 1. Базовый пример

Скрипт загрузки и результаты

Обзор

Откройте редактор загрузки данных и добавьте приведенный ниже скрипт загрузки на новую вкладку.

Скрипт загрузки содержит следующее:

- Набор данных, содержащий набор транзакций с 2020 по 2022 год, загружается в таблицу под именем «Transactions».
- Поле даты было предоставлено в формате системной переменной DateFormat (MM/DD/YYYY).
- Предыдущий оператор load, который содержит следующее:
  - Функция yearstart(), заданная как поле year_start.
  - Функция Timestamp(), заданная как поле year_start_timestamp.

Скрипт загрузки

```
SET DateFormat='MM/DD/YYYY';

Transactions:
    Load
       *,
       yearstart(date) as year_start,
       timestamp(yearstart(date)) as year_start_timestamp
       ;
Load
*
Inline
[
id,date,amount
8188,01/13/2020,37.23
8189,02/26/2020,17.17
8190,03/27/2020,88.27
8191,04/16/2020,57.42
8192,05/21/2020,53.80
8193,08/14/2020,82.06
8194,10/07/2020,40.39
8195,12/05/2020,87.21
8196,01/22/2021,95.93
8197,02/03/2021,45.89
8198,03/17/2021,36.23
8199,04/23/2021,25.66
8200,05/04/2021,82.77
8201,06/30/2021,69.98
8202,07/26/2021,76.11
8203,12/27/2021,25.12
8204,06/06/2022,46.23
8205,07/18/2022,84.21
8206,11/14/2022,96.24
```

```
8207,12/12/2022,67.67
];
```

Результаты

Загрузите данные и откройте лист. Создайте новую таблицу и добавьте эти поля как измерения:

- id
- date
- year_start
- year_start_timestamp

Результирующая таблица

id	date	year_start	year_start_timestamp
8188	01/13/2020	01/01/2020	1/1/2020 12:00:00 AM
8189	02/26/2020	01/01/2020	1/1/2020 12:00:00 AM
8190	03/27/2020	01/01/2020	1/1/2020 12:00:00 AM
8191	04/16/2020	01/01/2020	1/1/2020 12:00:00 AM
8192	05/21/2020	01/01/2020	1/1/2020 12:00:00 AM
8193	08/14/2020	01/01/2020	1/1/2020 12:00:00 AM
8194	10/07/2020	01/01/2020	1/1/2020 12:00:00 AM
8195	12/05/2020	01/01/2020	1/1/2020 12:00:00 AM
8196	01/22/2021	01/01/2021	1/1/2021 12:00:00 AM
8197	02/03/2021	01/01/2021	1/1/2021 12:00:00 AM
8198	03/17/2021	01/01/2021	1/1/2021 12:00:00 AM
8199	04/23/2021	01/01/2021	1/1/2021 12:00:00 AM
8200	05/04/2021	01/01/2021	1/1/2021 12:00:00 AM
8201	06/30/2021	01/01/2021	1/1/2021 12:00:00 AM
8202	07/26/2021	01/01/2021	1/1/2021 12:00:00 AM
8203	12/27/2021	01/01/2021	1/1/2021 12:00:00 AM
8204	06/06/2022	01/01/2022	1/1/2022 12:00:00 AM
8205	07/18/2022	01/01/2022	1/1/2022 12:00:00 AM
8206	11/14/2022	01/01/2022	1/1/2022 12:00:00 AM
8207	12/12/2022	01/01/2022	1/1/2022 12:00:00 AM

Поле year_start создано предшествующим оператором load с использованием функции yearstart (), где в качестве аргумента функции передано поле даты.

Функция `yearstart()` первоначально определяет, к какому году относится значение даты, и возвращает метку времени для первой миллисекунды этого года.

Диаграмма функции `yearstart()` и транзакции 8199.

Транзакция 8199 совершена 23 апреля 2021 года. Функция `yearstart()` возвращает первую миллисекунду этого года, то есть 1 января в 00:00 (12:00:00 AM).

Пример 2. Скрипт period_no

Скрипт загрузки и результаты

Обзор

Используется тот же набор данных и сценарий, что в первом примере.

Однако в этом примере задача состоит в том, чтобы создать поле «`previous_year_start`», которое возвращает метку времени даты начала года, предшествующего тому, в котором совершена транзакция.

Скрипт загрузки

```
SET DateFormat='MM/DD/YYYY';

Transactions:
    Load
       *,
       yearstart(date,-1) as previous_year_start,
       timestamp(yearstart(date,-1)) as previous_year_start_timestamp
       ;
Load
*
Inline
[
id,date,amount
8188,01/13/2020,37.23
8189,02/26/2020,17.17
8190,03/27/2020,88.27
8191,04/16/2020,57.42
8192,05/21/2020,53.80
8193,08/14/2020,82.06
8194,10/07/2020,40.39
8195,12/05/2020,87.21
8196,01/22/2021,95.93
8197,02/03/2021,45.89
8198,03/17/2021,36.23
8199,04/23/2021,25.66
```

```
8200,05/04/2021,82.77
8201,06/30/2021,69.98
8202,07/26/2021,76.11
8203,12/27/2021,25.12
8204,06/06/2022,46.23
8205,07/18/2022,84.21
8206,11/14/2022,96.24
8207,12/12/2022,67.67
];
```

Результаты

Загрузите данные и откройте лист. Создайте новую таблицу и добавьте эти поля как измерения:

- id
- date
- previous_year_start
- previous_ year_start_timestamp

Результирующая таблица

id	date	previous_year_start	previous_year_start_timestamp
8188	01/13/2020	01/01/2019	1/1/2019 12:00:00 AM
8189	02/26/2020	01/01/2019	1/1/2019 12:00:00 AM
8190	03/27/2020	01/01/2019	1/1/2019 12:00:00 AM
8191	04/16/2020	01/01/2019	1/1/2019 12:00:00 AM
8192	05/21/2020	01/01/2019	1/1/2019 12:00:00 AM
8193	08/14/2020	01/01/2019	1/1/2019 12:00:00 AM
8194	10/07/2020	01/01/2019	1/1/2019 12:00:00 AM
8195	12/05/2020	01/01/2019	1/1/2019 12:00:00 AM
8196	01/22/2021	01/01/2020	1/1/2020 12:00:00 AM
8197	02/03/2021	01/01/2020	1/1/2020 12:00:00 AM
8198	03/17/2021	01/01/2020	1/1/2020 12:00:00 AM
8199	04/23/2021	01/01/2020	1/1/2020 12:00:00 AM
8200	05/04/2021	01/01/2020	1/1/2020 12:00:00 AM
8201	06/30/2021	01/01/2020	1/1/2020 12:00:00 AM
8202	07/26/2021	01/01/2020	1/1/2020 12:00:00 AM
8203	12/27/2021	01/01/2020	1/1/2020 12:00:00 AM
8204	06/06/2022	01/01/2021	1/1/2021 12:00:00 AM

id	date	previous_year_start	previous_year_start_timestamp
8205	07/18/2022	01/01/2021	1/1/2021 12:00:00 AM
8206	11/14/2022	01/01/2021	1/1/2021 12:00:00 AM
8207	12/12/2022	01/01/2021	1/1/2021 12:00:00 AM

Поскольку в данном примере в качестве аргумента смещения в функции `yearstart()` используется `period_no = -1`, функция сначала определяет год, в течение которого совершены транзакции. Затем она возвращается на год назад и определяет первую миллисекунду этого года.

Диаграмма функции `yearstart()` с `period_no = -1`.

Транзакция 8199 совершена 23 апреля 2021 года. Функция `yearstart()` возвращает первую миллисекунду предыдущего года, то есть 1 января 2020 года в 00:00 (12:00:00 AM), для поля `previous_year_start`.

Пример 3. Аргумент first_month_of_year

Скрипт загрузки и результаты

Обзор

Используется тот же набор данных и сценарий, что в первом примере.

Однако в этом примере политика компании такова, что год начинается 1 апреля.

Скрипт загрузки

```
SET DateFormat='MM/DD/YYYY';

Transactions:
    Load
       *,
       yearstart(date,0,4) as year_start,
       timestamp(yearstart(date,0,4)) as year_start_timestamp
       ;
Load
*
Inline
[
id,date,amount
8188,01/13/2020,37.23
8189,02/26/2020,17.17
```

```
8190,03/27/2020,88.27
8191,04/16/2020,57.42
8192,05/21/2020,53.80
8193,08/14/2020,82.06
8194,10/07/2020,40.39
8195,12/05/2020,87.21
8196,01/22/2021,95.93
8197,02/03/2021,45.89
8198,03/17/2021,36.23
8199,04/23/2021,25.66
8200,05/04/2021,82.77
8201,06/30/2021,69.98
8202,07/26/2021,76.11
8203,12/27/2021,25.12
8204,06/06/2022,46.23
8205,07/18/2022,84.21
8206,11/14/2022,96.24
8207,12/12/2022,67.67
];
```

Результаты

Загрузите данные и откройте лист. Создайте новую таблицу и добавьте эти поля как измерения:

- id
- date
- year_start
- year_start_timestamp

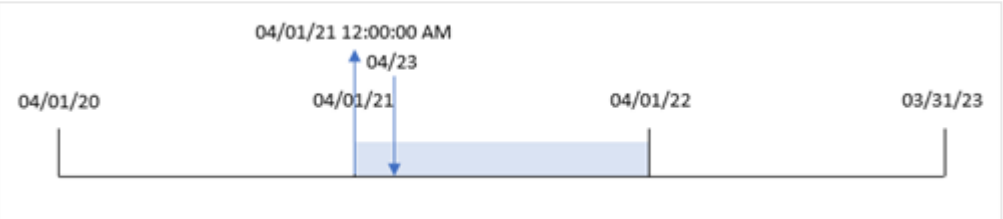
Результирующая таблица

id	date	year_start	year_start_timestamp
8188	01/13/2020	04/01/2019	4/1/2019 12:00:00 AM
8189	02/26/2020	04/01/2019	4/1/2019 12:00:00 AM
8190	03/27/2020	04/01/2019	4/1/2019 12:00:00 AM
8191	04/16/2020	04/01/2020	4/1/2020 12:00:00 AM
8192	05/21/2020	04/01/2020	4/1/2020 12:00:00 AM
8193	08/14/2020	04/01/2020	4/1/2020 12:00:00 AM
8194	10/07/2020	04/01/2020	4/1/2020 12:00:00 AM
8195	12/05/2020	04/01/2020	4/1/2020 12:00:00 AM
8196	01/22/2021	04/01/2020	4/1/2020 12:00:00 AM
8197	02/03/2021	04/01/2020	4/1/2020 12:00:00 AM
8198	03/17/2021	04/01/2020	4/1/2020 12:00:00 AM
8199	04/23/2021	04/01/2021	4/1/2021 12:00:00 AM

id	date	year_start	year_start_timestamp
8200	05/04/2021	04/01/2021	4/1/2021 12:00:00 AM
8201	06/30/2021	04/01/2021	4/1/2021 12:00:00 AM
8202	07/26/2021	04/01/2021	4/1/2021 12:00:00 AM
8203	12/27/2021	04/01/2021	4/1/2021 12:00:00 AM
8204	06/06/2022	04/01/2022	4/1/2022 12:00:00 AM
8205	07/18/2022	04/01/2022	4/1/2022 12:00:00 AM
8206	11/14/2022	04/01/2022	4/1/2022 12:00:00 AM
8207	12/12/2022	04/01/2022	4/1/2022 12:00:00 AM

Поскольку в этом примере аргумент `first_month_of_year = 4` используется в функции `yearstart()`, в качестве первого дня года задается 1 апреля, а в качестве последнего дня года – 31 марта.

Диаграмма функции `yearstart()`, где в качестве первого месяца года задан апрель.



Транзакция 8199 совершена 23 апреля 2021 года. Поскольку функция `yearstart()` задает в качестве начала года 1 апреля, она возвращает эту дату в качестве значения `year_start` для транзакции.

Пример 4. Пример объекта диаграммы

Скрипт загрузки и выражение диаграммы

Обзор

Используется тот же набор данных и сценарий, что в первом примере.

Однако в этом примере в приложение загружается неизмененный набор данных. Расчет, который возвращает метку даты начала года, в котором совершены транзакции, создается как мера в объекте диаграммы в приложении.

Скрипт загрузки

```
Transactions:
Load
*
Inline
[
id,date,amount
```

```
8188,01/13/2020,37.23
8189,02/26/2020,17.17
8190,03/27/2020,88.27
8191,04/16/2020,57.42
8192,05/21/2020,53.80
8193,08/14/2020,82.06
8194,10/07/2020,40.39
8195,12/05/2020,87.21
8196,01/22/2021,95.93
8197,02/03/2021,45.89
8198,03/17/2021,36.23
8199,04/23/2021,25.66
8200,05/04/2021,82.77
8201,06/30/2021,69.98
8202,07/26/2021,76.11
8203,12/27/2021,25.12
8204,06/06/2022,46.23
8205,07/18/2022,84.21
8206,11/14/2022,96.24
8207,12/12/2022,67.67
];
```

Результаты

Загрузите данные и откройте лист. Создайте новую таблицу и добавьте эти поля как измерения:

- `id`
- `date`

Чтобы рассчитать год, в котором совершена транзакция, создайте следующие меры:

- `=yearstart(date)`
- `=timestamp(yearstart(date))`

Результирующая таблица

id	date	=yearstart(date)	=timestamp(yearstart(date))
8188	06/06/2022	01/01/2022	1/1/2022 12:00:00 AM
8189	07/18/2022	01/01/2022	1/1/2022 12:00:00 AM
8190	11/14/2022	01/01/2022	1/1/2022 12:00:00 AM
8191	12/12/2022	01/01/2022	1/1/2022 12:00:00 AM
8192	01/22/2021	01/01/2021	1/1/2021 12:00:00 AM
8193	02/03/2021	01/01/2021	1/1/2021 12:00:00 AM
8194	03/17/2021	01/01/2021	1/1/2021 12:00:00 AM
8195	04/23/2021	01/01/2021	1/1/2021 12:00:00 AM
8196	05/04/2021	01/01/2021	1/1/2021 12:00:00 AM

id	date	=yearstart(date)	=timestamp(yearstart(date))
8197	06/30/2021	01/01/2021	1/1/2021 12:00:00 AM
8198	07/26/2021	01/01/2021	1/1/2021 12:00:00 AM
8199	12/27/2021	01/01/2021	1/1/2021 12:00:00 AM
8200	01/13/2020	01/01/2020	1/1/2020 12:00:00 AM
8201	02/26/2020	01/01/2020	1/1/2020 12:00:00 AM
8202	03/27/2020	01/01/2020	1/1/2020 12:00:00 AM
8203	04/16/2020	01/01/2020	1/1/2020 12:00:00 AM
8204	05/21/2020	01/01/2020	1/1/2020 12:00:00 AM
8205	08/14/2020	01/01/2020	1/1/2020 12:00:00 AM
8206	10/07/2020	01/01/2020	1/1/2020 12:00:00 AM
8207	12/05/2020	01/01/2020	1/1/2020 12:00:00 AM

Мера `start_of_year` создана в объекте диаграммы с использованием функции `yearstart()`, где в качестве аргумента функции передано поле даты.

Функция `yearstart()` первоначально определяет, к какому году относится значение даты, и возвращает метку времени для первой миллисекунды этого года.

Диаграмма функции `yearstart()` и транзакции 8199.

Транзакция 8199 совершена 23 апреля 2021 года. Функция `yearstart()` возвращает первую миллисекунду этого года, то есть 1 января в 00:00 (12:00:00 AM).

Пример 5. Сценарий

Скрипт загрузки и результаты

Обзор

Откройте редактор загрузки данных и добавьте приведенный ниже скрипт загрузки на новую вкладку.

Скрипт загрузки содержит следующее:

- Набор данных загружается в таблицу под именем `Loans`. Данная таблица содержит следующие поля:

- Идентификаторы кредита.
- Остаток на начало года.
- Простая процентная ставка, начисляемая по каждому кредиту в год.

Конечному пользователю требуется объект диаграммы, который будет отображать по идентификатору кредита текущий процент, начисленный по каждому кредиту за год до текущей даты.

Скрипт загрузки

```
Loans:
Load
*
Inline
[
loan_id,start_balance,rate
8188,$10000.00,0.024
8189,$15000.00,0.057
8190,$17500.00,0.024
8191,$21000.00,0.034
8192,$90000.00,0.084
];
```

Результаты

Загрузите данные и откройте лист. Создайте новую таблицу и добавьте эти поля как измерения:

- `loan_id`
- `start_balance`

Чтобы рассчитать накопленный процент, создайте следующую меру:

```
=start_balance*(rate*(today(1)-yearstart(today(1)))/365)
```

Задайте параметру **Формат чисел** меры значение **Денежный**.

Результирующая таблица

loan_id	start_balance	=start_balance*(rate*(today(1)-yearstart(today(1)))/365)
8188	$10000.00	$39.73
8189	$15000.00	$339.66
8190	$17500.00	$166.85
8191	$21000.00	$283.64
8192	$90000.00	$3003.29

Используя сегодняшнюю дату в качестве единственного аргумента, функция `yearstart()` возвращает дату начала текущего года. Вычитая этот результат из текущей даты, выражение возвращает количество дней, прошедших до сих пор в этом году.

Затем это значение умножается на процентную ставку и делится на 365, чтобы получить эффективную процентную ставку за период. После этого эффективная процентная ставка за период умножается на начальный остаток кредита, чтобы вернуть проценты, начисленные до сих пор в этом году.

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.

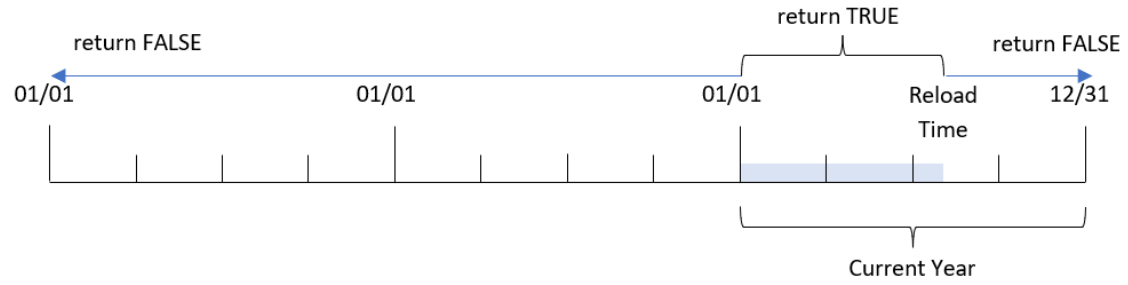
Синтаксис:

```
YearToDate(timestamp[ , yearoffset [ , firstmonth [ , todaydate] ] ])
```

Возвращаемые типы данных: Булево значение

В Qlik Sense логическое значение «истина» представлено как -1, а «ложь» – как 0.

Диаграмма с примером функции `yeartodate()`



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.

Другими словами, функция `yeartodate()` без дополнительных параметров используется для проверки метки времени и возвращает булево значение на основе того, произошла ли дата в календарном году до даты перезагрузки включительно.

Однако функция также может заменять начальную дату года с помощью аргумента `firstmonth` и, кроме того, выполнять сравнения с предшествующими или последующими годами с помощью аргумента `yearoffset`.

В заключение при использовании исторических наборов данных, функция `yeartodate()` передает параметр для настройки `todaydate`, в результате чего выполняется сравнение метки времени с календарным годом до даты, предоставленной в аргументе `todaydate`, включительно.

Аргументы

Аргумент	Описание
timestamp	Метка времени для проверки, например 10/12/2012.
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. Значение 1 будет указывать начало финансового года с 1 января, а значение 12 – с 1 декабря.
todaydate	Задав значение **todaydate** (метка времени последнего выполнения скрипта, если не указано), можно сместить день, используемый в качестве верхней границы периода.

Когда это следует использовать

Функция `yeartodate()` возвращает результат в виде логического значения. Обычно этот тип функции используется в качестве условия в выражении IF. Это возвращает агрегирование или расчет в зависимости от того, попадает ли проверяемая дата в год до рассматриваемой даты перезагрузки приложения включительно.

Например, функцию YearToDate() можно использовать для идентификации всего оборудования, изготовленного в текущем году до текущей даты.

В следующих примерах предполагается время последней перезагрузки = 11/18/2011.

Примеры функции

Пример	Результат
`yeartodate( '11/18/2010')`	возвращает False
`yeartodate( '02/01/2011')`	возвращает True
`yeartodate( '11/18/2011')`	возвращает True
`yeartodate( '11/19/2011')`	возвращает False
`yeartodate( '11/19/2011', 0, 1, '12/31/2011')`	возвращает True
`yeartodate( '11/18/2010', -1)`	возвращает True
`yeartodate( '11/18/2011', -1)`	возвращает False
`yeartodate( '04/30/2011', 0, 5)`	возвращает False
`yeartodate( '05/01/2011', 0, 5)`	возвращает True

Региональные настройки

Если не указано иное, в примерах, приведенных в данном разделе, используется следующий формат даты: ММ/ДД/ГГГГ. Формат даты указан в операторе `SET DateFormat` скрипта загрузки данных. В вашей системе может быть установлен другой формат даты по умолчанию в зависимости от региональных настроек и других факторов. Можно изменить формат в примерах в соответствии с потребностями. Или можно изменить форматы в скрипте загрузки в соответствии с этими примерами.

Региональные настройки по умолчанию в приложениях основаны на системных региональных настройках компьютера или сервера, где установлено ПО Qlik Sense. Если на сервере Qlik Sense, к которому обращается пользователь, выбран шведский язык, то редактор загрузки данных будет использовать шведские региональные настройки для даты, времени и валюты. Эти параметры регионального формата не связаны с языком, отображаемым в интерфейсе пользователя Qlik Sense. Qlik Sense будет отображаться на языке, который используется в браузере.

Пример 1. Базовый пример

Скрипт загрузки и результаты

Обзор

Откройте редактор загрузки данных и добавьте приведенный ниже скрипт загрузки на новую вкладку.

Скрипт загрузки содержит следующее:

- Набор данных, содержащий набор транзакций с 2020 по 2022 год, загружается в таблицу под именем `Transactions`.
- Поле даты было предоставлено в формате системной переменной `DateFormat` (MM/DD/YYYY).
- Создание поля `year_to_date`, которое определяет, какие транзакции совершены в течение календарного года до даты последней перезагрузки.

Текущая дата на момент создания – 26 апреля 2022 года.

Скрипт загрузки

```
SET DateFormat='MM/DD/YYYY';

Transactions:
    Load
        *,
        yeartodate(date) as year_to_date
        ;
Load
*
Inline
[
id,date,amount
```

```
8188,01/10/2020,37.23
8189,02/28/2020,17.17
8190,04/09/2020,88.27
8191,04/16/2020,57.42
8192,05/21/2020,53.80
8193,08/14/2020,82.06
8194,10/07/2020,40.39
8195,12/05/2020,87.21
8196,01/22/2021,95.93
8197,02/03/2021,45.89
8198,03/17/2021,36.23
8199,04/23/2021,25.66
8200,05/04/2021,82.77
8201,06/30/2021,69.98
8202,07/26/2021,76.11
8203,12/27/2021,25.12
8204,02/02/2022,46.23
8205,02/26/2022,84.21
8206,03/07/2022,96.24
8207,03/11/2022,67.67
];
```

Результаты

Загрузите данные и откройте лист. Создайте новую таблицу и добавьте эти поля как измерения:

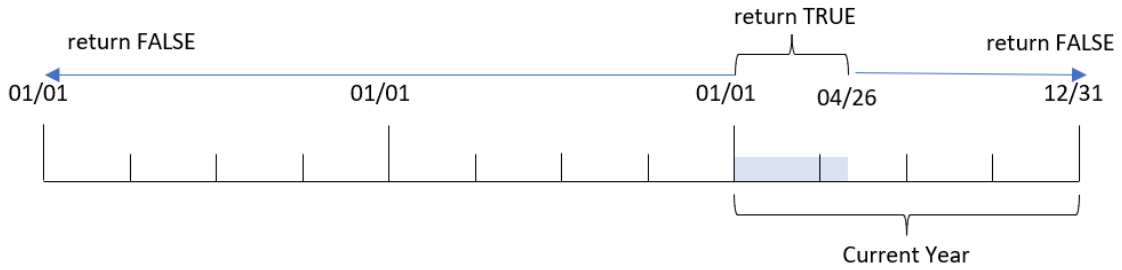
- `date`
- `year_to_date`

Результирующая таблица

date	year_to_date
01/10/2020	0
02/28/2020	0
04/09/2020	0
04/16/2020	0
05/21/2020	0
08/14/2020	0
10/07/2020	0
12/05/2020	0
01/22/2021	0
02/03/2021	0
03/17/2021	0
04/23/2021	0

date	year_to_date
05/04/2021	0
06/30/2021	0
07/26/2021	0
12/27/2021	0
02/02/2022	-1
02/26/2022	-1
03/07/2022	-1
03/11/2022	-1

Диаграмма функции `yeartodate()`, базовый пример



Поле «`year_to_date`» создано предшествующим оператором load с использованием функции `yeartodate()`, где в качестве аргумента функции передано поле `date`.

Так как функции `yeartodate()` не переданы дополнительные параметры, она сначала определяет дату перезагрузки и границы текущего календарного года (начало 1 января), которые будут возвращать булево значение `TRUE`.

Таким образом, любая транзакция, совершенная в период с 1 января по 26 апреля (дата перезагрузки), возвращает булев результат `TRUE`. Любая транзакция, совершенная до начала 2022 года, будет возвращать булев результат `FALSE`.

Пример 2. Аргумент yearoffset

Скрипт загрузки и результаты

Обзор

Откройте редактор загрузки данных и добавьте приведенный ниже скрипт загрузки на новую вкладку.

Скрипт загрузки содержит следующее:

- Используется тот же набор данных и сценарий, что в первом примере.
- Создание поля `two_years_prior`, определяющего, какие транзакции совершены на два полных года раньше сегмента текущего календарного года до рассматриваемой даты.

Скрипт загрузки

```
SET DateFormat='MM/DD/YYYY';

Transactions:
    Load
        *,
        yeartodate(date,-2) as two_years_prior
        ;
Load
*
Inline
[
id,date,amount
8188,01/10/2020,37.23
8189,02/28/2020,17.17
8190,04/09/2020,88.27
8191,04/16/2020,57.42
8192,05/21/2020,53.80
8193,08/14/2020,82.06
8194,10/07/2020,40.39
8195,12/05/2020,87.21
8196,01/22/2021,95.93
8197,02/03/2021,45.89
8198,03/17/2021,36.23
8199,04/23/2021,25.66
8200,05/04/2021,82.77
8201,06/30/2021,69.98
8202,07/26/2021,76.11
8203,12/27/2021,25.12
8204,02/02/2022,46.23
8205,02/26/2022,84.21
8206,03/07/2022,96.24
8207,03/11/2022,67.67
];
```

Результаты

Загрузите данные и откройте лист. Создайте новую таблицу и добавьте эти поля как измерения:

- `date`
- `two_years_prior`

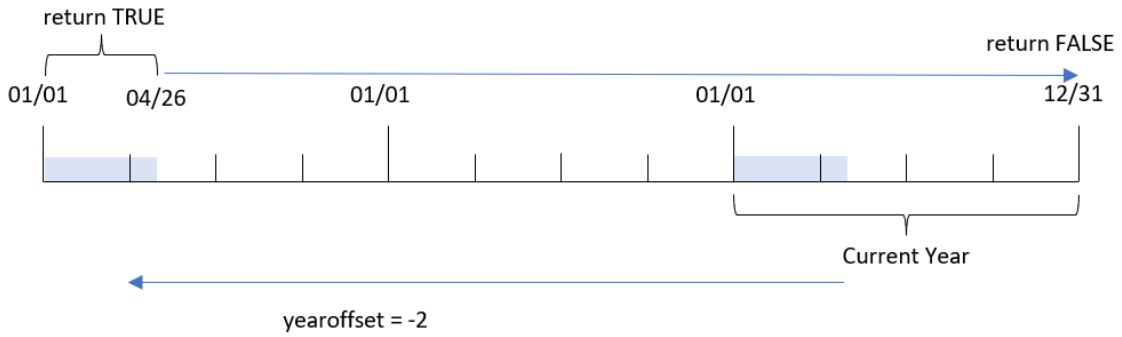
Результирующая таблица

date	two_years_prior
01/10/2020	-1

date	two_years_prior
02/28/2020	-1
04/09/2020	-1
04/16/2020	-1
05/21/2020	0
08/14/2020	0
10/07/2020	0
12/05/2020	0
01/22/2021	0
02/03/2021	0
03/17/2021	0
04/23/2021	0
05/04/2021	0
06/30/2021	0
07/26/2021	0
12/27/2021	0
02/02/2022	0
02/26/2022	0
03/07/2022	0
03/11/2022	0

Используя аргумент `yearoffset` = -2, функция `yeartodate()` сдвигает границы сравниваемого сегмента календарного года на два полных года. Первоначально сегмент года начинался 1 января и заканчивался 26 апреля 2022 года. Затем аргумент `yearoffset` сдвигает этот сегмент на два года назад. Таким образом границы сегмента переносятся на 1 января и 26 апреля 2020 года.

Диаграмма функции `yeartodate()`, пример с аргументом yearoffset



Таким образом, любая транзакция, совершенная в период с 1 января по 26 апреля 2020 года, возвращает булев результат `TRUE`. Все транзакции, совершенные до или после этого сегмента, возвращают `FALSE`.

Пример 3. Аргумент firstmonth

Скрипт загрузки и результаты

Обзор

Откройте редактор загрузки данных и добавьте приведенный ниже скрипт загрузки на новую вкладку.

Скрипт загрузки содержит следующее:

- Используется тот же набор данных и сценарий, что в первом примере.
- Создание поля `year_to_date`, которое определяет, какие транзакции совершены в течение календарного года до даты последней перезагрузки.

В данном примере в качестве начала финансового года задано 1 июля.

Скрипт загрузки

```
SET DateFormat='MM/DD/YYYY';

Transactions:
    Load
        *,
        yeartodate(date,0,7) as year_to_date
        ;
Load
*
Inline
[
id,date,amount
8188,01/10/2020,37.23
8189,02/28/2020,17.17
```

```
8190,04/09/2020,88.27
8191,04/16/2020,57.42
8192,05/21/2020,53.80
8193,08/14/2020,82.06
8194,10/07/2020,40.39
8195,12/05/2020,87.21
8196,01/22/2021,95.93
8197,02/03/2021,45.89
8198,03/17/2021,36.23
8199,04/23/2021,25.66
8200,05/04/2021,82.77
8201,06/30/2021,69.98
8202,07/26/2021,76.11
8203,12/27/2021,25.12
8204,02/02/2022,46.23
8205,02/26/2022,84.21
8206,03/07/2022,96.24
8207,03/11/2022,67.67
];
```

Результаты

Загрузите данные и откройте лист. Создайте новую таблицу и добавьте эти поля как измерения:

- `date`
- `year_to_date`

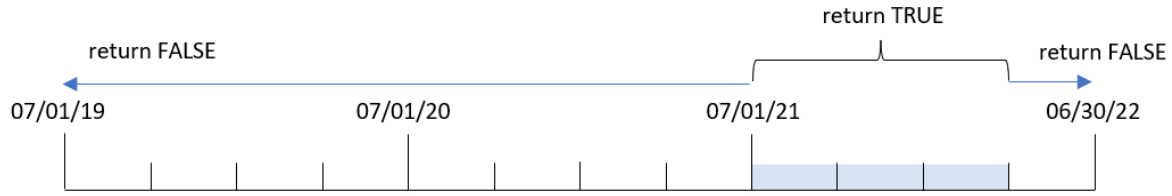
Результирующая таблица

date	year_to_date
01/10/2020	0
02/28/2020	0
04/09/2020	0
04/16/2020	0
05/21/2020	0
08/14/2020	0
10/07/2020	0
12/05/2020	0
01/22/2021	0
02/03/2021	0
03/17/2021	0
04/23/2021	0
05/04/2021	0

date	year_to_date
06/30/2021	0
07/26/2021	-1
12/27/2021	-1
02/02/2022	-1
02/26/2022	-1
03/07/2022	-1
03/11/2022	-1

Поскольку в этом примере используется аргумент `firstmonth` = 7, функция `yeartodate()` задает в качестве первого дня года 1 июля, а в качестве последнего дня года – 30 июня.

Диаграмма функции `yeartodate()`, пример с аргументом firstmonth



Таким образом, любая транзакция, совершенная в период с 1 июля 2021 года по 26 апреля 2022 года (дата перезагрузки), возвращает булев результат `TRUE`. Любая транзакция, совершенная до 1 июля 2021 года, будет возвращать булев результат `FALSE`.

Пример 4. Аргумент todaydate

Скрипт загрузки и результаты

Обзор

Откройте редактор загрузки данных и добавьте приведенный ниже скрипт загрузки на новую вкладку.

Скрипт загрузки содержит следующее:

- Используется тот же набор данных и сценарий, что в первом примере.
- Создание поля `year_to_date`, которое определяет, какие транзакции совершены в течение календарного года до даты последней перезагрузки.

Однако в этом примере требуется идентифицировать все транзакции, совершенные в течение календарного года до 1 марта 2022 года включительно.

Скрипт загрузки

```
SET DateFormat='MM/DD/YYYY';

Transactions:
    Load
        *,
        yeartodate(date, 0, 1, '03/01/2022') as year_to_date
;
Load
*
Inline
[
id,date,amount
8188,01/10/2020,37.23
8189,02/28/2020,17.17
8190,04/09/2020,88.27
8191,04/16/2020,57.42
8192,05/21/2020,53.80
8193,08/14/2020,82.06
8194,10/07/2020,40.39
8195,12/05/2020,87.21
8196,01/22/2021,95.93
8197,02/03/2021,45.89
8198,03/17/2021,36.23
8199,04/23/2021,25.66
8200,05/04/2021,82.77
8201,06/30/2021,69.98
8202,07/26/2021,76.11
8203,12/27/2021,25.12
8204,02/02/2022,46.23
8205,02/26/2022,84.21
8206,03/07/2022,96.24
8207,03/11/2022,67.67
];
```

Результаты

Загрузите данные и откройте лист. Создайте новую таблицу и добавьте эти поля как измерения:

- `date`
- `year_to_date`

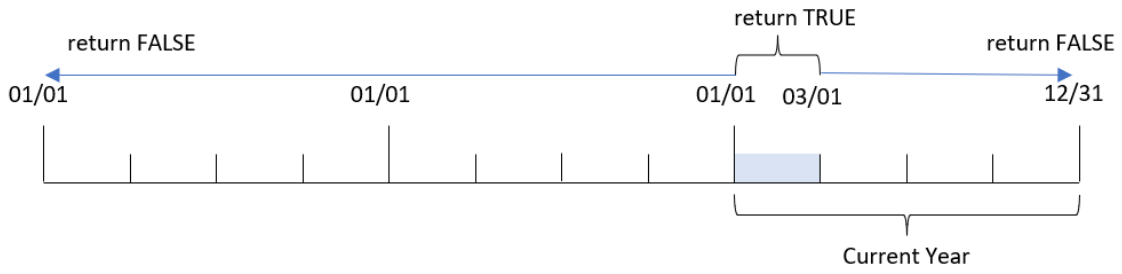
Результирующая таблица

date	year_to_date
01/10/2020	0
02/28/2020	0
04/09/2020	0
04/16/2020	0

date	year_to_date
05/21/2020	0
08/14/2020	0
10/07/2020	0
12/05/2020	0
01/22/2021	0
02/03/2021	0
03/17/2021	0
04/23/2021	0
05/04/2021	0
06/30/2021	0
07/26/2021	0
12/27/2021	0
02/02/2022	-1
02/26/2022	-1
03/07/2022	0
03/11/2022	0

В этом примере, так как используется аргумент `todaydate` = 03/01/2022, функция `yeartodate()` задает в качестве окончания сравниваемого сегмента календарного года 1 марта 2022 года. Критически важно предоставить параметр `firstmonth` (от 1 до 12); в противном случае функция вернет нулевые результаты (null).

Диаграмма функции `yeartodate()`, пример с аргументом todaydate



Таким образом, любая транзакция, совершенная в период с 1 января по 1 марта 2022 года (параметр `todaydate`), возвращает булев результат `TRUE`. Любая транзакция, совершенная до 1 января 2022 года или после 1 марта 2022 года, будет возвращать булев результат `FALSE`.

Пример 5. Пример объекта диаграммы

Скрипт загрузки и выражение диаграммы

Обзор

Откройте редактор загрузки данных и добавьте приведенный ниже скрипт загрузки на новую вкладку.

Скрипт загрузки содержит тот же набор данных и сценарий, что в первом примере.

Однако в этом примере в приложение загружается неизмененный набор данных. Расчет, который определяет, какие транзакции совершены в календарном году до текущей даты, создается в качестве меры в объекте диаграммы в приложении.

Скрипт загрузки

```
Transactions:
Load
*
Inline
[
id,date,amount
8188,01/10/2020,37.23
8189,02/28/2020,17.17
8190,04/09/2020,88.27
8191,04/16/2020,57.42
8192,05/21/2020,53.80
8193,08/14/2020,82.06
8194,10/07/2020,40.39
8195,12/05/2020,87.21
8196,01/22/2021,95.93
8197,02/03/2021,45.89
8198,03/17/2021,36.23
8199,04/23/2021,25.66
8200,05/04/2021,82.77
8201,06/30/2021,69.98
8202,07/26/2021,76.11
8203,12/27/2021,25.12
8204,02/02/2022,46.23
8205,02/26/2022,84.21
8206,03/07/2022,96.24
8207,03/11/2022,67.67
];
```

Результаты

Загрузите данные и откройте лист. Создайте новую таблицу и добавьте это поле как измерение: `date`.

Добавьте следующую меру:

```
=yeartodate(date)
```

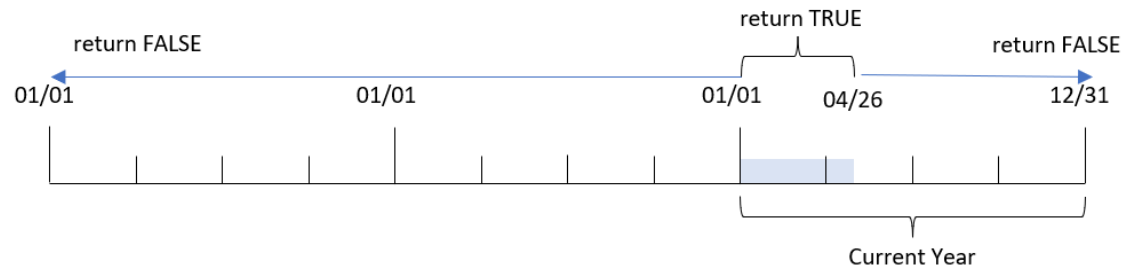
Результирующая таблица

date	=yeartodate(date)
01/10/2020	0
02/28/2020	0
04/09/2020	0
04/16/2020	0
05/21/2020	0
08/14/2020	0
10/07/2020	0
12/05/2020	0
01/22/2021	0
02/03/2021	0
03/17/2021	0
04/23/2021	0
05/04/2021	0
06/30/2021	0
07/26/2021	0
12/27/2021	0
02/02/2022	-1
02/26/2022	-1
03/07/2022	-1
03/11/2022	-1

Мера `year_to_date` создается в объекте диаграммы с использованием функции `yeartodate()`, где в качестве аргумента функции передано поле `date`.

Так как функции `yeartodate()` не переданы дополнительные параметры, она сначала определяет дату перезагрузки и границы текущего календарного года (начало 1 января), которые будут возвращать булево значение `TRUE`.

Диаграмма функции `yeartodate()`, пример с объектом диаграммы



Любая транзакция, совершенная в период с 1 января по 26 апреля (дата перезагрузки), возвращает булев результат `TRUE`. Любая транзакция, совершенная до начала 2022 года, будет возвращать булев результат `FALSE`.

Пример 6. Сценарий

Скрипт загрузки и выражение диаграммы

Обзор

Откройте редактор загрузки данных и добавьте приведенный ниже скрипт загрузки на новую вкладку.

Скрипт загрузки содержит следующее:

- Набор данных, содержащий набор транзакций с 2020 по 2022 год, загружается в таблицу под именем `Transactions`.
- Поле даты было предоставлено в формате системной переменной `DateFormat` (MM/DD/YYYY).

Конечному пользователю требуется объект ключевого показателя эффективности (КПЭ), который представляет общие продажи за период 2021 года, эквивалентный периоду текущего года до текущей даты по состоянию на время последней перезагрузки.

Текущая дата на момент создания — 16 июня 2022 года.

Скрипт загрузки

```
SET DateFormat='MM/DD/YYYY';

Transactions:
Load
*
Inline
[
id,date,amount
8188,01/10/2020,37.23
8189,02/28/2020,17.17
8190,04/09/2020,88.27
```

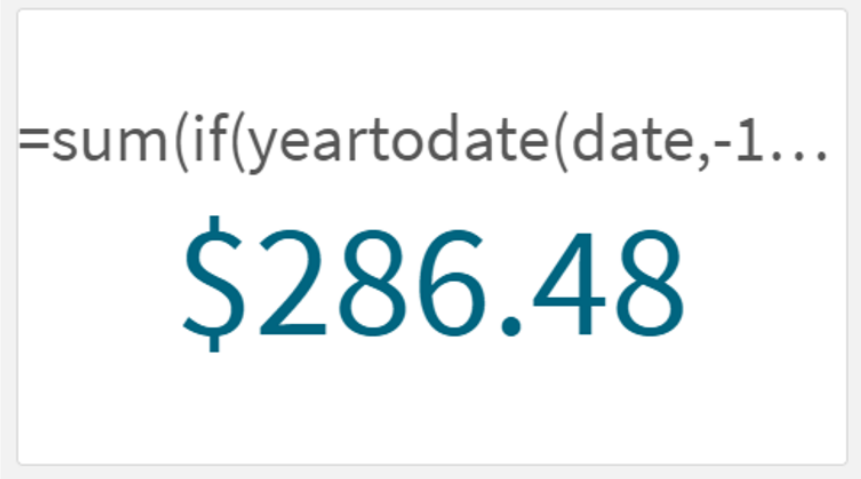
```
8191,04/16/2020,57.42
8192,05/21/2020,53.80
8193,08/14/2020,82.06
8194,10/07/2020,40.39
8195,12/05/2020,87.21
8196,01/22/2021,95.93
8197,02/03/2021,45.89
8198,03/17/2021,36.23
8199,04/23/2021,25.66
8200,05/04/2021,82.77
8201,06/30/2021,69.98
8202,07/26/2021,76.11
8203,12/27/2021,25.12
8204,02/02/2022,46.23
8205,02/26/2022,84.21
8206,03/07/2022,96.24
8207,03/11/2022,67.67
];
```

Результаты

Выполните следующие действия.

1. Создайте объект ключевого показателя эффективности.
2. Чтобы рассчитать общий объем продаж, создайте следующую меру агрегирования: `=sum(if(yeartodate(date,-1),amount,0))`
3. Задайте параметру **Формат чисел** меры значение **Денежный**.

Диаграмма КПЭ `yeartodate()` за 2021 год



Функция `yeartodate()` возвращает булево значение при проверке дат для каждого идентификатора транзакции. Так как перезагрузка выполнена 16 июня 2022 года, функция `yeartodate` рассматривает сегмент года с 01/01/2022 по 06/16/2022. Однако так как в функции использовался

аргумент `period_no` = -1, эти границы затем переносятся на год назад. Поэтому для любой транзакции, совершенной с 01/01/2021 по 06/16/2021, функция `yeartodate()` возвращает булево значение `TRUE` и суммирует количество.

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 1210)(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**.
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**.
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.

Аргументы	Описание
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)
2018/08/3	049681	13	transaction_date	transaction_date	customer_id

transaction_date	customer_id	transaction_quantity	=GetObjectDimension ()	=GetObjectDimension (0)	=GetObjectDimension (1)
0					
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**.

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 в формате UTC для даты и времени последнего изменения 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 заголовка XML из файла QVD при его наличии, в противном случае она возвращает значение NULL.Время в метке времени указано в формате UTC.

```
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 1493)

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 в формате UTC для даты и времени последнего изменения 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 заголовка XML из файла QVD при его наличии, в противном случае она возвращает значение NULL.Время в метке времени указано в формате UTC.

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. По умолчанию результат будет отформатирован как валюта..

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. По умолчанию результат будет отформатирован как валюта..

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. По умолчанию результат будет отформатирован как валюта..

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:

p *Функции интерпретации (page 1282)*

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 166)*

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 1284)(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 1285)(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 1286)(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 1287)(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 1288)(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 1290)(text[, format])
```

См. также:

p *Функции форматирования (page 1245)*

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 таблицах не допускается, если в каком-либо из выражений диаграммы используется функция диаграммы между записями. Данные возможности сортировки автоматически отключаются. Когда используется функция диаграммы между записями в визуализации или таблице, сортировка визуализации сбрасывается до сортированных входных данных функции между записями. Это ограничение не распространяется на эквивалентную функцию скрипта, если такая есть.

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 1330)(expr)
```

См. также:

p *Функции над выборкой (page 1351)*

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 1351)*
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.
- Сортировка по y-значениям в диаграммах или сортировка по столбцам выражений в таблицах не допускается, если в каком-либо из выражений диаграммы используется эта функция диаграммы. Данные возможности сортировки автоматически отключаются. Когда используется эта функция диаграммы в визуализации или таблице, сортировка визуализации сбрасывается до сортированных входных данных этой функции.

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:

p *Below – функция диаграммы (page 1300)*
p *Bottom – функция диаграммы (page 1304)*
p *Top – функция диаграммы (page 1331)*
p *RangeAvg (page 1354)*

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.
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 1351)*
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.
- Сортировка по y-значениям в диаграммах или сортировка по столбцам выражений в таблицах не допускается, если в каком-либо из выражений диаграммы используется эта функция диаграммы. Данные возможности сортировки автоматически отключаются. Когда используется эта функция диаграммы в визуализации или таблице, сортировка визуализации сбрасывается до сортированных входных данных этой функции.

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:

p *Above – функция диаграммы (page 1295)*
p *Bottom – функция диаграммы (page 1304)*
p *Top – функция диаграммы (page 1331)*
p *RangeAvg (page 1354)*

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 1351)*
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.
- Сортировка по y-значениям в диаграммах или сортировка по столбцам выражений в таблицах не допускается, если в каком-либо из выражений диаграммы используется эта функция диаграммы. Данные возможности сортировки автоматически отключаются. Когда используется эта функция диаграммы в визуализации или таблице, сортировка визуализации сбрасывается до сортированных входных данных этой функции.

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:

p *Top – функция диаграммы (page 1331)*

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:

- Рекурсивные вызовы возвращают значение NULL.
- Если элемент **ColumnNo** ссылается на столбец, для которого нет мер, возвращается значение NULL.

- Сортировка по y-значениям в диаграммах или сортировка по столбцам выражений в таблицах не допускается, если в каком-либо из выражений диаграммы используется эта функция диаграммы. Данные возможности сортировки автоматически отключаются. Когда используется эта функция диаграммы в визуализации или таблице, сортировка визуализации сбрасывается до сортированных входных данных этой функции.

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
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 1309)*.
Выполните выборку Customer A.	Выборка изменяет элемент Total Sales Value и, следовательно, элемент %Sales. См. пример *Процентное соотношение продаж для выбранного клиента (page 1309)*.

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.

Сортировка по y-значениям в диаграммах или сортировка по столбцам выражений в таблицах не допускается, если в каком-либо из выражений диаграммы используется эта функция диаграммы. Данные возможности сортировки автоматически отключаются. Когда используется эта функция диаграммы в визуализации или таблице, сортировка визуализации сбрасывается до сортированных входных данных этой функции.

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 ',');
```

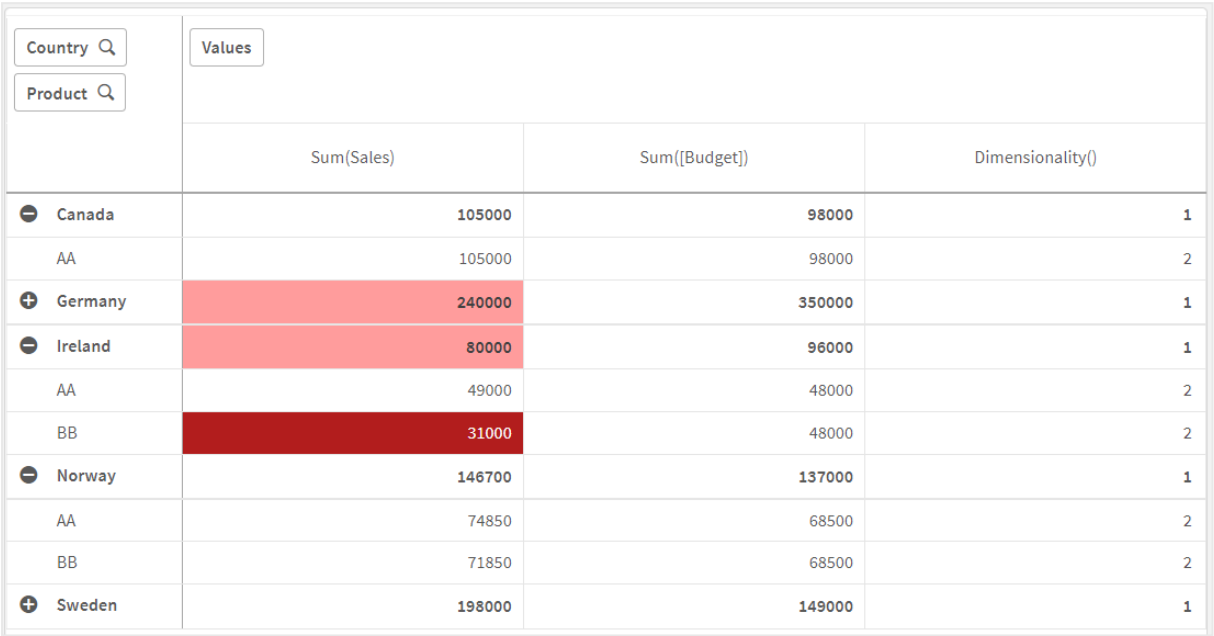
Выражение диаграммы

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.
- Сортировка по y-значениям в диаграммах или сортировка по столбцам выражений в таблицах не допускается, если в каком-либо из выражений диаграммы используется эта функция диаграммы. Данные возможности сортировки автоматически отключаются. Когда используется эта функция диаграммы в визуализации или таблице, сортировка визуализации сбрасывается до сортированных входных данных этой функции. Это ограничение не распространяется на эквивалентную функцию скрипта, если такая есть.

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.
- Сортировка по y-значениям в диаграммах или сортировка по столбцам выражений в таблицах не допускается, если в каком-либо из выражений диаграммы используется эта функция диаграммы. Данные возможности сортировки автоматически отключаются. Когда используется эта функция диаграммы в визуализации или таблице, сортировка визуализации сбрасывается до сортированных входных данных этой функции. Это ограничение не распространяется на эквивалентную функцию скрипта, если такая есть.

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|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 FieldValue('First name',1) as MyPos1
Resident 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.

Сортировка по y-значениям в диаграммах или сортировка по столбцам выражений в таблицах не допускается, если в каком-либо из выражений диаграммы используется эта функция диаграммы. Данные возможности сортировки автоматически отключаются. Когда используется эта функция диаграммы в визуализации или таблице, сортировка визуализации сбрасывается до сортированных входных данных этой функции. Это ограничение не распространяется на эквивалентную функцию скрипта, если такая есть.

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.
`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.

Примеры	Результаты
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).
match_ field_value	Значение, которое необходимо искать в поле **match_field_name**.

Аргумент	Описание
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
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:

p *ApplyMap (page 1344)*

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.

> *Сортировка по y-значениям в диаграммах или сортировка по столбцам выражений в таблицах не допускается, если в каком-либо из выражений диаграммы используется эта функция диаграммы. Данные возможности сортировки автоматически отключаются. Когда используется эта функция диаграммы в визуализации или таблице, сортировка визуализации сбрасывается до сортированных входных данных этой функции.*

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

Region	SubRegion	RowNo()	NoOfRows()	NoOfRows (Total)
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:

p *RowNo – функция диаграммы (page 595)*

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

Код сотрудника	StartDate	EndDate	FirstCode	LastCode
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

ID	Список	Значение
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 1351)*
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.
- Сортировка по y-значениям в диаграммах или сортировка по столбцам выражений в таблицах не допускается, если в каком-либо из выражений диаграммы используется эта функция диаграммы. Данные возможности сортировки автоматически отключаются. Когда используется эта функция диаграммы в визуализации или таблице, сортировка визуализации сбрасывается до сортированных входных данных этой функции.

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:

p *Bottom – функция диаграммы (page 1304)*
p *Above – функция диаграммы (page 1295)*
p *Sum – функция диаграммы (page 355)*
p *RangeAvg (page 1354)*
p *Функции над выборкой (page 1351)*

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 число

Ограничения:

- Функция **SecondaryDimensionality** всегда возвращает 0, за исключением случаев использования в сводных таблицах.
- Сортировка по y-значениям в диаграммах или сортировка по столбцам выражений в таблицах не допускается, если в каком-либо из выражений диаграммы используется эта функция диаграммы. Данные возможности сортировки автоматически отключаются. Когда используется эта функция диаграммы в визуализации или таблице, сортировка визуализации сбрасывается до сортированных входных данных этой функции.

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 ]])
```

Сортировка по y-значениям в диаграммах или сортировка по столбцам выражений в таблицах не допускается, если в каком-либо из выражений диаграммы используется эта функция диаграммы. Данные возможности сортировки автоматически отключаются. Когда используется эта функция диаграммы в визуализации или таблице, сортировка визуализации сбрасывается до сортированных входных данных этой функции.

Данная функция возвращает значение NULL во всех типах диаграмм, кроме сводных таблиц.

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 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.

Сортировка по y-значениям в диаграммах или сортировка по столбцам выражений в таблицах не допускается, если в каком-либо из выражений диаграммы используется эта функция диаграммы. Данные возможности сортировки автоматически отключаются. Когда используется эта функция диаграммы в визуализации или таблице, сортировка визуализации сбрасывается до сортированных входных данных этой функции.

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.

> *Сортировка по y-значениям в диаграммах или сортировка по столбцам выражений в таблицах не допускается, если в каком-либо из выражений диаграммы используется эта функция диаграммы. Данные возможности сортировки автоматически отключаются. Когда используется эта функция диаграммы в визуализации или таблице, сортировка визуализации сбрасывается до сортированных входных данных этой функции.*

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.

Сортировка по y-значениям в диаграммах или сортировка по столбцам выражений в таблицах не допускается, если в каком-либо из выражений диаграммы используется эта функция диаграммы. Данные возможности сортировки автоматически отключаются. Когда используется эта функция диаграммы в визуализации или таблице, сортировка визуализации сбрасывается до сортированных входных данных этой функции.

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.

Сортировка по y-значениям в диаграммах или сортировка по столбцам выражений в таблицах не допускается, если в каком-либо из выражений диаграммы используется эта функция диаграммы. Данные возможности сортировки автоматически отключаются. Когда используется эта функция диаграммы в визуализации или таблице, сортировка визуализации сбрасывается до сортированных входных данных этой функции.

Пример:

```
if( ColumnNo( )=1, 0, sum( Sales ) / before( sum( Sales )))
```

NoOfColumns – функция диаграммы

Функция **NoOfColumns()** возвращает количество столбцов в текущем сегменте строки сводной таблицы.

> *Сортировка по y-значениям в диаграммах или сортировка по столбцам выражений в таблицах не допускается, если в каком-либо из выражений диаграммы используется эта функция диаграммы. Данные возможности сортировки автоматически отключаются. Когда используется эта функция диаграммы в визуализации или таблице, сортировка визуализации сбрасывается до сортированных входных данных этой функции.*

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:

p *Функции между записями (page 1291)*

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:

p *Avg – функция диаграммы (page 407)*
p *Count – функция диаграммы (page 360)*

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:

p *Correl – функция диаграммы (page 410)*

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:

p *Count – функция диаграммы (page 360)*

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:

p *Above – функция диаграммы (page 1295)*
p *Fractile – функция диаграммы (page 414)*

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:

p *Функции между записями (page 1291)*

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:

p *Kurtosis – функция диаграммы (page 422)*

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:

p *MaxString – функция диаграммы (page 546)*

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:

p *Min – функция диаграммы (page 346)*

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:

p *MinString – функция диаграммы (page 549)*

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:

p *MissingCount – функция диаграммы (page 364)*

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:

p *Mode – функция диаграммы (page 350)*

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 1389)*.

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:

p *Функции между записями (page 1291)*

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:

p *NullCount – функция диаграммы (page 367)*

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:

p *NumericCount – функция диаграммы (page 370)*

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:

p *Only – функция диаграммы (page 352)*

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:

p *Skew – функция диаграммы (page 456)*

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:

p *Stdev – функция диаграммы (page 459)*

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:

p *Sum – функция диаграммы (page 355)*
p *Above – функция диаграммы (page 1295)*

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:

p *TextCount – функция диаграммы (page 374)*

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:

p *RangeIRR (page 1363)*

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:

p *Sum – функция диаграммы (page 355)*

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 1408)* и *KMeansND – функция диаграммы (page 1421)*. 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 1434)* и *KMeansCentroidND – функция диаграммы (page 1436)*.

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 1405)

Заголовок: DistributorData

Общее количество записей: 118

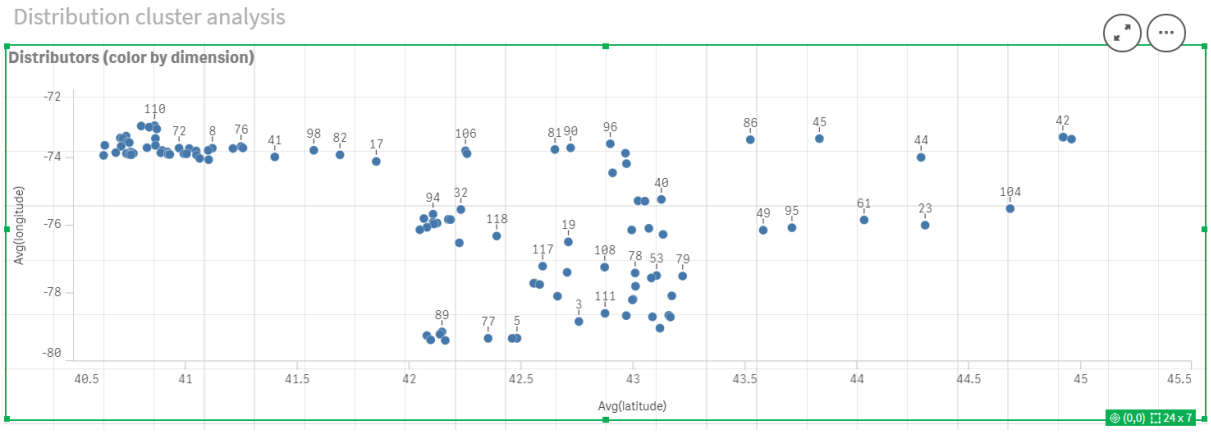
Применение функции KMeans2D

В этом примере демонстрируется настройка **точечной диаграммы** с использованием набора данных *DistributorData*, применяется функция **KMeans2D** и диаграмма раскрашивается по измерению.

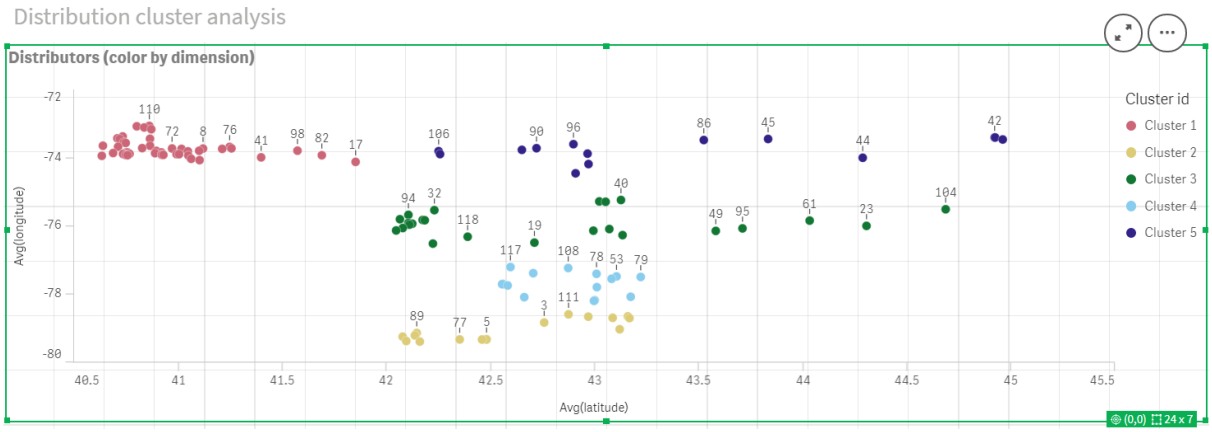
Обратите внимание, что функции k-средних Qlik Sense поддерживают автоматическую кластеризацию с помощью метода, называемого разницей глубины (DeD). Когда пользователь задает количество кластеров равным 0, оптимальное количество кластеров определяется для набора данных. Однако в данном примере переменная создается для аргумента **num_clusters** (синтаксис см. в разделе *KMeans2D – функция диаграммы (page 1408)*). 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

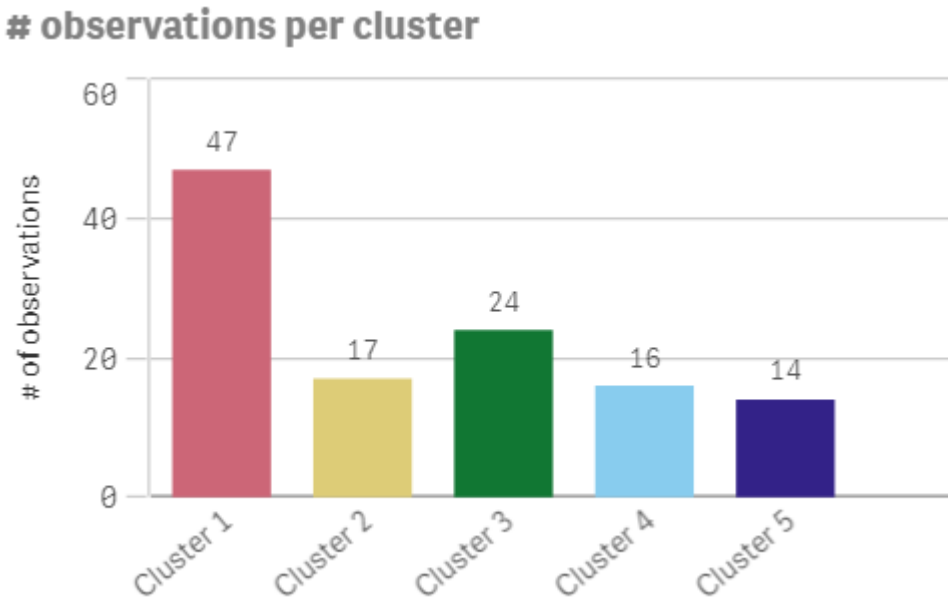
id	first_name	last_name
1	Kaiya	Snow
2	Dean	Roy
3	Eden	Paul
4	Bryanna	Higgins
5	Elisabeth	Lee
6	Skylar	Robinson
7	Cody	Bailey
8	Dario	Sims
9	Deacon	Hood

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

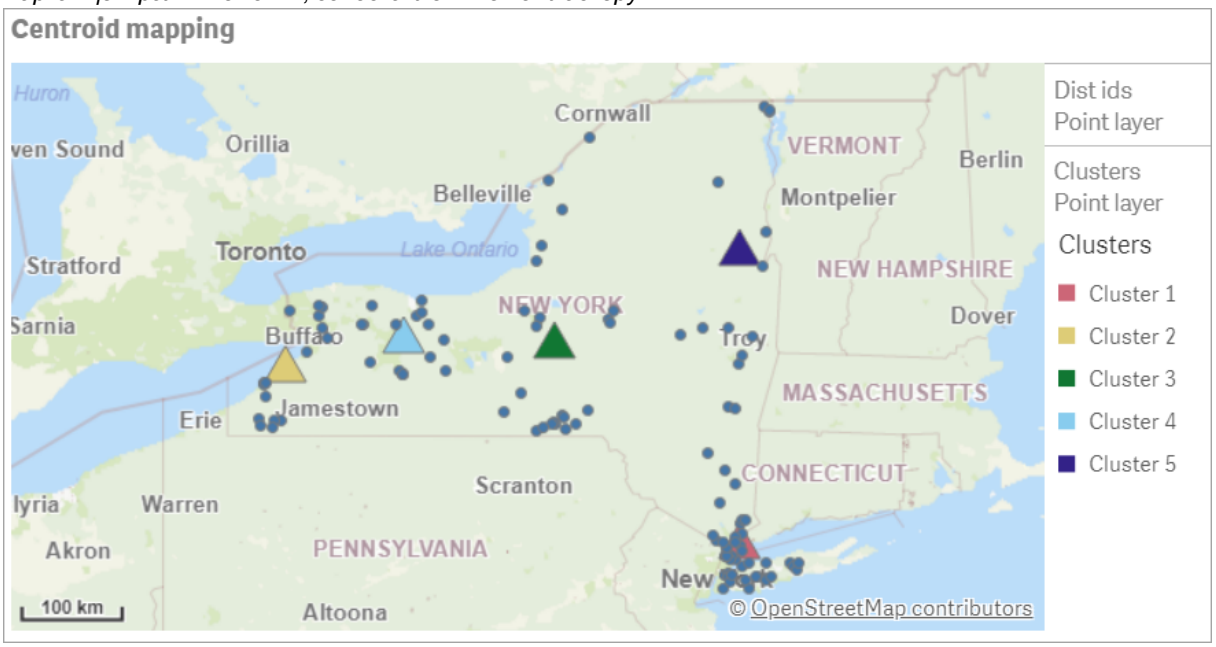
На следующем этапе выполняется сопоставление центральных точек. Разработчик приложения на свое усмотрение принимает решение о том, следует ли располагать визуализацию на отдельных листах.

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**.

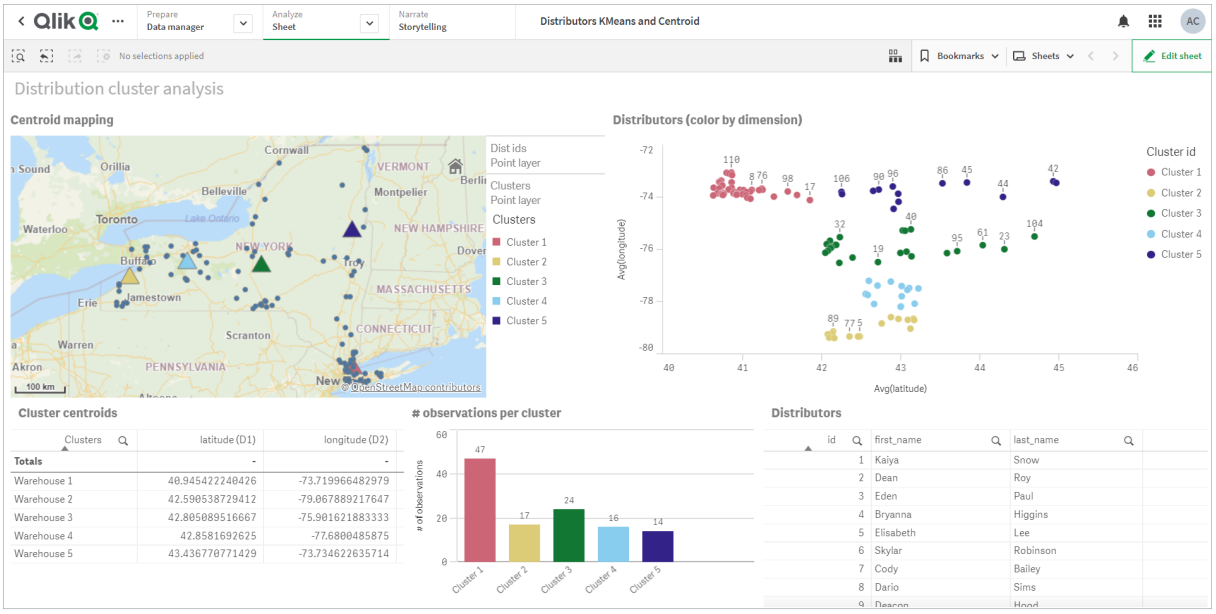
Карта: Центральные точки, сопоставленные по кластеру



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.

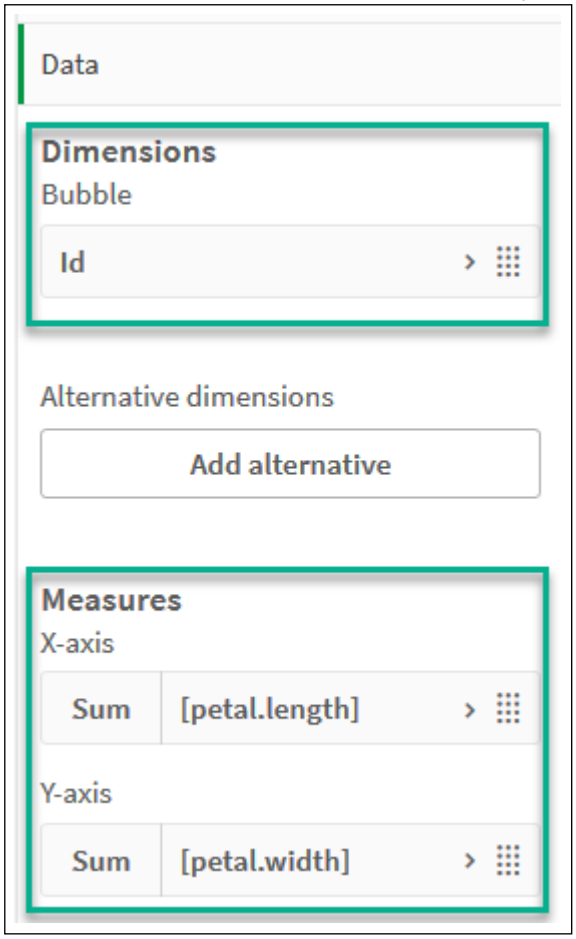
Также мы создаем переменную для аргумента *num_clusters* и затем используем поле ввода переменной, чтобы изменить количество кластеров.

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.

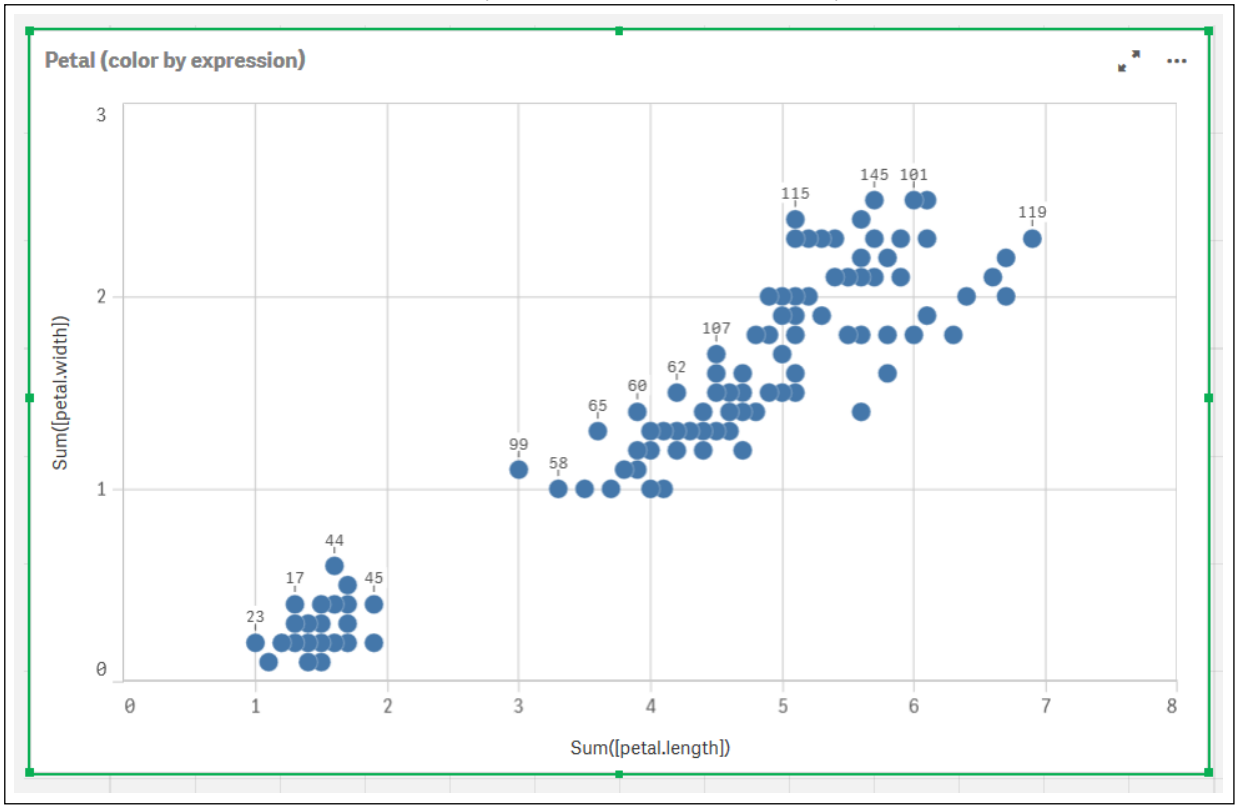
1. Перетащите **точечную диаграмму** на новый лист. Назовите диаграмму *Лепесток (раскрашивание по выражению)*.
2. Создайте переменную для указания количества кластеров. Для переменной **Имя** введите *KmeansPetalClusters*. Для переменной **Определение** введите *=2*.
3. Настройте **данные** для диаграммы.
    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.

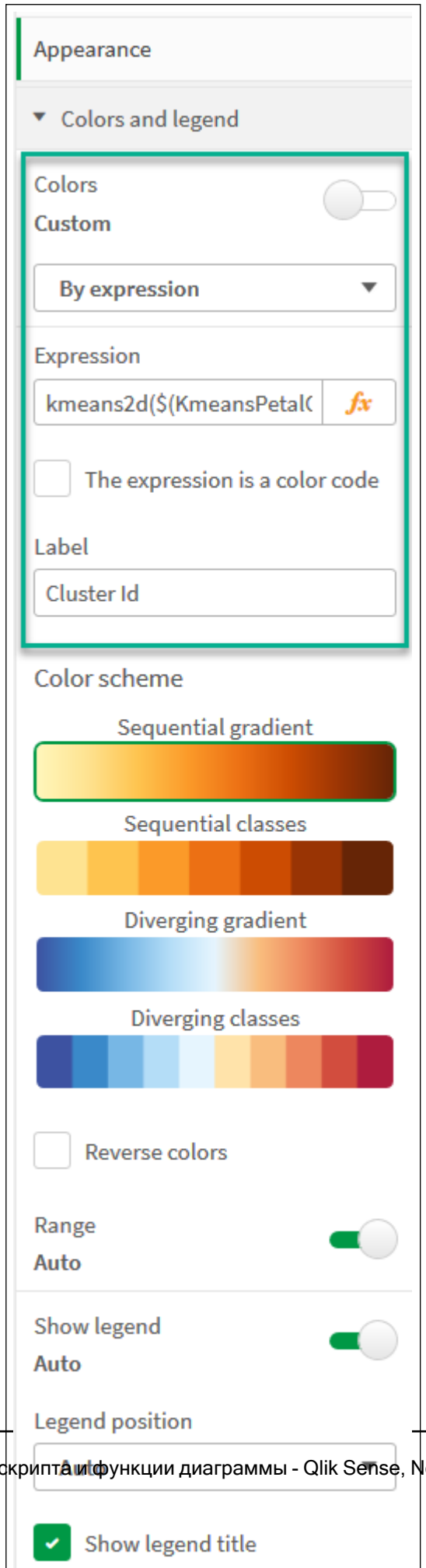
Точки диаграммы на диаграмме Лепесток (раскрашивание по выражению)



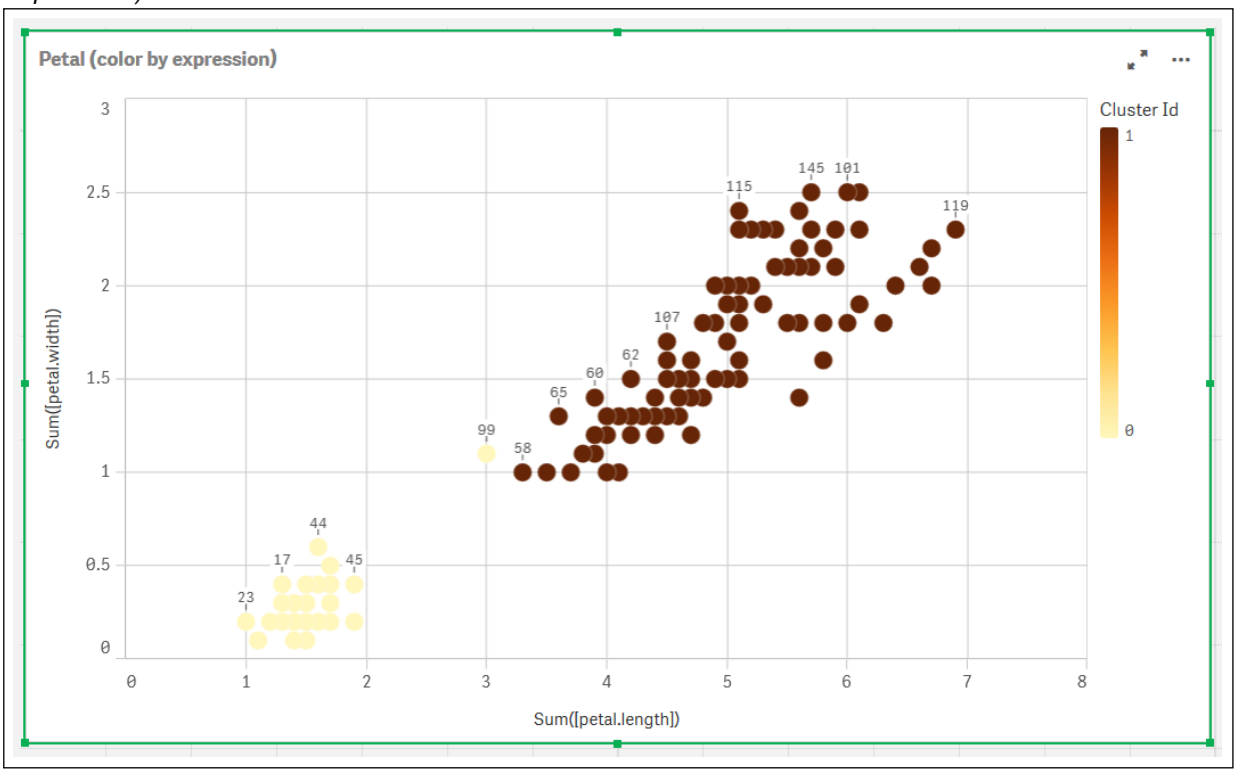
4. Настройте **Вид** диаграммы:
   i. В разделе **Цвета и легенда** выберите **Пользовательский** для параметра **Цвета**.
   ii. Выберите раскрашивание диаграммы **По выражению**.
   iii. Введите следующее для **выражения**: *kmeans2d($(KmeansPetalClusters), Sum ([petal.length]), Sum([petal.width]))*
   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
Auto
Show legend title

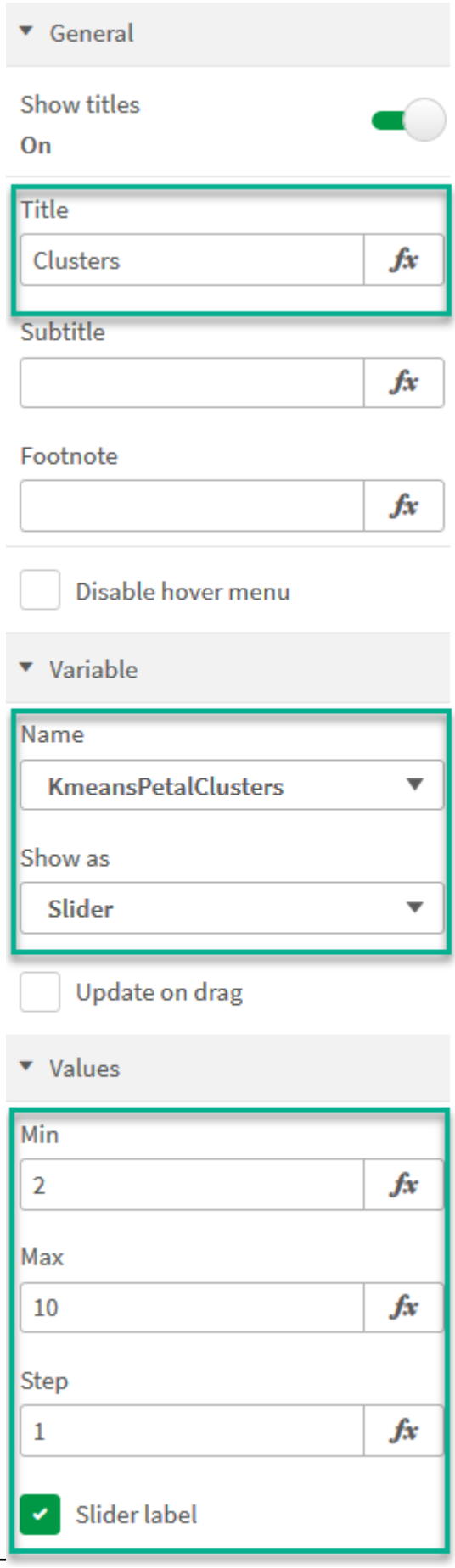
Два кластера на диаграмме раскрашены по выражению KMeans.
Кластеры, раскрашенные по выражению на диаграммы Лепесток (раскрашивание по выражению)



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**.

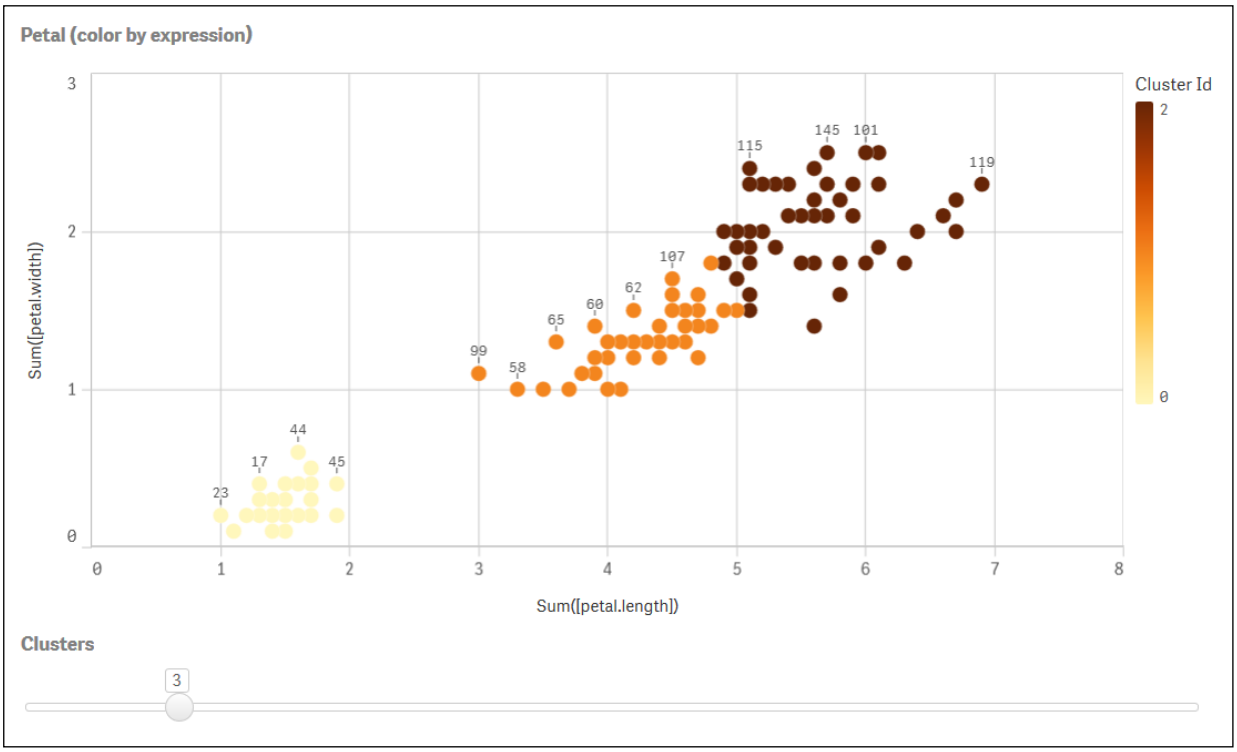
viii. Выберите **Значения** и настройте параметры как требуется,

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*.

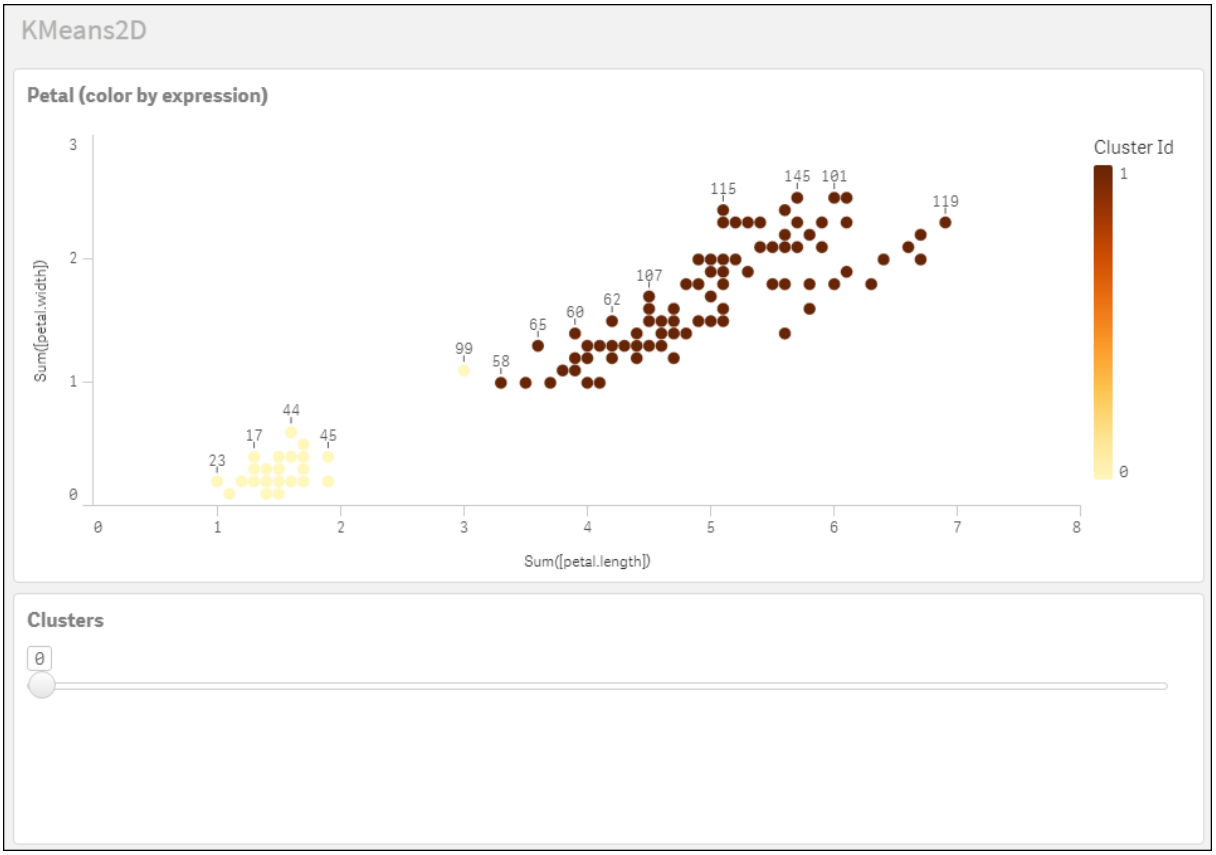
Кластеры, раскрашенные по выражению на диаграммы Лепесток (раскрашивание по выражению)



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.

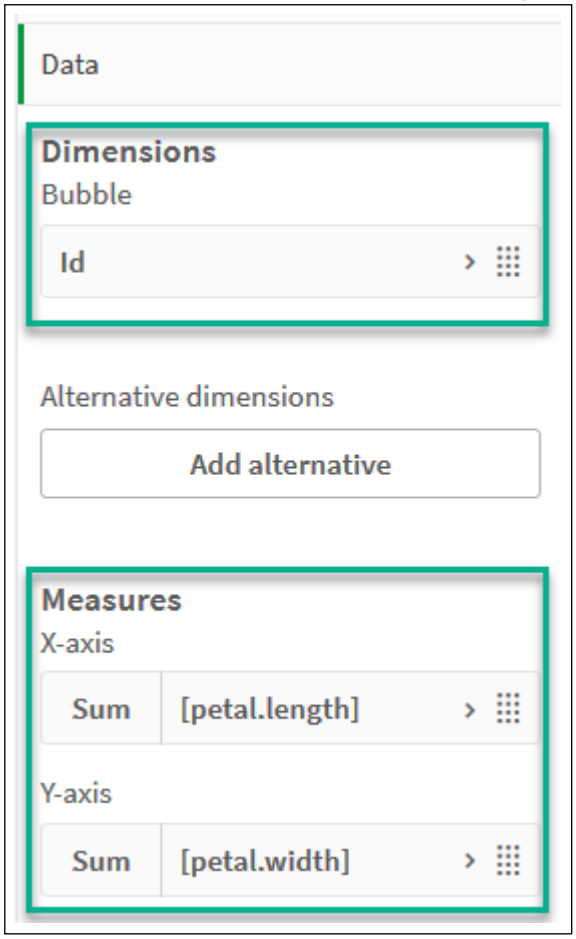
Кроме того, мы создаем переменную для аргумента *num_iter* и затем используем второе поле ввода переменной, чтобы изменить количество итераций.

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.

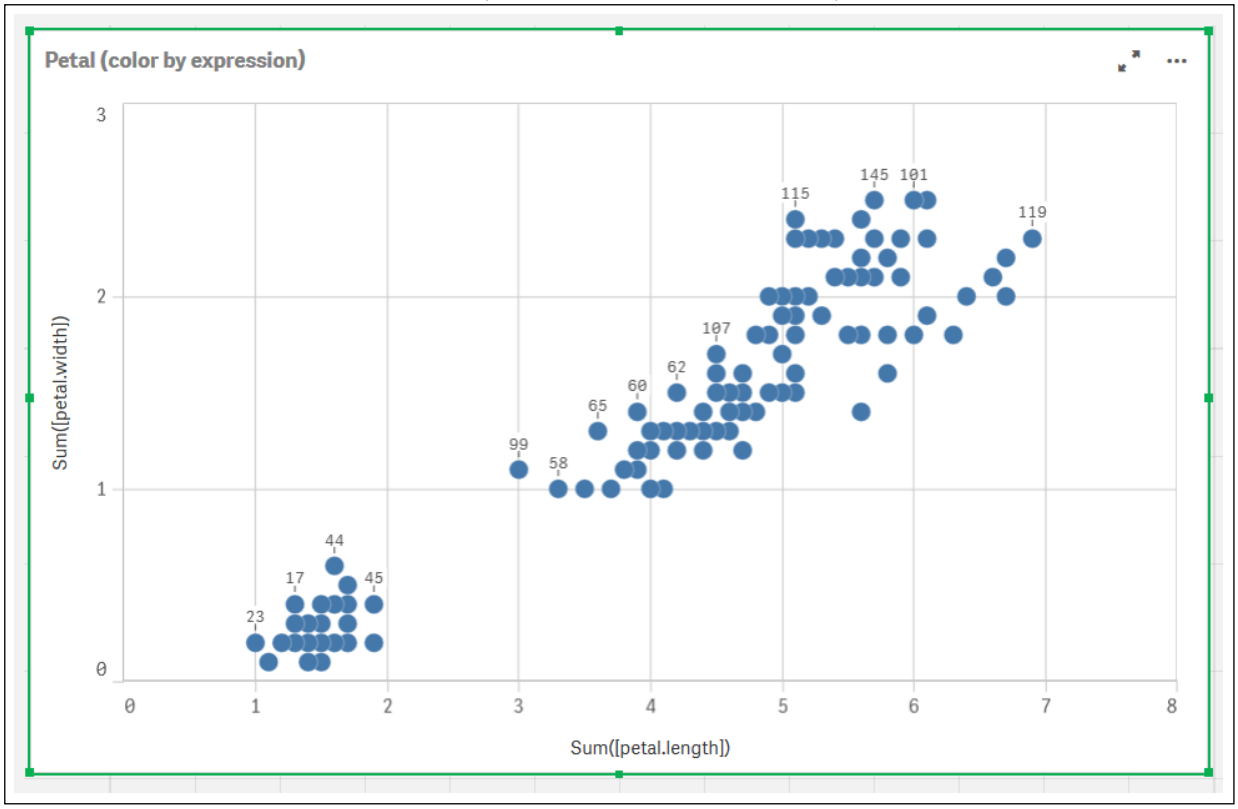
1. Перетащите **точечную диаграмму** на новый лист. Назовите диаграмму *Лепесток (раскрашивание по выражению)*.
2. Создайте переменную для указания количества кластеров. Для переменной **Имя** введите *KmeansPetalClusters*. Для переменной **Определение** введите *=2*.
3. Создайте переменную для указания 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.

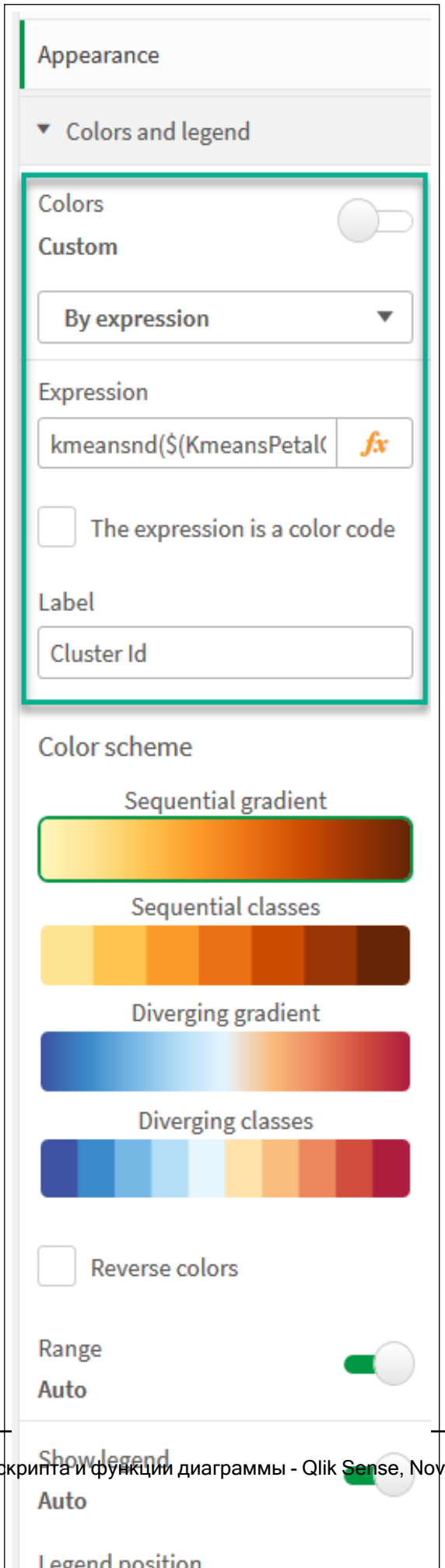
Точки диаграммы на диаграмме Лепесток (раскрашивание по выражению)



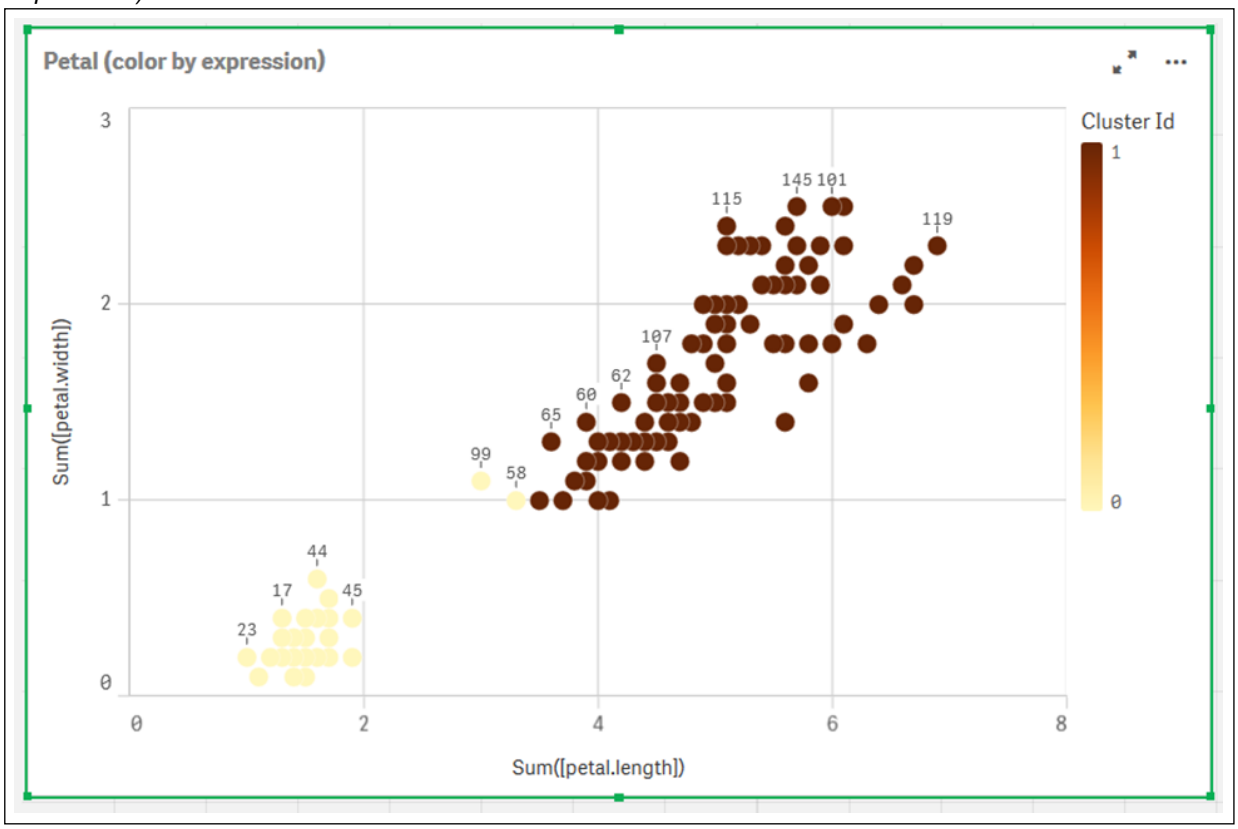
5. Настройте **Вид** диаграммы:
   i. В разделе **Цвета и легенда** выберите **Пользовательский** для параметра **Цвета**.
   ii. Выберите раскрашивание диаграммы **По выражению**.
   iii. Введите следующее для **выражения**: *kmeansnd ($(KmeansPetalClusters),$(KmeansNumberIterations), Sum([petal.length]), 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

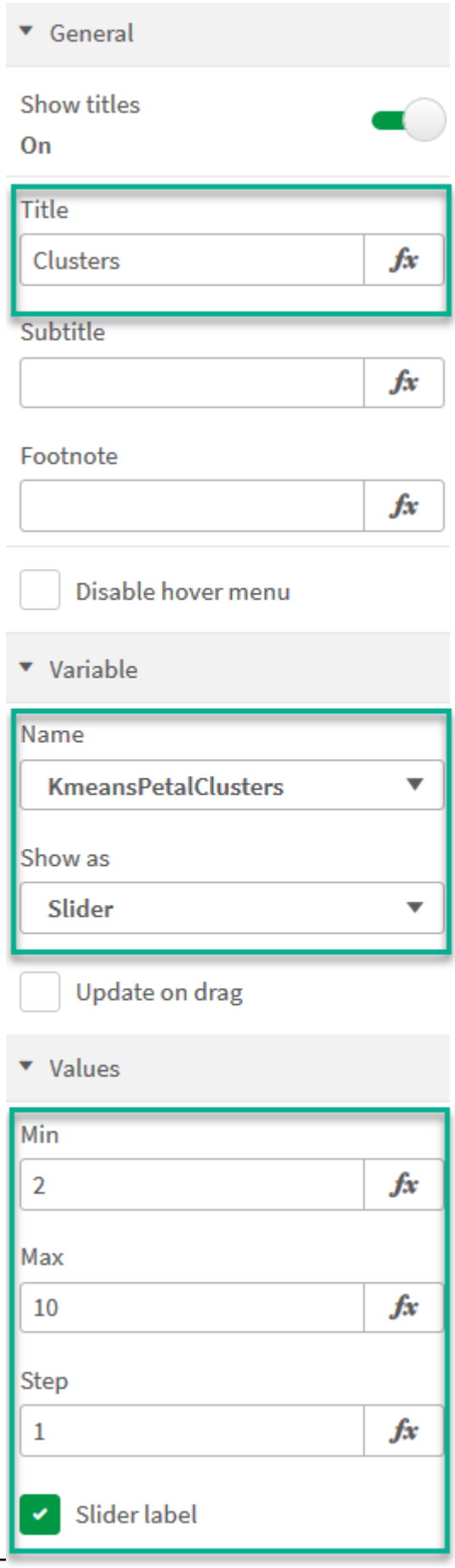
Два кластера на диаграмме раскрашены по выражению KMeans.
Кластеры, раскрашенные по выражению на диаграммы Лепесток (раскрашивание по выражению)



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,

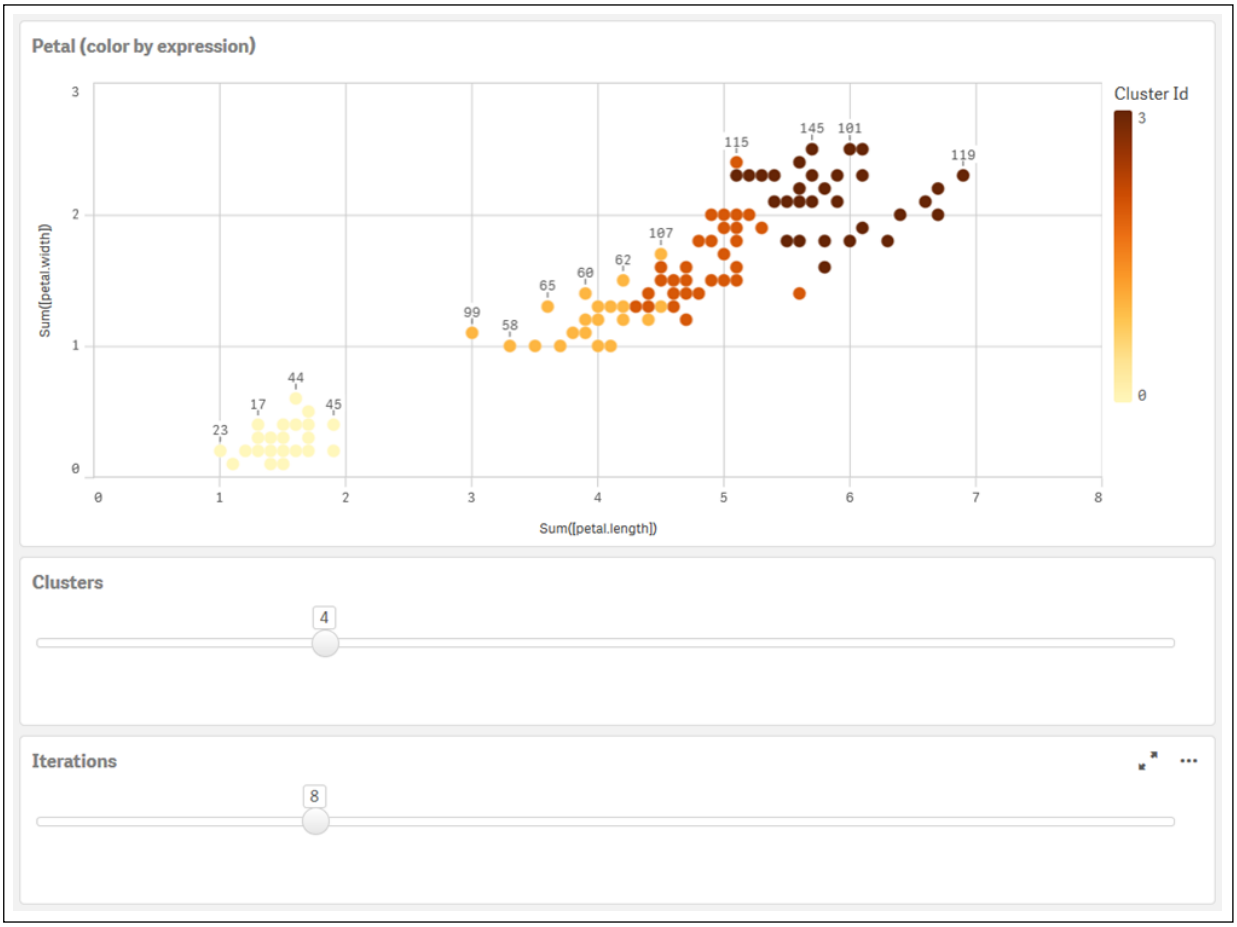
Вид поля ввода переменной Кластеры


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)

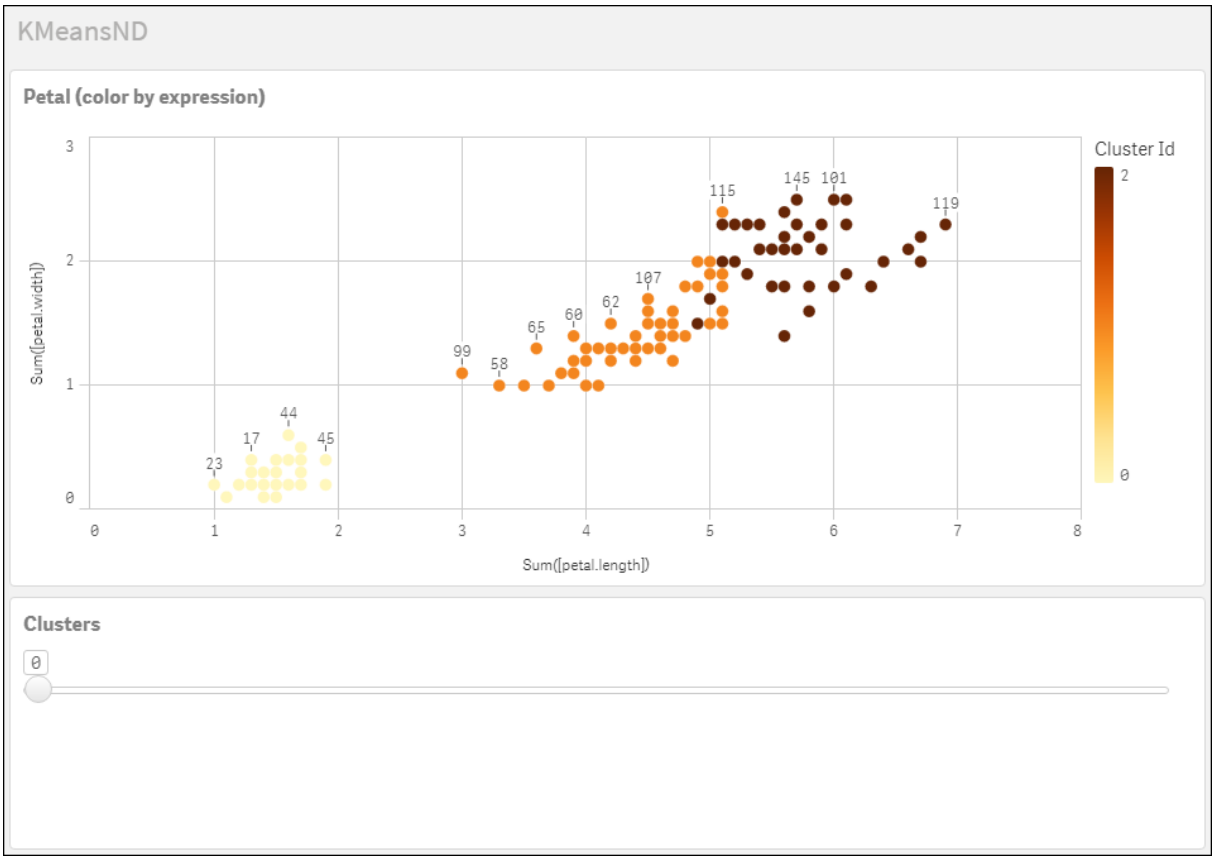


Автоматическая кластеризация

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 возвращают вероятность получения различных возможных результатов для заданной входной переменной. Эти функции можно использовать для расчета возможных значений точек диаграммы.

Три группы описанных ниже функций статистического распределения реализованы в программе Qlik Sense с помощью библиотеки функций Cephes. Ссылки и подробную информацию об используемых алгоритмах, точности и т. д. см. на веб-странице ≤ Cephes library. Библиотека функций Cephes используется с разрешения.

- Функции вероятности вычисляют вероятность в точке распределения, заданной предоставленным значением.
    - Плотность вероятности используется для дискретных распределений.
    - Плотность распределения используется для непрерывных функций.
- Функции распределения вычисляют накопленную вероятность в точке распределения, заданной предоставленным значением.
- Функции Inv вычисляют обратное значение, учитывая накопленную вероятность распределения.

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.

BetaDensity

`BetaDensity()` возвращает вероятность бета-распределения.

```
BetaDensity (value, alpha, beta)
```

BetaDist

`BetaDist()` возвращает накопленную вероятность бета-распределения.

```
BetaDist (value, alpha, beta)
```

BetaInv

`BetaINV()` возвращает обратную накопленную вероятность бета-распределения.

```
BetaInv (prob, alpha, beta)
```

BinomDist

`BinomDist()` возвращает накопленную вероятность биномиального распределения.

```
BinomDist (value, trials, trial_probability)
```

BinomFrequency

`BinomFrequency()` возвращает распределение биноминальной вероятности.

```
BinomFrequency (value, trials, trial_probability)
```

BinomInv

`BinomInv()` возвращает обратную накопленную вероятность биноминального распределения.

```
BinomInv (prob, trials, trial_probability)
```

ChiDensity

`ChiDensity()` возвращает одностороннюю вероятность хи^2-распределения. Функция плотности хи^2 связана с тестом хи^2.

```
ChiDensity (value, degrees_freedom)
```

ChiDist

`ChiDist()` возвращает одностороннюю вероятность хи^2-распределения. хи^2-распределение связано с тестом хи^2.

```
ChiDist (value, degrees_freedom)
```

ChiInv

`ChiInv()` возвращает обратную одностороннюю вероятность распределения chi^2.

```
ChiInv (prob, degrees_freedom)
```

FDensity

`FDensity()` возвращает вероятность `F`-распределения.

```
FDensity (value, degrees_freedom1, degrees_freedom2)
```

FDist

`FDist()` возвращает накопленную вероятность `F`-распределения.

```
FDist (value, degrees_freedom1, degrees_freedom2)
```

FInv

`FInv()` возвращает обратную накопленную вероятность `F`-распределения.

```
FInv (prob, degrees_freedom1, degrees_freedom2)
```

GammaDensity

`GammaDensity()` возвращает вероятность гамма-распределения.

```
GammaDensity (value, k, θ)
```

GammaDist

`GammaDist()` возвращает накопленную вероятность гамма-распределения.

```
GammaDist (value, k, θ)
```

GammaInv

`GammaInv()` возвращает обратную накопленную вероятность гамма-распределения.

```
GammaInv (prob, k, θ)
```

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)
```

PoissonDist

`PoissonDist()` возвращает накопленную вероятность распределения по закону Пуассона.

```
PoissonDist (value, mean)
```

PoissonFrequency

`PoissonFrequency()` возвращает распределение вероятности по закону Пуассона.

```
PoissonFrequency (value, mean)
```

PoissonInv

`PoissonInv()` возвращает обратную накопленную вероятность распределения по закону Пуассона.

```
PoissonInv (prob, mean)
```

TDensity

`TDensity()` возвращает значение функции плотности `t` распределения вероятности Стьюдента, где числовое значение является вычисляемым значением `t`, требующим расчета вероятности.

```
TDensity (value, degrees_freedom, tails)
```

TDist

`TDist()` возвращает распределение вероятности Стьюдента (`t`-критерий), при котором числовое значение является вычисляемым значением `t`, требующим расчета вероятности.

```
TDist (value, degrees_freedom, tails)
```

TInv

`TInv()` возвращает `t`-критерий `t`-распределения вероятности Стьюдента в виде функции вероятности и степеней свободы.

```
TInv (prob, degrees_freedom)
```

См. также:

p *Функции статистического агрегирования (page 399)*

BetaDensity

`BetaDensity()` возвращает вероятность бета-распределения.

Синтаксис:

```
BetaDensity(value, alpha, beta)
```

Возвращаемые типы данных: число

Аргументы

Аргумент	Описание
`value`	Значение, при котором необходимо оценить распределение. Значение должно быть от 0 до 1.
`alpha`	Положительное число, определяющее первый параметр формы. Это экспонента случайной переменной
`beta`	Положительное число, определяющее второй параметр фигуры. Оно указывает число степеней свободы знаменателя.

BetaDist

`BetaDist()` возвращает накопленную вероятность бета-распределения.

Синтаксис:

```
BetaDist(value, alpha, beta)
```

Возвращаемые типы данных: число

Аргументы

Аргумент	Описание
`value`	Значение, при котором необходимо оценить распределение. Значение должно быть от 0 до 1.
`alpha`	Положительное число, определяющее первый параметр формы. Это экспонента случайной переменной
`beta`	Положительное число, определяющее второй параметр формы. Это экспонента, которая управляет формой распределения.

Эта функция связана с функцией `BetaInv` следующим образом:
`If prob = BetaDist(value, alpha, beta), then BetaInv(prob, alpha, beta) = value`

BetaInv

`BetaINV()` возвращает обратную накопленную вероятность бета-распределения.

Синтаксис:

```
BetaInv(prob, alpha, beta)
```

Возвращаемые типы данных: число

Аргументы

Аргумент	Описание
`prob`	Вероятность, связанная с распределением бета-вероятности. Значение должно быть числом от 0 до 1.
`alpha`	Положительное число, определяющее первый параметр формы. Это экспонента случайной переменной
`beta`	Положительное число, определяющее второй параметр формы. Это экспонента, которая управляет формой распределения.

Эта функция связана с функцией `BetaDist` следующим образом:
`If prob = BetaDist(value, alpha, beta), then BetaInv(prob, alpha, beta) = value`

BinomDist

`BinomDist()` возвращает накопленную вероятность биномиального распределения.

Синтаксис:

```
BinomDist(value, trials, trial_probability)
```

Возвращаемые типы данных: число

Аргументы

Аргумент	Описание
`value`	Значение, при котором необходимо оценить распределение. Значение должно быть целым числом не меньше нуля и не больше количества испытаний.
`trials`	Положительное целое число, обозначающее количество испытаний.
`trial_probability`	Вероятность успеха для каждого испытания. Это всегда число от 0 до 1.

Эта функция связана с функцией `BinomInv` следующим образом:
`If prob = BinomDIST(value, trials, trial_probability), then BinomInv(prob, trials, trial_probability) = value`

BinomFrequency

`BinomFrequency()` возвращает распределение биноминальной вероятности.

Синтаксис:

```
BinomFrequency(value, trials, trial_probability)
```

Возвращаемые типы данных: число

Аргументы

Аргумент	Описание
value	Значение, при котором необходимо оценить распределение. Значение должно быть целым числом не меньше нуля и не больше количества испытаний.
trials	Положительное целое число, обозначающее количество испытаний
trial_probability	Вероятность успеха для каждого испытания. Это всегда число от 0 до 1.

BinomInv

`BinomInv()` возвращает обратную накопленную вероятность биноминального распределения.

Синтаксис:

```
BinomInv(prob, trials, trial_probability)
```

Возвращаемые типы данных: число

Аргументы

Аргумент	Описание
prob	Вероятность, связанная с распределением биноминальной вероятности. Значение должно быть числом от 0 до 1.
trials	Положительное целое число, обозначающее количество испытаний.
trial_probability	Вероятность успеха для каждого испытания. Это всегда число от 0 до 1.

Эта функция связана с функцией `BinomDist` следующим образом:
`If prob = BinomDist(value, trials, trial_probability), then BinomInv(prob, trials, trial_probability) = value`

ChiDensity

`ChiDensity()` возвращает одностороннюю вероятность хи^2-распределения. Функция плотности хи^2 связана с тестом хи^2.

Синтаксис:

```
ChiDensity(value, degrees_freedom)
```

Возвращаемые типы данных: число

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 числителя.

ChiDist

ChiDist() возвращает одностороннюю вероятность хи^2-распределения. хи^2-распределение связано с тестом хи^2.

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

FDensity

`FDensity()` возвращает вероятность `F`-распределения.

Синтаксис:

```
FDensity(value, degrees_freedom1, degrees_freedom2)
```

Возвращаемые типы данных: число

Аргументы

Аргумент	Описание
`value`	Значение, при котором необходимо оценить распределение. Значение не должно быть отрицательным.
`degrees_ freedom1`	Положительное целое число, которое указывает число степеней свободы числителя.
`degrees_ freedom2`	Положительное целое число, которое указывает число степеней свободы знаменателя.

FDist

`FDist()` возвращает накопленную 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. Значение не должно 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 накопленную 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

GammaDensity

`GammaDensity()` возвращает вероятность гамма-распределения.

Синтаксис:

```
GammaDensity(value, k, θ)
```

Возвращаемые типы данных: число

Аргументы

Аргумент	Описание
`value`	Значение, при котором необходимо оценить распределение. Значение не должно быть отрицательным.
`k`	Положительное число, определяющее параметр формы.
`θ`	Положительное число, определяющее параметр шкалы.

GammaDist

`GammaDist()` возвращает накопленную вероятность гамма-распределения.

Синтаксис:

```
GammaDist(value, k, θ)
```

Возвращаемые типы данных: число

Аргументы

Аргумент	Описание
value	Значение, при котором необходимо оценить распределение. Значение не должно быть отрицательным.
k	Положительное число, определяющее параметр формы.
θ	Положительное число, определяющее параметр шкалы.

Эта функция связана с функцией GammaINV следующим образом:
`If prob = GammaDist(value, k, θ), then GammaInv(prob, k, θ) = value`

GammaInv

`GammaInv()` возвращает обратную накопленную вероятность гамма-распределения.

Синтаксис:

```
GammaInv(prob, k, θ)
```

Возвращаемые типы данных: число

Аргументы

Аргумент	Описание
prob	Вероятность, связанная с распределением гамма-вероятности. Значение должно быть числом от 0 до 1.
k	Положительное число, определяющее параметр формы.
θ	Положительное число, определяющее параметр шкалы.

Эта функция связана с функцией GammaDist следующим образом:
`If prob = GammaDist(value, k, θ), then GammaInv(prob, k, θ) = value`

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

PoissonDist

`PoissonDist()` возвращает накопленную вероятность распределения по закону Пуассона.

Синтаксис:

```
PoissonDist(value, mean)
```

Возвращаемые типы данных: число

Аргументы

Аргумент	Описание
`value`	Значение, при котором необходимо оценить распределение. Значение не должно быть отрицательным.
`mean`	Положительное число, определяющее средний результат.

Эта функция связана с функцией `PoissonInv` следующим образом:
`If prob = PoissonDist(value, mean), then PoissonInv(prob, mean) = value`

PoissonFrequency

`PoissonFrequency()` возвращает распределение вероятности по закону Пуассона.

Синтаксис:

```
PoissonFrequency(value, mean)
```

Возвращаемые типы данных: число

Аргументы

Аргумент	Описание
`value`	Значение, при котором необходимо оценить распределение. Значение не должно быть отрицательным.
`mean`	Положительное число, определяющее средний результат.

PoissonInv

`PoissonInv()` возвращает обратную накопленную вероятность распределения по закону Пуассона.

Синтаксис:

```
PoissonInv(prob, mean)
```

Возвращаемые типы данных: число

Аргументы

Аргумент	Описание
`prob`	Вероятность, связанная с распределением вероятности по закону Пуассона. Значение должно быть числом от 0 до 1.
`mean`	Положительное число, определяющее средний результат.

Эта функция связана с функцией `PoissonDIST` следующим образом:
`If prob = PoissonDist(value, mean), then PoissonInv(prob, mean) = value`

TDensity

`TDensity()` возвращает значение функции плотности `t` распределения вероятности Стьюдента, где числовое значение является вычисляемым значением `t`, требующим расчета вероятности.

Синтаксис:

```
TDensity(value, degrees_freedom)
```

Возвращаемые типы данных: число

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.

TDist

`TDist()` возвращает распределение вероятности Стьюдента (`t`-критерий), при котором числовое значение является вычисляемым значением `t`, требующим расчета вероятности.

Синтаксис:

```
TDist(value, degrees_freedom, tails)
```

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.
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`-критерий `t`-распределения вероятности Стьюдента 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])
```

IsJson

IsJson() проверяет, содержит ли указанная строка допустимые данные JSON (JavaScript Object Notation). Также можно проверить конкретный тип данных JSON.

```
IsJson (json [, type])
```

JsonGet

JsonGet() возвращает путь к строке данных JSON (JavaScript Object Notation). Данные должны быть допустимым кодом JSON, но могут содержать дополнительные пробелы и символы новой строки.

```
JsonGet (json, path)
```

JsonSet

JsonSet() изменяет строку, содержащую данные JSON (JavaScript Object Notation). Может задать или вставить значение JSON с указанием нового местоположения в пути. Данные должны быть допустимым кодом JSON, но могут содержать дополнительные пробелы и символы новой строки.

```
JsonSet(json, path, value)
```

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

Строка	Capitalize(String)
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.
`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.

Аргумент	Описание
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 ];
```

IsJson

IsJson() проверяет, содержит ли указанная строка допустимые данные JSON (JavaScript Object Notation). Также можно проверить конкретный тип данных JSON.

Синтаксис:

```
value IsJson(json [, type])
```

Возвращаемые типы данных: двойное значение

Аргументы

Аргумент	Описание
json	Тестируемая строка. Может содержать дополнительные пробелы и символы новой строки.

Аргумент	Описание
type	Дополнительный аргумент, задающий тип данных JSON для тестирования. • 'value' (по умолчанию) • 'object' • 'array' • 'строка' • 'number' • 'Boolean' • 'null'

Пример: Допустимые JSON и тип

Пример	Результат
`IsJson('null')`	Возвращает -1 (true)
`IsJson('"abc"', 'value')`	Возвращает -1 (true)
`IsJson('"abc"', 'string')`	Возвращает -1 (true)
`IsJson(123, 'number')`	Возвращает -1 (true)

Пример: Недопустимый JSON или тип

Пример	Результат	Описание
`IsJson('text')`	Возвращает 0 (false)	'text' не является допустимым значением JSON
`IsJson('"text"', 'number')`	Возвращает 0 (false)	'"text"' не является допустимым числом JSON
`IsJson('"text"', 'text')`	Возвращает 0 (false)	'text' не является допустимым типом JSON

JsonGet

JsonGet() возвращает путь к строке данных JSON (JavaScript Object Notation). Данные должны быть допустимым кодом JSON, но могут содержать дополнительные пробелы и символы новой строки.

Синтаксис:

```
value JsonGet(json, path)
```

Возвращаемые типы данных: двойное значение

Аргументы

Аргумент	Описание
json	Строка, содержащая данные JSON.
path	Путь должен быть указан в соответствии с ≤ RFC 6901. Это позволит выполнять поиск свойств внутри данных JSON без применения сложных функций подстрок и индекса.

Пример: Допустимые JSON и путь

Пример	Результат
`JsonGet('{"a":{"foo":"bar"},"b":[123,"abc","ABC"]}', '')`	Возвращает '{"a":{"foo":"bar"},"b":[123,"abc","ABC"]}'
`JsonGet('{"a":{"foo":"bar"},"b":[123,"abc","ABC"]}', '/a')`	Возвращает '{"foo":"bar"}'
`JsonGet('{"a":{"foo":"bar"},"b":[123,"abc","ABC"]}', '/a/foo')`	Возвращает '"bar"'
`JsonGet('{"a":{"foo":"bar"},"b":[123,"abc","ABC"]}', '/b')`	Возвращает '[123,"abc","ABC"]'
`JsonGet('{"a":{"foo":"bar"},"b":[123,"abc","ABC"]}', '/b/0')`	Возвращает '123'
`JsonGet('{"a":{"foo":"bar"},"b":[123,"abc","ABC"]}', '/b/1')`	Возвращает '"abc"'
`JsonGet('{"a":{"foo":"bar"},"b":[123,"abc","ABC"]}', '/b/2')`	Возвращает '"ABC"'

Пример: Недопустимый JSON или путь

Пример	Результат	Описание
`JsonGet('{"a":"b"}','/b')`	Возвращает null	Этот путь не указывает на допустимую часть данных JSON.
`JsonGet('{"a"}','/a')`	Возвращает null	Данные JSON не являются допустимым кодом JSON (член «a» не имеет значения).

JsonSet

JsonSet() изменяет строку, содержащую данные JSON (JavaScript Object Notation). Может задать или вставить значение JSON с указанием нового местоположения в пути. Данные должны быть допустимым кодом JSON, но могут содержать дополнительные пробелы и символы новой строки.

Синтаксис:

```
value JsonSet(json, path, value)
```

Возвращаемые типы данных: двойное значение

Аргументы

Аргумент	Описание
json	Строка, содержащая данные JSON.
path	Путь должен быть указан в соответствии с ≤ RFC 6901. Это позволяет составлять свойства внутри данных JSON без применения сложных функций подстрок и индекса, а также конкатенации.
value	Новое значение строки в формате JSON.

Пример: Допустимые JSON, путь и значение

Пример	Результат
`JsonSet('{}','/a','"b"')`	Возвращает '{"a":"b"}'
`JsonSet('[]','/0','"x"')`	Возвращает '["x"]'
`JsonSet('"abc"','','123')`	Возвращает 123

Пример: Недопустимый JSON, путь или значение

Пример	Результат	Описание
`JsonSet ('"abc"','/x','123')`	Возвращает null	Этот путь не указывает на допустимую часть данных JSON.
`JsonSet('{"a": {"b":"c"}}','a/b','"x"')`	Возвращает null	Недопустимый путь.
`JsonSet(' {"a":"b"}','/a','abc')`	Возвращает null	Значение не является допустимым JSON. Строку необходимо заключить в кавычки.

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:

p *PurgeChar (page 1472)*

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

р См. также *Index (page 1461)* 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 206).*

Результат

Строка	StringLength	LtrimStringLength
def	6	5
abc	10	7

См. также:

p *RTrim (page 1476)*

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:

p *Index (page 1461)*

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:

р *KeepChar (page 1465)*

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 206).*

Результат

Строка	StringLength	RtrimStringLength
def	6	4
abc	10	6

См. также:

p *LTrim (page 1470)*

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 206).

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 1501)

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 в формате UTC для даты и времени последнего изменения 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 заголовка XML из файла QVD при его наличии, в противном случае она возвращает значение NULL.Время в метке времени указано в формате UTC.
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)

6 Скрипты на уровне диаграммы

При изменении данных диаграммы используется подгруппа скрипта Qlik Sense, которая содержит ряд 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.

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.

В этом разделе приводится алфавитный список всех операторов скриптов, операторов управления и префиксов, доступных в подгруппе скрипта, используемого при модификации данных диаграммы.

6.4 Операторы управления

При изменении данных диаграммы используется подгруппа скрипта Qlik Sense, которая содержит ряд 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.

К операторам управления никогда не применяются префиксы.

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

Обзор операторов управления для модификаторов 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.

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 ]
```

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.

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

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]
```

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]
```

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
```

Next

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

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
```

To

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

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.

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.
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**.

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.

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.
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.*

Аргумент	Описание
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.

6.5 Префиксы

Префиксы можно использовать с соответствующими обычными операторами, но не с операторами 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 для модификаторов 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.

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
```

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
```

Add

В контексте модификации диаграммы префикс **Add** используется с **LOAD** для добавления значений в таблицу *HC1*, которая представляет гиперкуб, рассчитанный в Qlik Associative Engine. Можно указать один или несколько столбцов. Qlik Associative Engine автоматически заполняет отсутствующие значения.

Синтаксис:

```
Add loadstatement
```

Пример:

Этот пример добавляет две строки в столбцы *Dates* и *Sales* из встроенного оператора

```
Add Load
x as Dates,
y as Sales
Inline
[
Dates,Sales
2001/09/1,1000
2001/09/10,-300
]
```

Replace

В контексте модификации диаграммы префикс **Replace** изменяет все значения таблицы *HC1* на вычисленные значения, определенные скриптом.

Синтаксис:

```
Replace loadstatement
```

Пример:

Этот пример перезаписывает все значения в столбце `z` суммой `x` и `y`.

```
Replace Load
x+y as z
Resident HC1;
```

6.6 Обычные операторы

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 для модификаторов 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.

LOAD

В контексте модификации диаграммы оператор **LOAD** загружает в гиперкуб дополнительные данные из данных, определенных в скрипте или из предварительно загруженной таблицы. Также можно загружать данные из аналитических подключений.

*Оператор **LOAD** должен иметь префикс **Replace** или **Add**, иначе он будет отклонен.*

```
Add | Replace Load [ distinct ] fieldlist
```

```
(
inline data [ format-spec ] |
resident table-label
) | extension pluginname.functionname([script] tabledescription)]
[ where criterion | while criterion ]
[ group by groupbyfieldlist ]
[order by orderbyfieldlist ]
```

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
```

Set

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

```
Set variablename=string
```

Put

Оператор **Put** используется для задания некоторого числового значения в гиперкубе.

HCValue

Оператор **HCValue** служит для извлечения значений в строке указанного столбца.

Load

В контексте модификации диаграммы оператор **LOAD** загружает в гиперкуб дополнительные данные из данных, определенных в скрипте или из предварительно загруженной таблицы. Также можно загружать данные из аналитических подключений.

*Оператор **LOAD** должен иметь префикс **Replace** или **Add**, иначе он будет отклонен.*

Синтаксис:

```
Add | Replace LOAD fieldlist
(
inline data [ format-spec ] |
resident table-label
) | extension pluginname.functionname([script] tabledescription)]
[ where criterion | while criterion ]
[ group by groupbyfieldlist ]
[order by orderbyfieldlist ]
```

Аргументы:

Аргументы

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
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 **LOAD**, используемому для создания исходной таблицы. В конце метки должно быть указано двоеточие.

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.
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.

Аргумент	Описание
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**.

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.

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;
```

Put

Оператор **put** используется для задания некоторого числового значения в гиперкубе.

Доступ к столбцам может осуществляться по меткам. Также можно получить доступ к столбцам и строкам в порядке объявления. Для получения дополнительных сведений см. примеры ниже.

Синтаксис:

```
put column(position)=value
```

Example 1:

Доступ к столбцам может осуществляться по меткам.

Этот пример задаст значение `1` в первой позиции столбца с меткой *Sales*.

```
Put Sales(1) = 1;
```

Example 2:

Доступ к столбцам мер осуществляется в порядке объявления с использованием формата `#hc1.measure` для мер.

Этот пример возвращает значение 1000 в десятой позиции окончательного сортированного гиперкуба.

```
Put #hc1.measure.2(10) = 1000;
```

Example 3:

Доступ к строкам измерений осуществляется в порядке объявления с использованием формата `#hc1.dimension` для измерений.

Этот пример помещает значение константы «пи» в пятую строку третьего объявленного измерения.

```
Put #hc1.dimension.3(5) = Pi();
```

Если такие измерения или выражения отсутствуют в значении или метках, возвращается ошибка о том, что столбец не найден. Если индекс столбца находится за пределами диапазона, ошибка не отображается.

HCValue

Функция **HCValue** служит для извлечения значений в строке указанного столбца.

Синтаксис:

```
HCValue(column,position)
```

Example 1:

Этот пример возвращает значение в первой позиции столбца с меткой Sales.

```
HCValue(Sales,1)
```

Example 2:

Этот пример возвращает значение в десятой позиции сортированного гиперкуба.

```
HCValue(#hc1.measure.2,10)
```

Example 3:

Этот пример возвращает значение в пятой строке третьего измерения.

```
HCValue(#hc1.dimension.3,5)
```

Если такие измерения или выражения отсутствуют в значении или метках, возвращается ошибка о том, что столбец не найден. Если индекс столбца находится за пределами диапазона, возвращается NULL.

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 1351)*
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 560)*
QlikViewVersion	Используйте **EngineVersion**. *EngineVersion (page 1485)*
ProductVersion	Используйте **EngineVersion**. *EngineVersion (page 1485)*
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.